A New Decoding Method for Reed–Solomon Codes Based on FFT and Modular Approach

Nianqi Tang and Yunghsiang S. Han, *Fellow, IEEE*

***Abstract*— Decoding algorithms for Reed–Solomon (RS) codes are of great interest for both practical and theoretical reasons. In this paper, an efficient algorithm, called the modular approach (MA), is devised for solving the Welch–Berlekamp (WB) key equation. By taking the MA as the key equation solver, we propose a new decoding algorithm for systematic RS codes. For (n, k) RS codes, where n is the code length and k is the code dimension, the proposed decoding algorithm has both the best asymptotic computational complexity $O(n \log(n-k) + (n-k)\log^2(n-k))$ and the smallest constant factor achieved to date. By comparing the number of field operations required, we show that when decoding practical RS codes, the new algorithm is significantly superior to the existing methods in terms of computational complexity. When decoding the (4096, 3584) RS code defined over $\mathbb{F}_{2^{12}}$, the new algorithm is 10 times faster than a conventional syndrome-based method. Furthermore, the new algorithm has a regular architecture and is thus suitable for hardware implementation.**

***Index Terms*— Modular approach, Reed–Solomon codes, fast Fourier transform, decoding algorithm.**

I. Introduction

REED–SOLOMON (RS) codes, first proposed in [1], are the most commonly used error correcting codes and have been widely applied in a variety of communication systems, including storage devices, digital television, and data transmission. Research into the decoding of RS codes is therefore of both practical and theoretical importance. Among the algorithms currently available for decoding RS codes, the most widely known is syndrome-based RS decoding, in which the key equation is solved using either the Berlekamp–Massey (BM) algorithm or the Euclidean algorithm. For an (n, k) RS code, where n is the code length and k is the code dimension, the computational complexity of syndrome-based decoding is $O(n(n-k)+(n-k)^2)$ (see [2], [3], [4] for more details). Here, the computational complexity of an algorithm is expressed using the asymptotic notation O, where $O(p(\varepsilon))$ denotes the set of functions $O(p(\varepsilon)) = \{q(\varepsilon)$: there exist positive constants c and ε_0 such that $0 \leq q(\varepsilon) \leq cp(\varepsilon)$ for all $\varepsilon > \varepsilon_0\}$ in which c is called the constant factor. Note that $q(\varepsilon) \in O(p(\varepsilon))$ represents the real computational complexity of the algorithm or an upper bound on it. Another decoding algorithm with complexity $O(n(n-k)+(n-k)^2)$ was presented in [5], where the syndrome and the error locator polynomial are related by the Welch–Berlekamp (WB) key equation, and the WB algorithm is used for solving this equation.

Much effort has been devoted to designing decoding algorithms with lower complexity by using the fast Fourier transform (FFT) over finite fields. Fedorenko and Trifonov [6] proposed an algorithm for finding roots of polynomials over finite fields that exploits a specific polynomial called the p-polynomial. Based on this algorithm, Lin et al. [7] then presented a fast algorithm for the syndrome calculation. Wu et al. [8] used the partial composite cyclotomic Fourier transform (CFT) to derive fast syndrome-based decoders. Bellini et al. [9] proposed a method to reduce the number of additions required in the CFT, and Fedorenko [10] further reduced the multiplicative complexity of the partial inverse CFT. Gao and Mateer [11] devised an additive FFT algorithm based on a Taylor expansion. Based on the well-known subspace polynomials, Lin et al. [12] proposed an efficient additive FFT and devised a decoding algorithm with complexity $O(n\log(n-k)+(n-k)\log^2(n-k))$ for (n, k) RS codes, which is the best asymptotic computational complexity achieved to date.

After the breakthrough work of Guruswami and Sudan [13], decoding RS codes beyond half of the minimum distance has drawn much attentions. Interpolation algorithms for solving the key equation of the Guruswami-Sudan (GS) algorithm were proposed in [14], [15], [16], [17], and [18]. On the other hand, one-pass Chase decoding algorithms were proposed in [19] and [20], which share computations among the hard-decision decodings of the different test error patterns.

In this paper, we devise an efficient algorithm, the modular approach (MA), for solving the WB key equation. We then show that this approach can be applied to solve the key equation proposed in [12], and we derive a new decoding algorithm for the (n, k) RS codes. Two versions of the MA are presented. The first, the frequency-domain modular approach (FDMA), updates only two polynomials in the frequency domain with complexity $O((n-k)^2)$. It is suitable for decoding short codes. The second, the fast modular approach (FMA), processes in a divide-and-conquer style and has a complexity

Manuscript received 11 February 2022; revised 25 July 2022; accepted 9 October 2022. Date of publication 19 October 2022; date of current version 19 December 2022. The associate editor coordinating the review of this article and approving it for publication was P. Trifonov. *(Corresponding author: Yunghsiang S. Han.)*

Nianqi Tang is with Huawei Technologies Company Ltd., Shenzhen 518129, China.

Yunghsiang S. Han is with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China (e-mail: yunghsiangh@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2022.3215998.

Digital Object Identifier 10.1109/TCOMM.2022.3215998

$O((n-k)\log^2(n-k))$ for arbitrary $n-k$. We show that the new decoding algorithm has both the best asymptotic computational complexity and the smallest constant factor achieved to date. We compare the proposed decoding algorithm with the existing methods by counting the number of field operations. The results show that the new algorithm is significantly superior to these other techniques. More precisely, for a (4096, 3584) RS code, the new algorithm is 10 times faster than conventional syndrome-based RS decoding. Furthermore, this new algorithm has a regular architecture and it is therefore suitable for practical implementation.

The remainder of this paper is organized as follows. Section II provides a detailed discussion of the MA. A new decoding algorithm for RS codes is then proposed in Section III. Next, we compare the new algorithm with other methods from the literature in Section IV. Finally, we conclude the paper in Section V.

II. Modular Approach

In this section, we describe the MA, which is capable of solving the WB key equation. The WB key equation problem can be expressed as follows: Find polynomials $W(x)$ and $N(x)$ with $\deg(N(x)) < \deg(W(x))$ satisfying

$$N(x_i) = W(x_i)y_i, \quad i = 1, 2, \ldots, \rho \tag{1}$$

for a given set of nonzero points $(x_i, y_i), i = 1, 2, \ldots, \rho$, over a field $\mathbb{F}_{2^m}$, where $W(x), N(x) \in \mathbb{F}_{2^m}[x]$. Note that for decoding RS codes, we have $\rho = 2t$, where t is the error correction capability. Without loss of generality, we assume that the x_i are distinct.

Definition 1: The rank of an ordered polynomial pair $(W(x), N(x))$ is defined as

$$\operatorname{rank}[W(x), N(x)] = \max\{2\deg(W(x)), 1 + 2\deg(N(x))\}.$$

Note that the rank of a polynomial pair containing a zero polynomial is dominated by its nonzero component, and we then define $\operatorname{rank}[0, 0] = 0$. It has been shown that there exist two complementary solutions $(W(x), N(x))$ and $(V(x), M(x))$ of problem (1) such that

$$\operatorname{rank}[W(x), N(x)] + \operatorname{rank}[V(x), M(x)] = 2\rho + 1$$

and

$$W(x)M(x) - V(x)N(x) = c\prod_{i=1}^{\rho}(x - x_i)$$

for some nonzero scalar $c \in \mathbb{F}_{2^m}$. We definitely have $\operatorname{rank}[W(x), N(x)] \neq \operatorname{rank}[V(x), M(x)]$, since $2\rho + 1$ is odd. Among the two complementary solutions, the one with lower rank is desired for decoding RS codes. It should be mentioned that the definition of rank presented here uses a different polynomial order from the original definition in [5] (more precisely, it uses $(W(x), N(x))$ instead of $(N(x), W(x))$), since it is convenient to have the same order as the basis matrix defined below. More detailed discussions can be found in [2], [5], and [21].

By characterizing the solution set of the WB key equation as an $\mathbb{F}_{2^m}[x]$-module, the so-called modular approach provides an efficient algorithm for constructing the desired solution. Before presenting the MA, we first review some essential concepts regarding modules and homomorphisms.

Definition 2: For the polynomial ring $\mathbb{F}_{2^m}[x]$, an $\mathbb{F}_{2^m}[x]$-module $\mathcal{Q}$ is an abelian group with a law of composition, written as $+$, together with a scalar multiplication $\mathbb{F}_{2^m}[x] \times \mathcal{Q} \to \mathcal{Q}$, written as $a, v \to av$, that satisfies the axioms

$$\begin{gathered} 1v = v, \quad (ab)v = a(bv), \\ (a+b)v = av + bv, \quad a(v + v') = av + av' \end{gathered} \tag{2}$$

for all $a, b \in \mathbb{F}_{2^m}[x]$, $v, v' \in \mathcal{Q}$ and such that the results of the operations in (2) are still in $\mathcal{Q}$.

Notice that these are precisely the axioms for a linear space except that the scalars come from a ring rather than a field. Thus, modules are natural generalizations of linear spaces to rings. Hence, the concepts of basis and independence can be carried over from linear spaces to modules. However, the number of elements of a basis for a module is called the rank, instead of the dimension.

Definition 3: An $\mathbb{F}_{2^m}[x]$-module $\mathcal{Q}$ is called a free $\mathbb{F}_{2^m}[x]$-module of rank 2 if there exist independent elements $v, v' \in \mathcal{Q}$ such that any $w \in \mathcal{Q}$ can be represented as a linear combination of v and v', i.e., $w = av + bv'$ for $a, b \in \mathbb{F}_{2^m}[x]$. The set $\{v, v'\}$ is called a basis of $\mathcal{Q}$.

Next, the concept of a module homomorphism is introduced so that the solution set to the WB key equation can be described as the kernel of a specific module homomorphism.

Definition 4: For two $\mathbb{F}_{2^m}[x]$-modules $\mathcal{Q}$ and $\mathcal{Q}'$, a module homomorphism $\varphi : \mathcal{Q} \to \mathcal{Q}'$ is a map that is compatible with the laws of composition: $\varphi(v + v') = \varphi(v) + \varphi(v')$ and $\varphi(av) = a\varphi(v)$ for all $v, v' \in \mathcal{Q}$ and $a \in \mathbb{F}_{2^m}[x]$. The kernel of φ, denoted by $\ker(\varphi)$, is the set of elements of $\mathcal{Q}$ that are mapped to the additive identity 0 in $\mathcal{Q}'$, i.e., $\ker(\varphi) = \{v \in \mathcal{Q} \mid \varphi(v) = 0\}$.

Detailed discussions of modules and homomorphisms can be found in a variety of modern algebra books; see, for example, [22].

We rewrite the WB key equation (1) in a more general form as

$$d_iW(x) + g_iN(x) \equiv 0 \pmod{x - x_i}, \quad i = 1, 2, \ldots, \rho, \tag{3}$$

by setting $d_i = y_i$ and $g_i = 1$. For each i, we define a module homomorphism $\phi_i : \mathbb{F}_{2^m}[x]^2 \to \mathbb{F}_{2^m}$ on the corresponding equation in (3) by

$$\begin{aligned} \phi_i(W(x), N(x)) &= d_iW(x) + g_iN(x) \bmod (x - x_i) \\ &= (W(x_i), N(x_i))\begin{pmatrix} d_i \\ g_i \end{pmatrix}. \end{aligned}$$

Note that $\mathbb{F}_{2^m}[x]^2$ represents the module of $\mathbb{F}_{2^m}[x]$-vectors, i.e., row vectors with entries in $\mathbb{F}_{2^m}[x]$. Clearly, $\mathbb{F}_{2^m}[x]^2$ is a free $\mathbb{F}_{2^m}[x]$-module of rank 2 with a basis $\{(1, 0), (0, 1)\}$. The kernel of ϕ_i characterizes the solution to the ith equation of (3), and the intersection $\ker(\phi_1) \cap \ker(\phi_2) \cap \cdots \cap \ker(\phi_\rho)$ is the solution to the set of congruences in (3). Next, as $x - x_i, i = 1, 2, \ldots, \rho$, are pairwise relatively prime, if we define the homomorphism $\phi : \mathbb{F}_{2^m}[x]^2 \to \mathbb{F}_{2^m}[x]$ by

$$\phi(W(x), N(x)) = D(x)W(x) + G(x)N(x) \bmod \Pi(x), \quad (4)$$

where $D(x_i) = d_i, G(x_i) = g_i$ and $\Pi(x) = \prod_{i=1}^{\rho}(x - x_i)$, then it follows that

$$\ker(\phi) = \ker(\phi_1) \cap \ker(\phi_2) \cap \cdots \cap \ker(\phi_\rho)$$

by the Chinese remainder theorem. Note that we can make the assumption that the greatest common divisor $\gcd(D(x), G(x))$ is relatively prime to $\Pi(x)$.

From the above discussion, the solution set to the WB key equation (1) is exactly the kernel of ϕ. Next, we demonstrate that $\ker(\phi)$ is a free $\mathbb{F}_{2^m}[x]$-module of rank 2 and that an irreducible basis matrix, which is desired for decoding RS codes, exists in $\ker(\phi)$. Before describing $\ker(\phi)$, we first develop the concept of a basis matrix. Hereinafter, **the notation $\mathcal{Q}$ represents a free $\mathbb{F}_{2^m}[x]$-module of rank 2 satisfying $\mathcal{Q} \subseteq \mathbb{F}_{2^m}[x]^2$.**

Definition 5: Ψ is called a basis matrix of $\mathcal{Q}$ if its rows form a basis of $\mathcal{Q}$.

Theorem 1 [23]: $\ker(\phi)$ is a free $\mathbb{F}_{2^m}[x]$-module of rank 2 and the followings hold:

1) For any basis matrix Ψ of $\ker(\phi)$, we have $\det(\Psi) = c\Pi(x)$ for some nonzero $c \in \mathbb{F}_{2^m}$.
2) Conversely, if the rows of $\Phi \in \mathbb{F}_{2^m}[x]^{2\times 2}$ are in $\ker(\phi)$ and $\det(\Phi) = c\Pi(x)$ for some nonzero $c \in \mathbb{F}_{2^m}$, then Φ is a basis matrix of $\ker(\phi)$.

Although $\ker(\phi)$ can be described by an arbitrary basis matrix, the one with the lowest complexity is desired for decoding RS codes. The complexity of a basis matrix is characterized by a property called irreducibility.

Definition 6: A basis matrix $\begin{pmatrix} W(x) & N(x) \\ V(x) & M(x) \end{pmatrix}$ of $\ker(\phi)$ is said to be irreducible if, for any basis matrix $\begin{pmatrix} W'(x) & N'(x) \\ V'(x) & M'(x) \end{pmatrix}$, we have

$$\begin{aligned}\operatorname{rank}[W(x), N(x)] &+ \operatorname{rank}[V(x), M(x)] \\ &\le \operatorname{rank}[W'(x), N'(x)] + \operatorname{rank}[V'(x), M'(x)].\end{aligned}$$

Lemma 1 [2]: A basis matrix $\begin{pmatrix} W(x) & N(x) \\ V(x) & M(x) \end{pmatrix}$ of $\ker(\phi)$ is irreducible if

$$\operatorname{rank}[W(x), N(x)] + \operatorname{rank}[V(x), M(x)] = 2\rho + 1.$$

According to Lemma 1, if we find a basis matrix which satisfies the rank constraint in Lemma 1, then its rows are solutions to the WB key equation with the lowest complexity. Hence, the WB key equation problem is converted to constructing a basis matrix of $\ker(\phi)$ which satisfies the rank constraint. We now turn to describe the MA, which is an efficient algorithm for finding such a matrix. The main idea behind the MA, roughly speaking, is to find a module chain step by step. For example, if we want to solve $\ker(\phi_1) \cap \ker(\phi_2)$, the MA first constructs an irreducible basis matrix of $\ker(\phi_1)$. Next, the homomorphism ϕ_2 is updated and restricted to $\ker(\phi_1)$ so that a desired solution can be found. As $\ker(\phi_1) \cap \ker(\phi_2)$ is a free $\mathbb{F}_{2^m}[x]$-module of rank 2, by Theorem 1, $\ker(\phi_1) \cap \ker(\phi_2) \subseteq \ker(\phi_1)$, which forms a module chain. Before describing the MA precisely, we introduce the concept of a projection of a module.

Definition 7: A homomorphism $\psi : \mathbb{F}_{2^m}[x]^2 \to \mathcal{Q}$ is called a projection of $\mathcal{Q}$ if and only if the image of $\mathbb{F}_{2^m}[x]^2$ under ψ is equal to $\mathcal{Q}$, i.e., $\operatorname{Im}(\psi) = \mathcal{Q}$.

Remarks: Let Ψ be any basis matrix of $\mathcal{Q}$. Then the homomorphism $\psi : \mathbb{F}_{2^m}[x]^2 \to \mathcal{Q}$ defined by $\psi(W(x), N(x)) = (W(x), N(x))\Psi$ is a projection of $\mathcal{Q}$.

The following lemma provides a powerful tool for finding the module chain.

Lemma 2: Let ψ be a projection of $\mathcal{Q}$ and let φ be a homomorphism that maps $\mathbb{F}_{2^m}[x]^2$ to $\mathbb{F}_{2^m}[x]^2$. Then $\mathcal{Q} \cap \ker(\varphi) = \psi(\ker(\varphi \circ \psi))$, where $\varphi \circ \psi$ denotes composition of maps, with ψ being applied first, followed by φ.

Proof: Note that $\psi(\ker(\varphi \circ \psi)) \subseteq \operatorname{Im}(\psi) = \mathcal{Q}$. Since φ is a module homomorphism, by definition, $\varphi(\psi(\ker(\varphi \circ \psi))) = 0$ such that $\psi(\ker(\varphi \circ \psi)) \subseteq \ker(\varphi)$. Then $\psi(\ker(\varphi \circ \psi)) \subseteq \mathcal{Q} \cap \ker(\varphi)$.

Conversely, $\varphi \circ \psi(\psi^{-1}(\mathcal{Q} \cap \ker(\varphi)))) = 0$, which implies $\psi^{-1}(\mathcal{Q} \cap \ker(\varphi)) \subseteq \ker(\varphi \circ \psi)$. It follows that $\mathcal{Q} \cap \ker(\varphi) \subseteq \psi(\ker(\varphi \circ \psi))$. Hence, one must have $\mathcal{Q} \cap \ker(\varphi) = \psi(\ker(\varphi \circ \psi))$. $\square$

We are now ready to describe the MA. For $i \in \{1, 2, \ldots, \rho\}$ and $j \in \{0, 1, \ldots, \rho\}$, recursively define the homomorphisms $\phi_i^j : \mathbb{F}_{2^m}[x]^2 \to \mathbb{F}_{2^m}$ by

$$\phi_i^j(W(x), N(x)) = \phi_i((W(x), N(x))\Psi_1^j),$$

where

$$\Psi_1^j = \begin{cases} \text{identity matrix } I_{2\times 2}, & j = 0, \\ \Psi_j \Psi_{j-1} \cdots \Psi_1, & j > 0, \end{cases}$$

and Ψ_j is a basis matrix of $\ker(\phi_j^{j-1})$ for $j > 0$. For any matrix $\Psi \in \mathbb{F}_{2^m}[x]^{2\times 2}$, $\Psi(x_i)$ denotes the matrix whose terms are the evaluations of the corresponding terms of Ψ at x_i.

Lemma 3: ϕ_i^j, Ψ_1^j, and Ψ_j are well-defined.

Proof: Because Ψ_1^0 is an identity matrix, we have $\phi_1^0 = \phi_1$, which implies that Ψ_1 exists according to Theorem 1. Now suppose that $\Psi_1, \Psi_2, \ldots, \Psi_{j-1}$ exist. Then Ψ_1^{j-1} must exist, which means that

$$\begin{aligned}\phi_j^{j-1}(W(x), N(x)) &= \phi_j((W(x), N(x))\Psi_1^{j-1}) \\ &= (W(x_j), N(x_j))\Psi_1^{j-1}(x_j)\begin{pmatrix} d_j \\ g_j \end{pmatrix}\end{aligned}$$

Since ϕ_j^{j-1} is a special case of ϕ, by a similar proof to that of Theorem 1, we have that $\ker(\phi_j^{j-1})$ is a free $\mathbb{F}_{2^m}[x]$-module of rank 2. Hence, Ψ_j exists by Theorem 1.

Since Ψ_j exists for $j = 1, 2, \ldots, \rho$, Ψ_1^j is well-defined, which also implies that ϕ_i^j is well-defined for all $i = 1, 2, \ldots, \rho$ and $j = 0, 1, \ldots, \rho$. $\square$

For $i \in \{1, 2, \ldots, \rho\}$ and $j \in \{0, 1, \ldots, \rho\}$, we rewrite the homomorphism ϕ_i^j as

$$\begin{aligned}\phi_i^j(W(x), N(x)) &= (W(x_i), N(x_i))\Psi_1^j(x_i)\begin{pmatrix} d_i \\ g_i \end{pmatrix} \\ &= (W(x_i), N(x_i))\begin{pmatrix} d_i^j \\ g_i^j \end{pmatrix}\end{aligned}$$

and define the homomorphisms $\psi_j, \psi_1^j : \mathbb{F}_{2^m}[x]^2 \to \mathbb{F}_{2^m}[x]^2$ by

$$\psi_j(W(x), N(x)) = (W(x), N(x))\Psi_j$$

and

$$\psi_1^j(W(x), N(x)) = (W(x), N(x))\Psi_1^j.$$

It is easy to see that $\psi_1^j = \psi_1^{j-1} \circ \psi_j$ and $\phi_i^j = \phi_i \circ \psi_1^j$.

Lemma 4: ϕ_j^{j-1} is nontrivial[1] for $j = 1, 2, \ldots, \rho$.

Proof: The proof is by induction on j. As $\phi_1^0 = \phi_1$, the claim is true for $j = 1$ by the assumption that ϕ_1 is nontrivial. Next, suppose that for $l = 1, 2, \ldots, j-1$, ϕ_l^{l-1} are nontrivial. It follows that $\det(\Psi_l) = c_l(x - x_l)$ and that $\det(\Psi_1^{j-1}) = \prod_{l=1}^{j-1} c_l(x - x_l)$ for nonzero scalars c_l. If ϕ_j^{j-1} is trivial, one must have $\det(\Psi_1^{j-1}(x_j)) = 0$, which is impossible since $x_1, \ldots, x_j$ are distinct. Hence, we can conclude that ϕ_j^{j-1} is nontrivial for $j = 1, 2, \ldots, \rho$. □

As ϕ_j^{j-1} is nontrivial, either d_j^{j-1} or g_j^{j-1} is nonzero. Next, we show that, by suitably choosing Ψ_j, Ψ_1^j is an irreducible basis matrix of $\ker(\phi_1) \cap \cdots \cap \ker(\phi_j)$. Define the map $R : \mathbb{F}_{2^m}[x]^{2\times 2} \to \{0, 1\}$ as

$$R(\Psi_1^j) = \begin{cases} 1 & \text{if the first row of } \Psi_1^j \text{ has a} \\ & \text{larger rank than its second row,} \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 5: Let the basis matrix Ψ_j be equal to

$$\begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ x - x_j & 0 \end{pmatrix} \tag{5}$$

if $g_j^{j-1} = 0$ or ($d_j^{j-1} \neq 0$ and $R(\Psi_1^{j-1}) = 0$), and to

$$\begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ 0 & x - x_j \end{pmatrix} \text{otherwise.} \tag{6}$$

Then $\Psi_1^j = \begin{pmatrix} W(x) & N(x) \\ V(x) & M(x) \end{pmatrix}$ satisfies

$$\mathrm{rank}[W(x), N(x)] + \mathrm{rank}[V(x), M(x)] = 2j + 1,$$

and Ψ_1^j is an irreducible basis matrix of $\ker(\phi_1) \cap \ker(\phi_2) \cap \cdots \cap \ker(\phi_j)$.

Proof: The proof is by induction on j. For $j = 1$, we have $\phi_1^0 = \phi_1$. Since $\Psi_1^0 = I$, $R(\Psi_1^0) = 0$. When $g_1^0 = 0$ or ($d_1^0 \neq 0$ and $R(\Psi_1^0) = 0$), Ψ_1 is equal to the matrix in (5). It is straightforward to see that $(-g_1^0, d_1^0)$ and $(x - x_1, 0)$ are included in $\ker(\phi_1)$ by applying ϕ_1 to them. Thus, Ψ_1 is a basis matrix of $\ker(\phi_1)$ by Theorem 1, since $\det(\Psi_1) = d_1^0(x - x_1)$. Next, we have $\mathrm{rank}[-g_1^0, d_1^0] + \mathrm{rank}[x - x_1, 0] = 1 + 2 = 3$. Therefore, by Lemma 1, $\Psi_1^1 = \Psi_1$ is an irreducible basis matrix of $\ker(\phi_1)$.

On the other hand, it is straightforward to verify the claims for the case in which Ψ_1 is equal to the matrix in (6) by repeating the above proof.

We have proved the claims for $j = 1$. Next, suppose that the claims are true for $1, 2, \ldots, j-1$.

For all cases, it is easy to verify that Ψ_j is a basis matrix of $\ker(\phi_j^{j-1})$ by repeating the above proof. By induction, Ψ_1^{j-1} is an irreducible basis matrix of $\ker(\phi_1) \cap \cdots \cap \ker(\phi_{j-1})$.

[1]A trivial homomorphism maps all elements to the additive identity.

So, ψ_1^{j-1} is a projection of $\ker(\phi_1) \cap \cdots \cap \ker(\phi_{j-1})$. According to Lemma 2, we have

$$\begin{aligned} \ker(\phi_1) \cap \cdots \cap \ker(\phi_{j-1}) \cap \ker(\phi_j) &= \psi_1^{j-1}(\ker(\phi_j \circ \psi_1^{j-1})) \\ &= \psi_1^{j-1}(\ker(\phi_j^{j-1})) \\ &= \psi_1^j(\mathbb{F}_{2^m}[x]^2), \end{aligned}$$

where the last equality follows because ψ_j is a projection of $\ker(\phi_j^{j-1})$. Then the polynomial pairs $\psi_1^j(1, 0)$ and $\psi_1^j(0, 1)$, which are exactly the two rows of Ψ_1^j, are contained in $\ker(\phi_1) \cap \cdots \cap \ker(\phi_j)$. Furthermore, as

$$\det(\Psi_1^j) = \det(\Psi_j)\det(\Psi_1^{j-1}) = c\prod_{l=1}^{j}(x - x_l)$$

for some nonzero $c \in \mathbb{F}_{2^m}$, Ψ_1^j is a basis matrix of $\ker(\phi_1) \cap \cdots \cap \ker(\phi_j)$ by Theorem 1.

It remains to show that Ψ_1^j is irreducible. We write Ψ_1^{j-1} as $\begin{pmatrix} W'(x) & N'(x) \\ V'(x) & M'(x) \end{pmatrix}$. Recall that $\mathrm{rank}[W'(x), N'(x)] + \mathrm{rank}[V'(x), M'(x)] = 2(j-1)$.

When $g_j^{j-1} = 0$ or ($d_j^{j-1} \neq 0$ and $R(\Psi_1^{j-1}) = 0$),

$$\begin{aligned} \Psi_1^j &= \begin{pmatrix} W(x) & N(x) \\ V(x) & M(x) \end{pmatrix} \\ &= \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ x - x_j & 0 \end{pmatrix}\begin{pmatrix} W'(x) & N'(x) \\ V'(x) & M'(x) \end{pmatrix}. \end{aligned}$$

If $g_j^{j-1} = 0$ or $R(\Psi_1^{j-1}) = 0$, then it is easy to verify that

$$\begin{aligned} &\mathrm{rank}[W(x), N(x)] \\ &\quad = \mathrm{rank}[-g_j^{j-1}W'(x) + d_j^{j-1}V'(x), \\ &\qquad -g_j^{j-1}N'(x) + d_j^{j-1}M'(x)] \\ &\quad = \mathrm{rank}[V'(x), M'(x)]. \end{aligned}$$

As

$$\begin{aligned} \mathrm{rank}[V(x), M(x)] &= \mathrm{rank}[(x - x_j)W'(x), (x - x_j)N'(x)] \\ &= \mathrm{rank}[W'(x), N'(x)] + 2, \end{aligned}$$

it follows that $\mathrm{rank}[W(x), N(x)] + \mathrm{rank}[V(x), M(x)] = 2(j-1) + 2 = 2j + 1$. Then Ψ_1^j is an irreducible basis matrix of $\ker(\phi_1) \cap \cdots \cap \ker(\phi_j)$ by Lemma 1.

On the other hand, if

$$\begin{aligned} \Psi_1^j &= \begin{pmatrix} W(x) & N(x) \\ V(x) & M(x) \end{pmatrix} \\ &= \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ 0 & x - x_j \end{pmatrix}\begin{pmatrix} W'(x) & N'(x) \\ V'(x) & M'(x) \end{pmatrix}, \end{aligned}$$

then, by preceding as before, one can verify that $\mathrm{rank}[W(x), N(x)] + \mathrm{rank}[V(x), M(x)] = 2j + 1$. By Lemma 1, Ψ_1^j is irreducible in $\ker(\phi_1) \cap \cdots \cap \ker(\phi_j)$. This completes the proof. □

A detailed description of the MA is presented in Algorithm 1, in which r_1^j and r_2^j denote the ranks of the two rows of Ψ_1^j. Since $\Psi_1^0 = I_{2\times 2}$, we have $r_1^0 = 0, r_2^0 = 1$ at the beginning. At the jth iteration, the MA first identifies the basis

Algorithm 1 Modular Approach

Input: $\{x_i, d_i, g_i\}, i = 1, 2, \ldots, \rho$.

Output: An irreducible basis matrix Ψ_1^ρ of $\ker(\phi_1) \cap \ker(\phi_2) \cap \cdots \cap \ker(\phi_\rho)$, r_1^ρ, r_2^ρ, where $\phi_i, i = 1, 2, \ldots, \rho$ are homomorphisms defined by $\phi_i(W(x), N(x)) = d_i W(x_i) + g_i N(x_i)$.

Initialization: $d_i^0 = d_i, g_i^0 = g_i$ for $i = 1, 2, \ldots, \rho$, $\Psi_1^0 = I_{2\times 2}$, $r_1^0 = 0, r_2^0 = 1$.

for $j = 1, 2, \ldots, \rho$ **do**

  if $g_j^{j-1} = 0$ or ($d_j^{j-1} \neq 0$ and $r_1^{j-1} < r_2^{j-1}$) **then**

    Let

$$\Psi_j = \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ x - x_j & 0 \end{pmatrix}$$

    and

$$\Psi_j(x_i) = \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ x_i - x_j & 0 \end{pmatrix} \text{ for } i = 1, 2, \ldots, \rho$$

    $r_1^j = r_2^{j-1}, r_2^j = r_1^{j-1} + 2$.

  else

    Let

$$\Psi_j = \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ 0 & x - x_j \end{pmatrix}$$

    and

$$\Psi_j(x_i) = \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ 0 & x_i - x_j \end{pmatrix} \text{ for } i = 1, 2, \ldots, \rho$$

    $r_1^j = r_1^{j-1}, r_2^j = r_2^{j-1} + 2$.

  end if

  for $i = j + 1, j + 2, \ldots, \rho$ **do**

$$\begin{pmatrix} d_i^j \\ g_i^j \end{pmatrix} = \Psi_j(x_i) \begin{pmatrix} d_i^{j-1} \\ g_i^{j-1} \end{pmatrix}$$

  end for

  $\Psi_1^j = \Psi_j \Psi_1^{j-1}$.

end for

return $\Psi_1^\rho, r_1^\rho, r_2^\rho$.

matrix Ψ_j of $\ker(\phi_j^{j-1})$ by checking the conditions $g_j^{j-1} \neq 0$, $d_j^{j-1} \neq 0$, and $r_1^{j-1} < r_2^{j-1}$. Then $\Psi_j(x_i)$ can be obtained by simply substituting x_i into Ψ_j. Next, d_i^j, g_i^j, and the basis matrix Ψ_1^j are updated in parallel. Finally, the MA returns an irreducible basis matrix Ψ_1^t for decoding RS codes.

The original MA was first proposed in [23]. However, there exist many differences between the original approach and the new one presented here. First, we define an irreducible basis matrix here and prove its existence for the desired kernel. Second, a more efficient algorithm is proposed for finding such an irreducible basis matrix. The original method in [23] needs to find a homomorphism with nonzero d_j^{j-1} during each iteration, which significantly limits its speed, especially in hardware implementation. However, our new method here eliminates the need for such a procedure, thereby enabling the development of a high-speed architecture, which is a prerequisite for real applications. Finally, the new method tracks the ranks during each iteration, which is essential for identifying uncorrectable errors.

It is well known that both the WB algorithm and the Euclidean algorithm are capable of solving the WB key equation (10). However, the WB algorithm executes the polynomial evaluations and the polynomial updates in sequence during each iteration. This means that the operations in each iteration of the WB algorithm cannot be done in parallel. Therefore, its hardware implementation has a long critical path. More details about the WB algorithm can be found in [5]. On the other hand, the Euclidean algorithm fails to provide an efficient method for decoding RS codes based on the FFT, and for that reason we do not discuss it here.

There are several algorithms which find specific elements of a module. Fitzpatrick [24] presented a method for finding a low-weight element of the solution to the key equation $z(x) \equiv \lambda(x) \bmod x^{2t}$. Algorithms for solving the rational interpolation problem in the GS algorithm were proposed in [14], [15], [16], [17], and [18]. Compared with these algorithms, an advantage of the proposed method is that the operations in each iteration can be done in parallel, which is an important feature in implementation. Furthermore, the coming section proves that the fast modular approach is superior in terms of complexity.

III. Decoding Reed–Solomon Codes Based on FFT

In this section, a new algorithm is presented for decoding RS codes based on the FFT, which takes the MA as the key equation solver. Two versions of the MA are presented. The first, the frequency-domain modular approach (FDMA), is suitable for decoding short RS codes. The second, the fast modular approach (FMA), is suitable for decoding medium or long RS codes. We shall see that the new decoding algorithm has the smallest constant factor achieved to date, while also reaching the best known asymptotic computational complexity.

A. FFT Algorithm

Let $(v_0, v_1, \ldots, v_{m-1})$ be a basis of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$. The elements in $\mathbb{F}_{2^m}$ can be represented by

$$\omega_l = l_0 v_0 + l_1 v_1 + \cdots + l_{m-1} v_{m-1}, \quad 0 \leq l < 2^m,$$

where $l_0, \ldots, l_{m-1} \in \{0, 1\}$ is the binary representation of l. The subspace polynomial is defined as $s_\tau(x) = \prod_{l=0}^{2^\tau - 1}(x - \omega_l)$ for $\tau = 0, 1, \ldots, m$. Obviously, we have $\deg(s_\tau(x)) = 2^\tau$. Then the polynomial given by

$$\bar{X}_l(x) = \frac{s_0(x)^{l_0} s_1(x)^{l_1} \cdots s_{m-1}(x)^{l_{m-1}}}{s_0(v_0)^{l_0} s_1(v_1)^{l_1} \cdots s_{m-1}(v_{m-1})^{l_{m-1}}}$$

has degree l for $l = 0, 1, \ldots, 2^m - 1$. Therefore, the set $\bar{\mathbb{X}} = \{\bar{X}_0(x), \bar{X}_1(x), \ldots, \bar{X}_{2^m-1}(x)\}$ is a basis of the linear space $\mathbb{F}_{2^m}[x]/(x^{2^m} - x)$, which implies that any polynomial $f(x)$ in this space can be represented as a linear combination: $f(x) = \sum_{l=0}^{2^m-1} \bar{f}_l \bar{X}_l(x)$. The vector $\bar{\mathbf{f}} = (\bar{f}_0, \bar{f}_1, \ldots, \bar{f}_{2^m-1})$ is the coordinate vector of $f(x)$ with respect to the basis $\bar{\mathbb{X}}$.

Given that $\deg(f(x)) < 2^\tau$, the fast Fourier transform (FFT), denoted by $\mathrm{FFT}_{\bar{\mathbb{X}}}$, evaluates $f(x)$ at points $\{\omega_l + \beta \mid l = 0, 1, \ldots, 2^\tau - 1\}$:

$$\begin{aligned} \mathrm{FFT}_{\bar{\mathbb{X}}}(\bar{\mathbf{f}}, \tau, \beta) &= \mathbf{F} \\ &= (f(\omega_0 + \beta), f(\omega_1 + \beta), \ldots, f(\omega_{2^\tau - 1} + \beta)), \end{aligned}$$

for any $\beta \in \mathbb{F}_{2^m}$ and $\tau = 0, 1, \ldots, m$, which involves $\tau 2^\tau/2$ field multiplications and $\tau 2^\tau$ field additions [25]. The inverse FFT, denoted by $\mathrm{IFFT}_{\bar{\mathbb{X}}}$, calculates $\bar{\mathbf{f}}$ given $\mathbf{F}$, which also involves $\tau 2^\tau/2$ field multiplications and $\tau 2^\tau$ field additions in a direct implementation. Algorithms 2 and 3 present the details of $\mathrm{FFT}_{\bar{\mathbb{X}}}$ and $\mathrm{IFFT}_{\bar{\mathbb{X}}}$, respectively.

Algorithm 2 $\mathrm{FFT}_{\bar{\mathbb{X}}}$ [25]

Input: $\bar{\mathbf{f}} = (\bar{f}_0, \bar{f}_1, \ldots, \bar{f}_{2^\tau-1})$, τ, β.
Output: $(f(\omega_0+\beta), f(\omega_1+\beta), \ldots, f(\omega_{2^\tau-1}+\beta))$.
if $\tau = 0$ **then**
  return $\bar{f}_0$
end if
for $l = 0, 1, \ldots, 2^{\tau-1}-1$ **do**
  $a_l^{(0)} = \bar{f}_l + \dfrac{s_{\tau-1}(\beta)}{s_{\tau-1}(v_{\tau-1})}\bar{f}_{l+2^{\tau-1}}$
  $a_l^{(1)} = a_l^{(0)} + \bar{f}_{l+2^{\tau-1}}$
end for
$\mathbf{a}^{(0)} = (a_0^{(0)}, \ldots, a_{2^{\tau-1}-1}^{(0)}), \mathbf{a}^{(1)} = (a_0^{(1)}, \ldots, a_{2^{\tau-1}-1}^{(1)})$
Calculate $\mathbf{A}_0 = \mathrm{FFT}_{\bar{\mathbb{X}}}(\mathbf{a}^{(0)}, \tau-1, \beta)$, $\mathbf{A}_1 = \mathrm{FFT}_{\bar{\mathbb{X}}}(\mathbf{a}^{(1)}, \tau-1, v_{\tau-1}+\beta)$
return $(\mathbf{A}_0, \mathbf{A}_1)$

Algorithm 3 Inverse Transform of the Basis $\bar{\mathbb{X}}$ [25]

Input: $\mathbf{F} = (f(\omega_0+\beta), f(\omega_1+\beta), \ldots, f(\omega_{2^\tau-1}+\beta)), \tau, \beta$
Output: $\bar{\mathbf{f}}$ such that $\mathbf{F} = \mathrm{FFT}_{\bar{\mathbb{X}}}(\bar{\mathbf{f}}, \tau, \beta)$
if $\tau = 0$ **then**
  return $f(\omega_0+\beta)$
end if
$\mathbf{A}_0 = (f(\omega_0+\beta), \ldots, f(\omega_{2^{\tau-1}-1}+\beta)), \mathbf{A}_1 = (f(\omega_{2^{\tau-1}}+\beta)), \ldots, f(\omega_{2^\tau-1}+\beta))$
$\mathbf{a}^{(0)} = \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{A}_0, \tau-1, \beta)$, $\mathbf{a}^{(1)} = \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{A}_1, \tau-1, v_{\tau-1}+\beta)$
for $l = 0, 1, \ldots, 2^{\tau-1}-1$ **do**
  $\bar{f}_{l+2^{\tau-1}} = a_l^{(0)} + a_l^{(1)}$
  $\bar{f}_l = a_l^{(0)} + \dfrac{s_{\tau-1}(\beta)}{s_{\tau-1}(v_{\tau-1})}\bar{f}_{l+2^{\tau-1}}$
end for
return $\bar{\mathbf{f}}$

For $\mu \in \{0, 1, \ldots, \tau\}$, if we let

$$\mathbf{F} = (\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_{2^{\tau-\mu}}) \text{ and } \bar{\mathbf{f}} = (\bar{\mathbf{f}}_1, \bar{\mathbf{f}}_2, \ldots, \bar{\mathbf{f}}_{2^{\tau-\mu}}),$$

where

$$\begin{aligned}\mathbf{F}_i &= (f(\omega_{(i-1)2^\mu}+\beta), f(\omega_{(i-1)2^\mu+1}+\beta), \ldots, \\ &\qquad f(\omega_{i2^\mu-1}+\beta)), \\ \bar{\mathbf{f}}_i &= (\bar{f}_{(i-1)2^\mu}, \bar{f}_{(i-1)2^\mu+1}, \ldots, \bar{f}_{i2^\mu-1}).\end{aligned}$$

then [12, Lemma 10] and [26, Lemma 1]

$$\begin{aligned}&\mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}_1, \mu, \beta) + \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}_2, \mu, \omega_{2^\mu}+\beta) + \cdots \\ &\qquad + \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}_{2^{\tau-\mu}}, \mu, \omega_{2^\tau-2^\mu}+\beta) = \bar{\mathbf{f}}_{2^{\tau-\mu}}.\end{aligned}$$

As we shall see later, this important property is crucial for encoding and decoding RS codes.

B. Encoding Reed–Solomon Codes

For an (n, k) RS code where $n = 2^m$, $k = 2^m - 2^\mu$ and $\mu \in \{0, 1, \ldots, m-1\}$, the codewords are given by $\mathrm{FFT}_{\bar{\mathbb{X}}}(\bar{\mathbf{f}}, m, 0) = \mathbf{F} = (f(\omega_0), f(\omega_1), \ldots, f(\omega_{2^m-1}))$ for all polynomials $f(x)$ of degree less than $2^m - 2^\mu$. It follows that

$$\begin{aligned}&\mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}_1, \mu, 0) + \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}_2, \mu, \omega_{2^\mu}) + \cdots \\ &\qquad + \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}_{2^{m-\mu}}, \mu, \omega_{2^m-2^\mu}) = \bar{\mathbf{f}}_{2^{m-\mu}} = \mathbf{0}.\end{aligned}$$

If we let $\mathbf{F}_1$ be the check locations and $\mathbf{F}_2, \mathbf{F}_3, \ldots, \mathbf{F}_{2^{m-\mu}}$ be the message locations, then the encoding process is

$$\begin{aligned}&\mathrm{FFT}_{\bar{\mathbb{X}}}(\mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}_2, \mu, \omega_{2^\mu}) + \cdots + \\ &\qquad \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}_{2^{m-\mu}}, \mu, \omega_{2^m-2^\mu}), \mu, 0).\end{aligned}$$

The computational complexity of the encoding algorithm is $O(n\log(n-k))$.

C. Decoding Reed–Solomon Codes

The received vector can be represented by

$$\begin{aligned}\mathbf{r} &= \mathbf{F} + \mathbf{e} \\ &= (f(\omega_0), f(\omega_1), \ldots, f(\omega_{2^m-1})) + (e_0, e_1, \ldots, e_{2^m-1}),\end{aligned}$$

where $\mathbf{e}$ is the error pattern. If we write $E = \{\omega_l \mid e_l \neq 0$ for $0 \leq l \leq 2^m-1\}$, then the error locator polynomial can be defined as $\lambda(x) = \prod_{a\in E}(x-a)$. Note that there exists a polynomial $r(x) \in \mathbb{F}_{2^m}[x]$ with degree less than 2^m satisfying $r(\omega_l) = f(\omega_l) + e_l$ for $l = 0, 1, \ldots, 2^m-1$, which implies that $f(\omega_l)\lambda(\omega_l) = r(\omega_l)\lambda(\omega_l)$. Thus, the congruence $f(x)\lambda(x) \equiv r(x)\lambda(x) \pmod{s_m(x)}$ holds. Therefore, there exists $q(x) \in \mathbb{F}_{2^m}[x]$ such that

$$f(x)\lambda(x) = r(x)\lambda(x) + q(x)s_m(x). \tag{7}$$

Clearly, we have $\deg(f(x)) < 2^m - 2^\mu$, $\deg(\lambda(x)) \leq 2^{\mu-1}$, $\deg(r(x)) < 2^m$ and $\deg(s_m(x)) = 2^m$. Then the equation (7) implies that $\deg(q(x)) < \deg(\lambda(x))$. Dividing $s_m(x)$, $f(x)\lambda(x)$, and $r(x)$ by $p_{2^m-2^\mu}\bar{X}_{2^m-2^\mu}(x)$, where

$$p_{2^m-2^\mu} = s_0(v_0)^{l_0}s_1(v_1)^{l_1}\cdots s_{m-1}(v_{m-1})^{l_{m-1}}$$

and $(l_0, l_1, \ldots, l_{m-1})$ is the binary representation of $2^m - 2^\mu$, it follows that

$$\begin{aligned}s_m(x) &= p_{2^m-2^\mu}\bar{X}_{2^m-2^\mu}(x)(s_\mu(x) + s_\mu(v_\mu)) + \eta_s(x), \\ f(x)\lambda(x) &= p_{2^m-2^\mu}\bar{X}_{2^m-2^\mu}(x)z'(x) + \eta_f(x), \\ r(x) &= p_{2^m-2^\mu}\bar{X}_{2^m-2^\mu}(x)u(x) + \eta_r(x),\end{aligned}$$

where $\deg(\eta_s(x)), \deg(\eta_f(x)), \deg(\eta_r(x))$ are less than $\deg(p_{2^m-2^\mu}\bar{X}_{2^m-2^\mu}(x))$. When we divide both sides of (7) by $p_{2^m-2^\mu}\bar{X}_{2^m-2^\mu}(x)$ and keep the quotients, it becomes

$$z'(x) = u(x)\lambda(x) + q(x)s_\mu(x) + s_\mu(v_\mu)q(x) + \theta(x),$$

where $\theta(x)$ is the quotient of $q(x)\eta_s(x) + \lambda(x)\eta_r(x)$ and $\deg(\theta(x)) < \deg(\lambda(x))$. As $\deg(f(x)\lambda(x)) < 2^m - 2^\mu + \deg(\lambda(x))$, one can conclude that $\deg(z'(x)) < \deg(\lambda(x))$.

Let $z(x) = z'(x) - s_\mu(x)q(x) - \theta(x)$. We can then obtain the key equation:

$$z(x) = u(x)\lambda(x) + q(x)s_\mu(x), \tag{8}$$

where $\deg(z(x)) < \deg(\lambda(x))$. Note that if the received vector $\mathbf{r}$ is a codeword, the degree of $r(x)$ is less than $2^m - 2^\mu$, which implies that $u(x) = 0$. Hence, $u(x)$ can be treated as the syndrome polynomial. Given $\mathbf{r}$, the coordinate vector of $u(x)$ with respect to $\bar{\mathbb{X}}$ can be computed by

$$\sum_{i=0}^{2^{m-\mu}-1} \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{r}_i, \mu, \omega_{i\cdot 2^\mu})/p_{2^m-2^\mu}, \tag{9}$$

where $\mathbf{r}_i = (r_{i\cdot 2^\mu}, r_{i\cdot 2^\mu+1}, \ldots, r_{i\cdot 2^\mu+2^\mu-1})$ is the sub-vector of $\mathbf{r}$. Details are given in [12] and [26].

The key equation (8) can be rewritten as

$$z(x) = u(x)\lambda(x) \bmod \prod_{i=0}^{2^\mu-1}(x - \omega_i), \tag{10}$$

where $\deg(z(x)) < \deg(\lambda(x))$. This is in the WB form and hence can be solved by the MA.

Once the error locator polynomial $\lambda(x)$ has been obtained, its roots can be calculated by the FFT algorithm:

$$\mathrm{FFT}_{\bar{\mathbb{X}}}(\bar{\lambda}, \mu, \omega_{l\cdot 2^\mu}), \quad l = 0, 1, \ldots, 2^{m-\mu} - 1, \tag{11}$$

where $\bar{\lambda}$ is the coordinate vector of $\lambda(x)$ with respect to $\bar{\mathbb{X}}$.

It remains to compute the error values. The formal derivative of (7) is

$$\begin{aligned} &f'(x)\lambda(x) + f(x)\lambda'(x) \\ &\quad = r'(x)\lambda(x) + r(x)\lambda'(x) \ + q'(x)s_m(x) + q(x). \end{aligned}$$

For an error locator $\omega_l \in E$, we have $f(\omega_l)\lambda'(\omega_l) = r(\omega_l)\lambda'(\omega_l) + q(\omega_l)$. It follows that $f(\omega_l) - r(\omega_l) = q(\omega_l)/\lambda'(\omega_l)$. If ω_l is a message location, then, by (8), we have $q(\omega_l) = z(\omega_l)/s_\mu(\omega_l)$. Hence, Forney's formula for solving the error value is

$$f(\omega_l) - r(\omega_l) = \frac{z(\omega_l)}{s_\mu(\omega_l)\lambda'(\omega_l)}. \tag{12}$$

Note that there is no need to correct the errors in check locations.

Detecting uncorrectable errors is crucial in real applications. In the above decoding algorithm, a correctable error occurs if and only if

$$\deg(\lambda(x)) \le 2^{\mu-1}, \tag{13}$$

$$\deg(z(x)) < \deg(\lambda(x)), \tag{14}$$

$$|\{\omega_l \mid \lambda(\omega_l) = 0, l = 0, 1, \ldots, 2^m - 1\}| = \deg(\lambda(x)). \tag{15}$$

Note that the MA always ensures that $\deg(\lambda(x)) \le 2^{\mu-1}$, since $\mathrm{rank}[\lambda(x), z(x)] \le 2^\mu$. In addition, if $\deg(z(x)) \ge \deg(\lambda(x))$, then $\mathrm{rank}[\lambda(x), z(x)]$ must be odd. Hence, tracking the ranks is enough to check the condition (14). Finally, the condition (15) can be checked by the FFT algorithm (11). As a result, all of the uncorrectable errors can be detected.

The computational complexities of computing the syndrome, finding roots of $\lambda(x)$, and Forney's formula are $O(n\log(n-k))$. More detailed discussions can be found in [12] and [26].

D. Frequency-Domain Modular Approach

We now turn to solving the key equation (10) using the MA. Clearly, if we set $x_i = \omega_{i-1}$, $d_i = u(\omega_{i-1})$, and $g_i = 1$ for $i = 1, 2, \ldots, 2^\mu$, then Algorithm 1 provides two polynomial pairs satisfying the key equation, and the one with lower rank is exactly the desired solution. However, the FFT algorithm given in (11) requires that the polynomials to be evaluated be represented with respect to $\bar{\mathbb{X}}$, and therefore basis transformations are needed if the polynomials obtained are represented with respect to the monomial basis. To avoid the need for these basis transformations, we devise the FDMA. The FDMA updates $\Psi_1^j(\omega_i)$ instead of Ψ_1^j. Note that as $\deg(z(x)) \le 2^{\mu-1}$ and $\deg(\lambda(x)) \le 2^{\mu-1}$, $2^{\mu-1}+1$ points in the frequency domain are enough for determining $\lambda(x)$ or $z(x)$, which implies that we need to update only $\Psi_1^j(\omega_i), i = 0, 1, \ldots, 2^{\mu-1}$. Furthermore, because $(\lambda(x), z(x))$ satisfies the key equation (10), we must have $z(\omega_i) = u(\omega_i)\lambda(\omega_i)$ for $i = 0, 1, \ldots, 2^{\mu-1}$. Thus, the evaluations of $z(x)$ can be performed immediately once $\lambda(\omega_i)$ are available. To sum up, the FDMA computes only the first column of $\Psi_1^j(\omega_i)$ during the iterations and then identifies $\lambda(\omega_i)$ by the rank. Next, it computes $z(\omega_i)$ once the iterations have been done. Finally, extended $\mathrm{IFFT}_{\bar{\mathbb{X}}}$ algorithms are used to obtain the coordinate vectors of $\lambda(x), z(x)$ with respect to $\bar{\mathbb{X}}$. Algorithm 4 shows the details of the FDMA. Note that we set $x_i = \omega_{i-1}$ here. Compared with Algorithm 1, the FDMA computes only two polynomials in the frequency domain, rather than four. This further reduces the computational complexity and makes the FDMA suitable for hardware implementation.

Lemma 6: Given $f(\omega_i + \beta), i = 0, 1, \ldots, 2^\mu, \mu$, and any $\beta \in \mathbb{F}_{2^m}$, Algorithm 5 outputs the corresponding $f(x)$, and its complexity is $O(\mu 2^\mu)$.

Proof: Since $\hat{f}(x)$ is obtained by calling Algorithm 3, it follows that $\hat{f}(\omega_i + \beta) = f(\omega_i + \beta)$ for $i = 0, 1, \ldots, 2^\mu - 1$ and that $\deg(\hat{f}(x)) < 2^\mu$. Because $\bar{X}_{2^\mu}(x) = s_\mu(x)/s_\mu(v_\mu)$, we have $\bar{X}_{2^\mu}(\omega_i + \beta) = s_\mu(\omega_i)/s_\mu(v_\mu) + s_\mu(\beta)/s_\mu(v_\mu)$. Recall that $s_\mu(x) = \prod_{l=0}^{2^\mu-1}(x - \omega_l)$. So, for $i = 0, 1, \ldots, 2^\mu - 1$, we have $\bar{X}_{2^\mu}(\omega_i + \beta) = s_\mu(\beta)/s_\mu(v_\mu)$. Hence, for $i = 0, 1, \ldots, 2^\mu - 1$,

$$\begin{aligned} &(f(\omega_{2^\mu} + \beta) - \hat{f}(\omega_{2^\mu} + \beta))\left(\bar{X}_{2^\mu}(\omega_i + \beta) - \frac{s_\mu(\beta)}{s_\mu(v_\mu)}\right) \\ &\qquad + \hat{f}(\omega_i + \beta) \\ &\quad = f(\omega_i + \beta). \end{aligned}$$

Furthermore, if $i = \omega_{2^\mu}$, we have

$$\begin{aligned} &(f(\omega_{2^\mu} + \beta) - \hat{f}(\omega_{2^\mu} + \beta))\left(\bar{X}_{2^\mu}(\omega_{2^\mu} + \beta) - \frac{s_\mu(\beta)}{s_\mu(v_\mu)}\right) \\ &\quad + \hat{f}(\omega_{2^\mu} + \beta) \\ &= (f(\omega_{2^\mu} + \beta) - \hat{f}(\omega_{2^\mu} + \beta))s_\mu(v_\mu)/s_\mu(v_\mu) + \hat{f}(\omega_{2^\mu} + \beta) \\ &= f(\omega_{2^\mu} + \beta). \end{aligned}$$

Recall that $\bar{X}_0(x) = 1$. Therefore, Algorithm 5 outputs the desired polynomial with respect to $\bar{\mathbb{X}}$. Clearly, the computational complexity of Algorithm 3 is $O(\mu 2^\mu)$, and the evaluation of $\hat{f}(x)$ at a single point needs $O(2^\mu)$ operations. Finally, according to the properties of the subspace polynomial $s_\mu(x)$,

Algorithm 4 Frequency-Domain Modular Approach (FDMA)

Input: $\{\omega_{i-1}, u(\omega_{i-1})\}, i = 1, 2, \ldots, 2^\mu$.
Output: $(\lambda(x), z(x))$ that are represented with respect to $\bar{\mathbb{X}}$ and $\mathrm{rank}[\lambda(x), z(x)]$.

Initialization: $d_i^0 = u(\omega_{i-1}), g_i^0 = 1$ for $i = 1, 2, \ldots, 2^\mu$. $W(\omega_i) = 1, V(\omega_i) = 0$ for $i = 0, 1, \ldots, 2^{\mu-1}$, $r_1^0 = 0, r_2^0 = 1$.
for $j = 1, 2, \ldots, 2^\mu$ **do**
  if $g_j^{j-1} = 0$ or ($d_j^{j-1} \neq 0$ and $r_1^{j-1} < r_2^{j-1}$) **then**
    Let

$$\Psi_j(\omega_i) = \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ \omega_i - \omega_{j-1} & 0 \end{pmatrix}$$

    for $i \in \{0, 1, \ldots, 2^\mu - 1\}$.
    $r_1^j = r_2^{j-1}, r_2^j = r_1^{j-1} + 2$.
  else
    Let

$$\Psi_j(\omega_i) = \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ 0 & \omega_i - \omega_{j-1} \end{pmatrix}$$

    for $i \in \{0, 1, \ldots, 2^\mu - 1\}$.
    $r_1^j = r_1^{j-1}, r_2^j = r_2^{j-1} + 2$.
  end if
  for $i = j+1, j+2, \ldots, 2^\mu$ **do**

$$\begin{pmatrix} d_i^j \\ g_i^j \end{pmatrix} = \Psi_j(\omega_{i-1}) \begin{pmatrix} d_i^{j-1} \\ g_i^{j-1} \end{pmatrix}$$

  end for
  for $i = 0, 1, \ldots, 2^{\mu-1}$ **do**

$$\begin{pmatrix} W(\omega_i) \\ V(\omega_i) \end{pmatrix} = \Psi_j(\omega_i) \begin{pmatrix} W(\omega_i) \\ V(\omega_i) \end{pmatrix}$$

  end for
end for
if $r_1^{2^\mu} > r_2^{2^\mu}$ **then**
  $\lambda(\omega_i) = V(\omega_i), i = 0, 1, \ldots, 2^{\mu-1}$.
else
  $\lambda(\omega_i) = W(\omega_i), i = 0, 1, \ldots, 2^{\mu-1}$.
end if
$z(\omega_i) = \lambda(\omega_i) d_{i+1}^0, i = 0, 1, \ldots, 2^{\mu-1}$.
Call Algorithm 5 to obtain $\lambda(x)$ and $z(x)$.
return $(\lambda(x), z(x))$ and $\mathrm{rank}[\lambda(x), z(x)] = \min(r_1^{2^\mu}, r_2^{2^\mu})$.

the evaluation $s_\mu(\beta)$ or $s_\mu(v_\mu)$ has the same complexity as a field multiplication. Hence, the total computational complexity of Algorithm 5 is $O(\mu 2^\mu)$. □

Since $n-k = 2^\mu$, the complexity of calling Algorithm 5 twice in Algorithm 4 is $O((n-k)\log(n-k))$. It follows that the computational complexity of the FDMA is $O((n-k)^2)$.

E. Fast Modular Approach

In this subsection, we present the FMA for solving (10). The idea behind the FMA is that ϕ_i^j and Ψ_1^j need not to be

Algorithm 5 Extended IFFT$_{\bar{\mathbb{X}}}$

Input: $f(\omega_i + \beta), i = 0, 1, \ldots, 2^\mu$; μ, β.
Output: $f(x)$ represented in $\bar{\mathbb{X}}$.

Call Algorithm 3 with input $(f(\omega_0 + \beta), f(\omega_1 + \beta), \ldots, f(\omega_{2^\mu - 1} + \beta)), \mu, \beta$ to obtain $\hat{f}(x)$
Evaluate $\hat{f}(x)$ at $\omega_{2^\mu} + \beta$ to obtain $\hat{f}(\omega_{2^\mu} + \beta)$
Let

$$\begin{aligned} f(x) &= (f(\omega_{2^\mu} + \beta) - \hat{f}(\omega_{2^\mu} + \beta))\left(\bar{X}_{2^\mu}(x) - \frac{s_\mu(\beta)}{s_\mu(v_\mu)}\right) \\ &\quad + \hat{f}(x) \\ &= (f(\omega_{2^\mu} + \beta) - \hat{f}(\omega_{2^\mu} + \beta))\bar{X}_{2^\mu}(x) + \hat{f}(x) \\ &\quad - \frac{s_\mu(\beta)}{s_\mu(v_\mu)}(f(\omega_{2^\mu} + \beta) - \hat{f}(\omega_{2^\mu} + \beta))\bar{X}_0(x) \end{aligned}$$

return $f(x)$

computed at each iteration until enough information has been collected. Define the notation $\Psi_j^l = \Psi_l \Psi_{l-1} \cdots \Psi_j$ for any $1 \leq j \leq l \leq 2^\mu$. Recall that

$$\begin{aligned} \phi_i^j(W(x), N(x)) &= \phi_i \circ \psi_1^j(W(x), N(x)) \\ &= \phi_i((W(x), N(x))\Psi_1^j) \\ &= (W(\omega_{i-1}), N(\omega_{i-1})\Psi_1^j(\omega_{i-1}) \begin{pmatrix} d_i \\ g_i \end{pmatrix} \end{aligned}$$

and $\Psi_1^j = \Psi_j \Psi_{j-1} \cdots \Psi_1 = \Psi_{j/2+1}^j \Psi_1^{j/2}$ if j is even. Hence, if we first compute the irreducible basis matrix $\Psi_1^{2^{\mu-1}}$ in $\ker(\phi_1) \cap \ker(\phi_2) \cap \cdots \cap \ker(\phi_{2^{\mu-1}})$, which is a subproblem of (10), then Algorithm 2 can be used to obtain $\Psi_1^{2^{\mu-1}}(\omega_{i-1})$ for $i = 1, 2, \ldots, 2^\mu$ by setting $\beta = 0$. Next, we update $\phi_i^{2^{\mu-1}}$ for $i = 2^{\mu-1}+1, 2^{\mu-1}+2, \ldots, 2^\mu$. Given $\phi_i^{2^{\mu-1}}$ for $i = 2^{\mu-1}+1, 2^{\mu-1}+2, \ldots, 2^\mu$, we then compute $\Psi_{2^{\mu-1}+1}^{2^\mu}$, which can be obtained in a similar way. Finally, we obtain the product $\Psi_1^{2^\mu} = \Psi_{2^{\mu-1}+1}^{2^\mu} \Psi_1^{2^{\mu-1}}$ by the well-known convolution theorem. More precisely, if $\Psi_1^{2^{\mu-1}}(\omega_{i-1}), \Psi_{2^{\mu-1}+1}^{2^\mu}(\omega_{i-1})$ for $i = 1, 2, \ldots, 2^\mu + 1$ are available, $\Psi_1^{2^\mu}(\omega_{i-1})$ can be computed by simple matrix multiplication. Then Algorithm 5 can be used to obtain $\Psi_1^{2^\mu}$. Obviously, $\Psi_{2^{\mu-1}+1}^{2^\mu}(\omega_{i-1})$ can also be obtained using Algorithm 2.

We can generalize the above idea. In general, if $\omega_{i-1} = \omega_{i-j} + \omega_{j-1}$ for $i = j, j+1, \ldots, j+2^\mu-1$, then we have $(\omega_{j-1}, \omega_j, \ldots, \omega_{j+2^\mu-2}) = (\omega_0 + \omega_{j-1}, \omega_1 + \omega_{j-1}, \ldots, \omega_{2^\mu-1} + \omega_{j-1})$. Thus, Algorithm 2 can be used for evaluating a polynomial at points $\omega_{j-1}, \omega_j, \ldots, \omega_{j+2^\mu-2}$ by setting $\beta = \omega_{j-1}$. Hence, if we want to obtain $\Psi_j^{j+2^\mu-1}$ with input $\phi_j^{j-1}, \phi_{j+1}^{j-1}, \ldots, \phi_{j+2^\mu-1}^{j-1}$, we first compute $\Psi_j^{j+2^{\mu-1}-1}$. Then we update $\phi_i^{j+2^{\mu-1}-1}$ for $i = j+2^{\mu-1}, \ldots, j+2^\mu-1$. Next, based on these updated homomorphisms, we compute $\Psi_{j+2^{\mu-1}}^{j+2^\mu-1}$ by induction. Finally, we obtain $\Psi_j^{j+2^\mu-1} = \Psi_{j+2^{\mu-1}}^{j+2^\mu-1} \Psi_j^{j+2^{\mu-1}-1}$. A detailed description of this procedure is given in Algorithm 6. Note that since $\bar{X}_0(x) = 1$, we use 1 instead of $\bar{X}_0(x)$ for clarity.

Algorithm 6 Fast Modular Approach (FMA)

Input: $\{\omega_{i-1}, d_i^{j-1}, g_i^{j-1}\}, i = j, j+1, \ldots, j+2^\mu-1$, which satisfies $\omega_{i-1} = \omega_{i-j} + \omega_{j-1}$ and r_1^{j-1}, r_2^{j-1}.

Output: $\Psi_j^{j+2^\mu-1}, r_1^{j+2^\mu-1}, r_2^{j+2^\mu-1}$, where the polynomials are represented with respect to $\bar{\mathbb{X}}$;

1: **if** $\mu = 0$ **then**

2: **if** $g_j^{j-1} = 0$ or ($d_j^{j-1} \neq 0$ and $r_1^{j-1} < r_2^{j-1}$) **then**

3:

$$\Psi_j^{j+2^\mu-1} = \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ \omega_1 \bar{X}_1(x) - \omega_{j-1} & 0 \end{pmatrix}$$

4: $r_1^{j+2^\mu-1} = r_2^{j-1}, r_2^{j+2^\mu-1} = r_1^{j-1} + 2$

5: **else**

6:

$$\Psi_j^{j+2^\mu-1} = \begin{pmatrix} -g_j^{j-1} & d_j^{j-1} \\ 0 & \omega_1 \bar{X}_1(x) - \omega_{j-1} \end{pmatrix}$$

7: $r_1^{j+2^\mu-1} = r_1^{j-1}, r_2^{j+2^\mu-1} = r_2^{j-1} + 2$

8: **end if**

9: **else**

10: Call $\mathrm{FMA}(\{\omega_{i-1}, d_i^{j-1}, g_i^{j-1}\}, i = j, j+1, \ldots, j+2^{\mu-1}-1, r_1^{j-1}, r_2^{j-1})$ to obtain

$$(\Psi_j^{j+2^{\mu-1}-1}, r_1^{j+2^{\mu-1}-1}, r_2^{j+2^{\mu-1}-1})$$

11: Call Algorithm 2 to obtain

$$\Psi_j^{j+2^{\mu-1}-1}(\omega_{j-1}), \ldots, \Psi_j^{j+2^{\mu-1}-1}(\omega_{j+2^\mu-2})$$

and compute $\Psi_j^{j+2^{\mu-1}-1}(\omega_{j+2^\mu-1})$.

12: **for** $i = j+2^{\mu-1}, j+2^{\mu-1}+1, \ldots, j+2^\mu-1$ **do**

13:

$$\begin{pmatrix} d_i^{j+2^{\mu-1}-1} \\ g_i^{j+2^{\mu-1}-1} \end{pmatrix} = \Psi_j^{j+2^{\mu-1}-1}(\omega_{i-1}) \begin{pmatrix} d_i^{j-1} \\ g_i^{j-1} \end{pmatrix}.$$

14: **end for**

15: Let $l = j + 2^{\mu-1}$.

16: Call $\mathrm{FMA}(\{\omega_{h-1}, d_h^{l-1}, g_h^{l-1}\}, h = l, \ldots, l+2^{\mu-1}-1, r_1^{l-1}, r_2^{l-1})$ to obtain

$$(\Psi_l^{l+2^{\mu-1}-1}, r_1^{l+2^{\mu-1}-1}, r_2^{l+2^{\mu-1}-1})$$

17: Call Algorithm 2 to obtain

$$\Psi_l^{l+2^{\mu-1}-1}(\omega_{j-1}), \ldots, \Psi_l^{l+2^{\mu-1}-1}(\omega_{j+2^\mu-2})$$

and compute $\Psi_l^{l+2^{\mu-1}-1}(\omega_{j+2^\mu-1})$.

18: **for** $i = j, j+1, \ldots, j+2^\mu$ **do**

19:

$$\begin{aligned} &\Psi_j^{j+2^\mu-1}(\omega_{i-1}) \\ =&\Psi_l^{l+2^{\mu-1}-1}(\omega_{i-1}) \Psi_j^{j+2^{\mu-1}-1}(\omega_{i-1}). \end{aligned}$$

20: **end for**

21: Call Algorithm 5 to obtain each component of $\Psi_j^{j+2^\mu-1}$.

22: **end if**

23: **return** $\Psi_j^{j+2^\mu-1}, r_1^{j+2^\mu-1}, r_2^{j+2^\mu-1}$.

Lemma 7: Given the input $\{\omega_{i-1}, d_i^{j-1}, g_i^{j-1}\}, i = j, j+1, \ldots, j+2^\mu-1$, which satisfies $\omega_{i-1} = \omega_{i-j} + \omega_{j-1}$ and r_1^{j-1}, r_2^{j-1}, Algorithm 6 outputs $\Psi_j^{j+2^\mu-1}$, $r_1^{j+2^\mu-1}$, $r_2^{j+2^\mu-1}$.

Proof: The proof is by induction on μ. If $\mu = 0$, since we have $x = \omega_1 \bar{X}_1(x)$ and $\bar{X}_0(x) = 1$, then according to the proof of Lemma 5, Algorithm 6 outputs the desired answer $\Psi_j^j = \Psi_j$, r_1^j, r_2^j for any j. Therefore, the claim holds for $\mu = 0$.

Suppose that the claim holds for $0, 1, \ldots, \mu-1$. Then $\Psi_j^{j+2^{\mu-1}-1}$, $r_1^{j+2^{\mu-1}-1}$, $r_2^{j+2^{\mu-1}-1}$ can be obtained by the recursive call of the FMA in line 10 by induction. Since $\omega_{i-1} = \omega_{i-j} + \omega_{j-1}$, it follows that $(\omega_{j-1}, \omega_j, \ldots, \omega_{j+2^\mu-2}) = (\omega_0 + \omega_{j-1}, \omega_1 + \omega_{j-1}, \ldots, \omega_{2^\mu-1} + \omega_{j-1})$. Therefore, Algorithm 2 can be called for evaluating the matrix $\Psi_j^{j+2^{\mu-1}-1}$ at points $\omega_{j-1}, \ldots, \omega_{j+2^\mu-2}$ by setting $\tau = \mu$ and $\beta = \omega_{j-1}$. The evaluation $\Psi_j^{j+2^{\mu-1}-1}(\omega_{j+2^\mu-1})$ can be computed immediately. Because $\Psi_1^{j+2^{\mu-1}-1}(\omega_{i-1}) = \Psi_j^{j+2^{\mu-1}-1}(\omega_{i-1}) \Psi_1^{j-1}(\omega_{i-1})$, we have

$$\begin{aligned} \begin{pmatrix} d_i^{j+2^{\mu-1}-1} \\ g_i^{j+2^{\mu-1}-1} \end{pmatrix} &= \Psi_1^{j+2^{\mu-1}-1}(\omega_{i-1}) \begin{pmatrix} d_i \\ g_i \end{pmatrix} \\ &= \Psi_j^{j+2^{\mu-1}-1}(\omega_{i-1}) \begin{pmatrix} d_i^{j-1} \\ g_i^{j-1} \end{pmatrix} \end{aligned}$$

for $i = j+2^{\mu-1}, j+2^{\mu-1}+1, \ldots, j+2^\mu-1$. Let $l = j+2^{\mu-1}$. For $h = l, l+1, \ldots, l+2^{\mu-1}-1$, we have $\omega_{h-1} = \omega_{h-j} + \omega_{j-1} = \omega_{h-l} + \omega_{l-j} + \omega_{j-1} = \omega_{h-l} + \omega_{l-1}$, where the first and the last equalities hold by induction and the second equality is true because $h-l < 2^{\mu-1}$ and $\omega_{l-j} = \omega_{2^{\mu-1}}$. Hence, the recursive call of the FMA in line 16 outputs the desired $\Psi_l^{l+2^{\mu-1}-1}$, r_1^l, $r_2^{l+2^{\mu-1}-1}$ by induction.

Next, we show that the degrees of the components of $\Psi_j^{j+2^\mu-1}$ are less than or equal to 2^μ, which implies that $\Psi_j^{j+2^\mu-1}$ is determined uniquely by $\Psi_j^{j+2^\mu-1}(\omega_{i-1}), i = j, j+1, \ldots, j+2^\mu$. It is clear that this conclusion is true for $\mu = 0$. Suppose that it is also true for $1, 2, \ldots, \mu-1$. Then the degrees of the components of $\Psi_j^{j+2^\mu-1}$ must be less than or equal to 2^μ, since the degrees of the components of $\Psi_l^{l+2^{\mu-1}-1}$ and $\Psi_j^{j+2^{\mu-1}-1}$ are less than or equal to $2^{\mu-1}$ by induction. Hence, we can determine $\Psi_j^{j+2^\mu-1}$ by Algorithm 5 once $\Psi_j^{j+2^\mu-1}(\omega_{i-1}), i = j, j+1, \ldots, j+2^\mu$, have been obtained. This completes the proof. $\square$

Theorem 2: Given $\{\omega_{i-1}, d_i^0, g_i^0\}, i = 1, 2, \ldots, 2^\mu, r_1^0, r_2^0$, Algorithm 6 outputs $\Psi_1^{2^\mu}$, $r_1^{2^\mu}$, $r_2^{2^\mu}$.

Proof: Since ω_0 is the additive identity in $\mathbb{F}_{2^m}$, we have $\omega_{i-1} = \omega_{i-j} + \omega_{j-1}$ for $i = 1, 2, \ldots, 2^\mu$ and $j = 1$. The theorem then follows by Lemma 7. $\square$

For solving (10), we set $d_i^0 = u(\omega_{i-1}), g_i^0 = 1$ for $i = 1, 2, \ldots, 2^\mu$, $r_1^0 = 0, r_2^0 = 1$.

We now analyze the computational complexity of Algorithm 6. Denote the complexity of Algorithm 6 by $T(2^\mu)$. If $\mu = 0$, the algorithm outputs the solution in a

TABLE I

COMPLEXITY COMPARISON BETWEEN SYNDROME-BASED DECODING (RIBM) FOR THE (255, 223) RS CODE AND THE NEW DECODING (FDMA) FOR THE (256, 224) RS CODE OVER $\mathbb{F}_{2^8}$

Components	Syndrome-based decoding (RiBM)			New decoding (FDMA)		
	Mul.	Add.	Div.	Mul.	Add.	Div.
Syndrome	8,160	8,160	0	752	1,696	0
Key equation	3,136	1,568	0	3,233	2,244	0
Chien search	4,335	4,335	0	640	1,280	0
Formal derivative	0	0	0	80	80	0
Forney's formula	544	528	16	544	528	16
Total	16,175	14,591	16	5,249	5,828	16

TABLE II

COMPLEXITY COMPARISON BETWEEN SYNDROME-BASED DECODING (RIBM) FOR THE (1023, 895) RS CODE AND THE NEW DECODING (FDMA) FOR THE (1024, 896) RS CODE OVER $\mathbb{F}_{2^{10}}$

Components	Syndrome-based decoding (RiBM)			New decoding (FDMA)		
	Mul.	Add.	Div.	Mul.	Add.	Div.
Syndrome	130,944	130,944	0	4,160	9,088	0
Key equation	49,408	24,704	0	49,921	33,796	0
Chien search	66,495	66,495	0	3,584	7,168	0
Formal derivative	0	0	0	448	448	0
Forney's formula	8,320	8,256	64	8,320	8,256	64
Total	255,167	230,399	64	66,433	58,756	64

straightforward manner with complexity $O(1)$. Assume that $\mu > 1$. Two recursive calls take $2T(2^{\mu-1})$. The complexity of calling Algorithm 2 twice for evaluating $\Psi_l^{l+2^{\mu-1}-1}$ and $\Psi_j^{j+2^{\mu-1}-1}$ at points $\omega_{j-1}, \ldots, \omega_{j+2^{\mu}-2}$ is $O(\mu 2^{\mu})$, and the complexity of evaluating $\Psi_l^{l+2^{\mu-1}-1}$ and $\Psi_j^{j+2^{\mu-1}-1}$ at a single point $\omega_{j+2^{\mu}-1}$ is $O(2^{\mu})$. In addition, computing $\phi_i^{j+2^{\mu-1}-1}$ for $i = j + 2^{\mu-1}, \ldots, j + 2^{\mu} - 1$ involves $O(2^{\mu-1})$ operations, while the matrix multiplication between $\Psi_l^{l+2^{\mu-1}-1}$ and $\Psi_j^{j+2^{\mu-1}-1}$ in the frequency domain involves $O(2^{\mu})$ operations. Finally, the complexity of calling Algorithm 5 four times is $O(\mu 2^{\mu})$. It follows that $T(2^{\mu}) = 2T(2^{\mu-1}) + O(\mu 2^{\mu})$ and $T(2^{\mu}) = O(2^{\mu} \log^2(2^{\mu}))$. As $n-k = 2^{\mu}$, we have $T(n-k) = O((n-k)\log^2(n-k))$.

Evidently, the computational complexity of this new decoding algorithm is $O(n\log(n-k)+(n-k)\log^2(n-k))$. In the next section, we show that the FMA has a smaller constant factor than the Half-GCD algorithm proposed in [12]. This implies that the new algorithm has the smallest constant factor to date. It should be mentioned that although the complexity of the FDMA is $O((n-k)^2)$, it is more efficient for decoding short codes, which we shall see in the next section.

The complete decoding algorithm is presented in Algorithm 7. Note that this method can be generalized to arbitrary code length n and code dimension k and its complexity remains to be $O(n\log(n-k)+(n-k)\log^2(n-k))$. Detailed discussion is provided in [27].

Algorithm 7 Decoding Algorithm

Input: Received vector $\mathbf{r} = \mathbf{F} + \mathbf{e}$.
Output: The codeword $\mathbf{F}$.
1: Compute the syndrome polynomial $u(x)$ according to (9).
2: Evaluate $u(x)$ at points $\omega_0, \omega_1, \ldots, \omega_{2^{\mu}-1}$ by Algorithm 2.
3: Given $\phi_i(W(x), N(x)) = u(\omega_{i-1})W(\omega_{i-1}) + N(\omega_{i-1}), i = 1, 2, \ldots, 2^{\mu}$, compute the error locator polynomial $\lambda(x)$ and the error evaluator polynomial $z(x)$ by Algorithms 4 or 6.
4: Find the error locations by (11).
5: Compute the error pattern $\mathbf{e}$ by (12).
6: **return** $\mathbf{r} + \mathbf{e}$.

IV. COMPARISON AND ANALYSIS

In this section, we compare the proposed algorithm with other methods.

A. Comparison With Conventional Syndrome-Based Decoding

The most commonly used decoding algorithm for RS codes is syndrome-based decoding, which is based on Horner's rule. Here, we compare the algorithm described in Section III with syndrome-based decoding. Three RS codes, namely, the (255, 223) RS code over $\mathbb{F}_{2^8}$, the (1023, 895) RS code over $\mathbb{F}_{2^{10}}$, and the (4095, 3583) RS code over $\mathbb{F}_{2^{12}}$, are selected, and the comparisons are done by counting the numbers of field multiplications, additions, and divisions required by the two decoding algorithms. Since there is no field inversion in the FDMA or FMA, a modified BM algorithm that involves no field inversion, called the reformulated inversionless BM algorithm (RiBM), is chosen for comparison. Further discussion of the RiBM algorithm can be found in [28]. Tables I, II, and III present the comparisons in detail. According to these tables, the proposed algorithm saves 68%, 74%, and 90% multiplications and 60%, 74%, and 84% additions over $\mathbb{F}_{2^8}$, $\mathbb{F}_{2^{10}}$, and $\mathbb{F}_{2^{12}}$, respectively. Evidently, the proposed algorithm is 10 times faster than conventional decoding on a given machine when decoding (4095, 3583) RS codes. Note that the FMA is suitable for RS codes with a medium or long length, while the FDMA is more efficient when decoding short

TABLE III

COMPLEXITY COMPARISON BETWEEN SYNDROME-BASED DECODING (RIBM) FOR THE (4095, 3583) RS CODE AND THE NEW DECODING (FMA) FOR THE (4096, 3584) RS CODE OVER $\mathbb{F}_{2^{12}}$

Components	Syndrome-based decoding (RiBM)			New decoding (FMA)		
	Mul.	Add.	Div.	Mul.	Add.	Div.
Syndrome	2,096,640	2,096,640	0	21,248	45,568	0
Key equation	787,456	393,728	0	239,616	357,372	0
Chien search	1,052,415	1,052,415	0	18,432	36,864	0
Formal derivative	0	0	0	2,304	2,304	0
Forney's formula	131,584	131,328	256	131,584	131,328	256
Total	4,068,095	3,674,111	256	413,184	573,436	256

TABLE IV

COMPLEXITY OF SYNDROME COMPUTATION FOR RS CODES OVER $\mathbb{F}_{2^m}$

Field	Code	Method in [7]		Method in [8]		Method in [9]		Method in [10]	Proposed algorithm	
		Mul.	Add.	Mul.	Add.	Mul.	Add.	Mul.	Mul.	Add.
$\mathbb{F}_{2^8}$	$(255, 223)$	3,060	4,998	252	3,064	149	2,931	138	752	1,696
$\mathbb{F}_{2^{10}}$	$(1023, 895)$	33,620	73,185	2,868	19,339	824	36,981	/	4,160	9,088

RS codes.[2] It can be seen from Table III that the complexity of the FMA is significantly better than that of the RiBM algorithm for medium or long RS codes.

B. Comparison With Other RS Algorithms Based on FFT

There are many other efficient RS algorithms based on various FFT methods [7], [8], [9], [10]. Note that it has been shown in [29] that the additive FFT based on the Taylor expansion is worse than the FFT used here in terms of additive complexity. Thus, we do not consider the algorithm in [11] for comparison. Table IV compares the new decoding algorithm with the methods in [7], [8], [9], and [10] by counting the field operations in the syndrome computation. The result shows that the new algorithm has the lowest additive complexity and a medium multiplicative complexity. It should be mentioned that although some existing algorithms have a lower multiplicative complexity, they sacrifice a regular structure, which is vital in hardware implementation. The new decoding algorithm uses a FFT algorithm in which a butterfly structure is present; see [12] for details. This makes the new algorithm suitable for hardware implementation. Furthermore, the existing decoding algorithms have no fast key equation solver. Hence, the new algorithm is significantly better than them for decoding medium or long RS codes, as we have seen in Table III.

C. Comparison With the Half-GCD Algorithm and the Guruswami-Sudan Algorithm

The Half-GCD algorithm, proposed in [12], is able to solve (10) with complexity $O((n-k)\log^2(n-k))$. Although the FMA algorithm has the same complexity order as the Half-GCD algorithm, it has a smaller constant factor and a regular structure. It is clear that Algorithm 6 involves two recursive calls, matrix multiplications, eight times 2^μ-point $\mathrm{FFT}_{\bar{\mathbb{X}}}$, and four times Algorithm 5. As a 2^μ-point $\mathrm{FFT}_{\bar{\mathbb{X}}}$ involves $\frac{1}{2}\mu 2^\mu$ multiplications and $\mu 2^\mu$ additions, if we assume that the multiplication and addition have the same complexity, the constant factor of $\mathrm{FFT}_{\bar{\mathbb{X}}}$ is 1.5. In other words, a 2^μ-point $\mathrm{FFT}_{\bar{\mathbb{X}}}$ costs $1.5\mu 2^\mu$ field operations. Furthermore, the constant factor of Algorithm 5 is also 1.5. Hence, as the matrix multiplications involve $O(2^\mu)$ operations, we have

$$\begin{aligned} T(2^\mu) &= 2T(2^\mu/2) + (8+4)\times 1.5\mu 2^\mu + o(2^\mu \log(2^\mu)) \\ &< 2T(2^\mu/2) + 19\mu 2^\mu. \end{aligned}$$

This implies that the constant factor of Algorithm 6 is less than 9.5. For comparison, the Half-GCD algorithm involves two recursive calls, at least 15 times 2^μ-point $\mathrm{FFT}_{\bar{\mathbb{X}}}$, and 15 times 2^μ-point $\mathrm{IFFT}_{\bar{\mathbb{X}}}$. This means that the constant factor of Half-GCD is at least 22.5. Hence, the FMA has a significantly improved decoding complexity compared with Half-GCD. Moreover, with some effort, one can show that the FMA does not require that $n-k$ be a power of two. Therefore, the FMA is more flexible than Half-GCD in real applications.

Let the list size $\kappa = 1$ and the multiplicity $\upsilon = 1$. The GS algorithm is equivalent to bounded distance decoding. By taking the fast polynomial multiplication into account, fast interpolation algorithms were proposed in [16], [17], and [18] for solving the key equation of the GS algorithm. Their complexities are $O(n\log^2 n \log\log n)$. Compared with these interpolation algorithms, the method proposed here is faster since there is no factor $\log\log n$.

V. CONCLUSION

We have presented the MA, which is an efficient algorithm for solving the WB key equation. Based on the MA, a new decoding algorithm for RS codes has been proposed that has the best asymptotic computational complexity and the smallest constant factor achieved to date. The results of comparisons show that the new decoding algorithm is significantly better than the existing methods in terms of complexity when decoding practical RS codes. Since the complexity of the new algorithm is $O(n\log(n-k)+(n-k)\log^2(n-k))$, this makes

[2]The reason that the FMA performs worse than the FDMA for short codes is due to the hidden cost for dividing the problem and merging the solutions obtained from the subproblems when performing the divide and conquer approach (FMA).

it possible to use long RS codes in real applications. One potential route for future work is to transfer this new algorithm into a circuit design. Another interesting issue is to devise a fast list decoding algorithm based on the techniques presented here. Finally, whether the proposed algorithm can be used in the one-pass Chase decoding presented in [20] is open yet.

REFERENCES

[1] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[2] T. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, NJ, USA: Wiley, 2005.

[3] R. E. Blahut, *Algebraic Codes for Data Transmission*. New York, NY, USA: Cambridge Univ. Press, 2003.

[4] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[5] L. R. Welch and E. R. Berlekamp, "Error correction for algebraic block codes," U.S. Patent 4 633 470, Dec. 1986.

[6] S. V. Fedorenko and P. V. Trifonov, "Finding roots of polynomials over finite fields," *IEEE Trans. Commun.*, vol. 50, no. 11, pp. 1709–1711, Nov. 2002.

[7] T.-C. Lin, T. K. Truong, and P. D. Chen, "A fast algorithm for the syndrome calculation in algebraic decoding of Reed–Solomon codes," *IEEE Trans. Commun.*, vol. 55, no. 12, pp. 2240–2244, Dec. 2007.

[8] X. Wu, Z. Yan, and J. Lin, "Reduced-complexity decoders of long Reed–Solomon codes based on composite cyclotomic Fourier transforms," *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3920–3925, Jul. 2012.

[9] S. Bellini, M. Ferrari, and A. Tomasoni, "On the structure of cyclotomic Fourier transforms and their applications to Reed–Solomon codes," *IEEE Trans. Commun.*, vol. 59, no. 8, pp. 2110–2118, Aug. 2011.

[10] S. V. Fedorenko, "Efficient syndrome calculation via the inverse cyclotomic discrete Fourier transform," *IEEE Signal Process. Lett.*, vol. 26, no. 9, pp. 1320–1324, Sep. 2019.

[11] S. Gao and T. Mateer, "Additive fast Fourier transforms over finite fields," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6265–6272, Dec. 2010.

[12] S.-J. Lin, T. Y. Al-Naffouri, and Y. S. Han, "FFT algorithm for binary extension finite fields and its application to Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5343–5358, Oct. 2016.

[13] V. Guruswami and M. Sudan, "Improved decoding of Reed–Solomon and algebraic-geometry codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1757–1767, Sep. 1999.

[14] H. O'Keeffe and P. Fitzpatrick, "Gröbner basis solutions of constrained interpolation problems," *Linear Algebra Appl.*, vol. 351, pp. 533–551, Aug. 2002.

[15] K. Lee and M. E. O'Sullivan, "List decoding of Reed–Solomon codes from a Gröbner basis perspective," *J. Symbolic Comput.*, vol. 43, no. 9, pp. 645–658, 2008.

[16] M. Alekhnovich, "Linear diophantine equations over polynomials and soft decoding of Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2257–2265, Jul. 2005.

[17] P. Beelen and K. Brander, "Key equations for list decoding of Reed–Solomon codes and how to solve them," *J. Symbolic Comput.*, vol. 45, no. 7, pp. 773–786, Jul. 2010.

[18] M. F. I. Chowdhury, C.-P. Jeannerod, V. Neiger, É. Schost, and G. Villard, "Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2370–2387, May 2015.

[19] Y. Wu, "Fast chase decoding algorithms and architectures for Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 1, pp. 109–129, Jan. 2012.

[20] Y. Shany and A. Berman, "A Gröbner-bases approach to syndrome-based fast chase decoding of Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2300–2318, Apr. 2022.

[21] M. Morii and M. Kasahara, "Generalized key-equation of remainder decoding algorithm for Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 38, no. 6, pp. 1801–1807, Nov. 1992.

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

[23] D. Dabiri and I. F. Blake, "Fast parallel algorithms for decoding Reed–Solomon codes based on remainder polynomials," *IEEE Trans. Inf. Theory*, vol. 41, no. 4, pp. 873–885, Jul. 1995.

[24] P. Fitzpatrick, "On the key equation," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1290–1302, Sep. 1995.

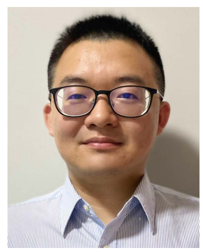
[25] S.-J. Lin, W.-H. Chung, and Y. S. Han, "Novel polynomial basis and its application to Reed–Solomon erasure codes," in *Proc. 55th Annu. Symp. Found. Comput. Sci. (FOCS)*, 2014, pp. 316–325.

[26] N. Tang and Y. Lin, "Fast encoding and decoding algorithms for arbitrary (n,k) Reed–Solomon codes over $\mathbb{F}_{2m}$," *IEEE Commun. Lett.*, vol. 24, no. 4, pp. 716–719, Apr. 2020.

[27] N. Tang and Y. S. Han, "New decoding of Reed–Solomon codes based on FFT and modular approach," 2022, *arXiv:2207.11079*.

[28] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed–Solomon decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 5, pp. 641–655, Oct. 2001.

[29] S.-J. Lin, T. Y. Al-Naffouri, Y. S. Han, and W.-H. Chung, "Novel polynomial basis with fast Fourier transform and its application to Reed–Solomon erasure codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 11, pp. 6284–6299, Nov. 2016.



Nianqi Tang received the Ph.D. degree in communication and information systems from Xidian University, Xi'an, China, in 2019. He joined at Huawei Technologies Company Ltd., where he is a Senior Engineer. His research interests include error control coding, network coding, and information theory.

Yunghsiang S. Han (Fellow, IEEE) was born in Taipei, Taiwan, in 1962. He received the B.Sc. and M.Sc. degrees in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and the Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, USA, in 1993.

From 1986 to 1988, he was a Lecturer at the Ming-Hsin Engineering College, Hsinchu. He was a Teaching Assistant from 1989 to 1992 and a Research Associate at the School of Computer and Information Science, Syracuse University, from 1992 to 1993. From 1993 to 1997, he was an Associate Professor at the Department of Electronic Engineering, Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering, National Chi Nan University, Nantou, Taiwan, from 1997 to 2004. He was promoted to a Professor in 1998. He was a Visiting Scholar at the Department of Electrical Engineering, University of Hawaii, Manoa, HI, USA, from June to October 2001, the SUPRIA Visiting Research Scholar at the Department of Electrical Engineering and Computer Science and the CASE Center, Syracuse University, NY, USA, from September 2002 to January 2004 and July 2012 to June 2013, and a Visiting Scholar at the Department of Electrical and Computer Engineering, University of Texas at Austin, TX, USA, from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering, National Taipei University, Taipei, from August 2004 to July 2010. From August 2010 to January 2017, he was a Chair Professor at the Department of Electrical Engineering, National Taiwan University of Science and Technology. From February 2017 to February 2021, he was with the School of Electrical Engineering and Intelligentization, Dongguan University of Technology, China. Currently, he is with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China. He has been a Chair Professor with National Taipei University, since February 2015. His research interests include error-control coding, wireless networks, and security. He received the 1994 Syracuse University Doctoral Prize. One of his papers won the Prestigious 2013 ACM CCS Test-of-Time Award in cybersecurity.