Generalized Inverse Discrete Fourier Transform with Application to Goppa Codes

Nianqi Tang, Yunghsiang S. Han *Fellow, IEEE*, Chao Chen, and Danyang Pei

***Abstract*—This paper introduces the concept of the generalized inverse discrete Fourier transform over a finite field and studies its application to the decoding of codes. It also proposes an efficient algorithm for computing the generalized inverse discrete Fourier transform, called the fast generalized inverse discrete Fourier transform. Based on this fast transform, the paper shows that, for any (n, k) generalized Reed–Solomon code, if the underlying field contains an additive or multiplicative subgroup of order $2^\mu \geq n-k$ and n is proportional to the field size, then there exists a decoding algorithm with complexity $O\big(n\log(n-k)+(n-k)\log^2(n-k)\big)$. The paper further discusses the additive fast generalized inverse discrete Fourier transform over binary extension fields, which leads to a fast decoding algorithm for binary Goppa codes with complexity $O(n\log r + r\log^2 r)$, where n is the code length and r is the degree of the Goppa polynomial. This achieves the best known decoding complexity for binary Goppa codes to date. A complexity comparison shows that the proposed algorithm reduces the number of field operations by 88% compared to traditional methods when decoding the Goppa code with parameters $n = 8192$ and $r = 128$, which is used in the McEliece cryptosystem as a candidate scheme for post-quantum cryptography.**

***Index Terms*—Discrete Fourier transform, inverse discrete Fourier transform, generalized inverse discrete Fourier transform, generalized Reed–Solomon codes, decoding algorithm, Goppa codes.**

I. Introduction

GENERALIZED Reed–Solomon (GRS) codes are an important class of error-correcting codes that occupy a prominent place in the theory and practice of error correction. Theoretically, many well-known codes are subclasses or subfield subcodes of GRS codes, including Reed–Solomon (RS) codes and Goppa codes. In practice, GRS codes have relatively simple, implementable encoding and decoding techniques, making them highly competitive in real-world applications.

An (n, k) GRS code over $GF(q)$ can be defined by

$$\{(w_0 f(\alpha_0), w_1 f(\alpha_1), \ldots, w_{n-1} f(\alpha_{n-1})) \mid \deg(f(x)) < k\},$$

where $w_0, w_1, \ldots, w_{n-1}$ are nonzero elements in the field $GF(q)$, $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ are n distinct elements in $GF(q)$ and $f(x) \in GF(q)[x]$. Clearly, GRS codes can be characterized by the discrete Fourier transform (DFT) over finite fields since the DFT establishes the relationship between the polynomial $f(x) \in GF(q)[x]$ and its evaluations $f(\alpha_i)$, where $f(x)$ is usually called the *time domain* and its evaluations are usually called the *frequency domain*. In designing a decoding algorithm for GRS codes, an important approach is to investigate the time domain from its frequency domain. Over the past few years, the fast Fourier transform (FFT), a fast implementation of the DFT, has increasingly played a central role in the design of efficient decoding algorithms for GRS codes. For example, Gao proposed an FFT-based decoding algorithm for RS codes in [1] (Note that RS codes are a subclass of GRS codes). Justesen derived a decoding algorithm for RS codes over the Fermat field $GF(q)$ in which $q = 2^{2^m} + 1$ for some integer m, whose complexity is $O(q\log^2(q))$ [2]. An algorithm for RS codes based on the cyclotomic Fourier transform was derived in [3]. Decoding algorithms for an (n, k) RS code based on the additive FFT over $GF(2^m)$ were presented in both [4] and [5], whose complexity are $O(n\log(n-k)+(n-k)\log^2(n-k))$. Among these known methods, a common requirement is that an n-point FFT must be available when decoding an (n, k) GRS code over $GF(q)$. This imposes a severe parameter limitation on GRS codes for which an efficient decoding algorithm exists, since an n-point FFT over $GF(q)$ usually requires that n be smooth (i.e., equal to a product of small primes) and that a subgroup of order n exist in $GF(q)$. However, contrary to intuition, the requirement of n-point FFT is unnecessary for designing an efficient algorithm for (n, k) GRS codes, as we will show. Recall that an (n, k) GRS code over $GF(q)$ is a vector subspace of $GF(q)^n$. Then the decoding can be accomplished by determining the coset of the code to which the received vector belongs. Let $\boldsymbol{\gamma}$ denote the received vector, which can be seen as the frequency domain (We ignore the elements $w_0, w_1, \ldots, w_{n-1}$ here), and let $f(x)$ denote the time domain corresponding to $\boldsymbol{\gamma}$. If we use division with remainder in the Euclidean domain $GF(q)[x]$ to write

$$f(x) = f_\theta(x)\theta(x) + \eta(x),$$

where $f_\theta(x)$ is the quotient, $\theta(x)$ is some fixed divisor such that $\deg(\theta(x)) = k$ and $\eta(x)$ is the remainder such that $\deg(\eta(x)) < k$, the coset to which $\boldsymbol{\gamma}$ belongs is totally determined by the quotient $f_\theta(x)$ since the frequency domain of $\eta(x)$ is contained in the code (Recall the definition of GRS codes). This implies that, to decode the received vector, it suffices to investigate only the quotient $f_\theta(x)$, rather than the whole time domain $f(x)$. Based on this observation, we propose the concept of the generalized inverse discrete Fourier transform (GIDFT), which is essentially a powerful

A part of the material of this paper has been presented at the 11th International Workshop on Signal Design and its Applications in Communications (IWSDA 2025). This work was supported by the National Key Research and Development Program of China under Grant 2022YFA1004902.

Nianqi Tang (724973040@qq.com), Yunghsiang S. Han (yunghsiangh@gmail.com), and Danyang Pei (202322280103@std.uestc.edu.cn) are with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen, China. Chao Chen (cchen@xidian.edu.cn) is with the State Key Lab of ISN, Xidian University, Xi'an, China.

technique to investigate the quotient $f_\theta(x)$ for various $\theta(x)$. Indeed, the idea of investigating $f_\theta(x)$ had already been used in the conventional decoding algorithm of RS codes, which essentially computes the high-degree coefficients of $f(x)$ and is equivalent to the special case $\theta(x) = x^k$. However, the GIDFT is a method for investigating the quotient for various $\theta(x)$. This flexibility enables the design of efficient decoding algorithms. As we shall show, several new results can be derived using the GIDFT.

In this paper, we first propose the concept of the GIDFT and investigate its properties. Then we propose a fast algorithm for computing the GIDFT, called the fast generalized inverse discrete Fourier transform (FGIDFT). The FGIDFT is employed to decode GRS codes, resulting in a new decoding algorithm. Notably, this new decoding algorithm does not require the availability of an n-point FFT when decoding an (n, k) GRS code. Instead, an $(n-k)$-point FFT suffices. We show that, if $GF(q)$ contains an additive or a multiplicative subgroup of order 2^μ such that $2^\mu \geq n-k$, then for any (n, k) GRS code over $GF(q)$, there exists a decoding algorithm whose complexity is $O(n\log(n-k)+(n-k)\log^2(n-k))$ provided n is proportional to q. This result significantly extends the parameter range of GRS codes that admit fast-decoding algorithms. For example, the prime field $GF(593)$ contains a multiplicative subgroup of order 16. Then a $(592, 576)$ RS code defined over $GF(593)$ admits a fast decoding algorithm, though a 592-point FFT is not available in $GF(593)$. Since the binary extension field $GF(2^m)$ is commonly used in practice, we also discuss the FGIDFT for $GF(2^m)$ in detail, which is called the additive FGIDFT in this paper since it relies on an additive subgroup of $GF(2^m)$. Based on the additive FGIDFT, this paper derives a fast decoding algorithm for binary Goppa codes, whose complexity is $O(n\log(r) + r\log^2(r))$, where n is the code length and r is the degree of the Goppa polynomial. This achieves the best known complexity to date. We also provide complexity comparisons among the proposed algorithm, the Patterson algorithm presented in [6], and the decoding method presented in [7], [8] when decoding the Goppa codes that have been used in the McEliece cryptosystem as a candidate for post-quantum cryptography to be submitted to National Institute of Standards and Technology (NIST), see [9]. When decoding the Goppa code in which $n = 8192$ and $r = 128$, the proposed algorithm reduces the number of field operations by 88% compared to other methods. This shows that the proposed algorithm is superior in terms of computational complexity for practical use.

The main contributions of this paper can be summarized as follows:

1) We propose the concept of the generalized inverse discrete Fourier transform and investigate its properties.
2) A fast generalized inverse discrete Fourier transform algorithm is derived.
3) We prove that, if $GF(q)$ contains an additive or a multiplicative subgroup of order 2^μ such that $2^\mu \geq n-k$ and if n is proportional to q, then for any (n, k) GRS code defined over $GF(q)$, there is a fast decoding algorithm whose complexity is $O(n\log(n-k)+(n-k)\log^2(n-k))$.
4) A detailed description of the additive FGIDFT is provided. Based on the additive FGIDFT, we derive a fast decoding algorithm for binary Goppa codes, whose complexity is $O(n\log(r) + r\log^2(r))$, where n is the code length and r is the degree of the Goppa polynomial. This achieves the best known complexity to date. Complexity comparisons between the new method and others in the literature also demonstrate that the new one is superior in terms of complexity when decoding practical Goppa codes.

The paper is organized as follows. Section II proposes the concept of the GIDFT and investigates its properties. Section III derives the FGIDFT, which computes the GIDFT efficiently. In Section IV, we present a detailed description of the FGIDFT when $GF(q)$ has a characteristic two. Section V applies the FGIDFT to decode GRS codes. Section VI derives a new decoding algorithm for binary Goppa codes. Complexity comparisons between the new method and others in the literature are also provided. Section VII presents several comments, along with some open issues. Finally, Section VIII concludes the paper.

II. Generalized Inverse Discrete Fourier Transform

In this section, we introduce the concept of the generalized inverse discrete Fourier transform (GIDFT) and investigate its properties. Before beginning the discussion, we review the concepts of the discrete Fourier transform and its inverse, the inverse discrete Fourier transform.

Definition 1 ([10]): Let $\boldsymbol{f} = (f_0, f_1, \ldots, f_{n-1})$ be a vector over $GF(q)$, and let α be an element of $GF(q)$ of order n. The ***discrete Fourier transform*** (DFT) of the vector $\boldsymbol{f}$ is the vector $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, \ldots, \gamma_{n-1})$ with components given by

$$\gamma_j = \sum_{i=0}^{n-1} \alpha^{ij} f_i, j = 0, 1, \ldots, n-1.$$

The ***inverse discrete Fourier transform*** (IDFT) of $\boldsymbol{\gamma}$ is defined by

$$f_i = \frac{1}{n}\sum_{j=0}^{n-1} \alpha^{-ij}\gamma_j,$$

where n is interpreted as an integer of the field (i.e., n is the sum of n copies of the multiplicative identity 1 in $GF(q)$). □

Notably, if we write the polynomial $f(x) \in GF(q)[x]$ as

$$f(x) = f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + \cdots + f_1 x + f_0,$$

then γ_j is equal to the evaluation of $f(x)$ at α^j, that is,

$$\gamma_j = f(\alpha^j), j = 0, 1, \ldots, n-1.$$

This implies that the DFT is mathematically equivalent to evaluating a polynomial.

In Definition 1, the DFT and IDFT are defined over a multiplicative subgroup of order n of $GF(q)$. Since the generalized inverse discrete Fourier transform can be defined over an arbitrary subset of $GF(q)$, Definition 1 is insufficient to establish the relationship between the DFT/IDFT and GIDFT.

In the following, we redefine the DFT and IDFT over an arbitrary subset of $GF(q)$ by polynomial evaluations.

Definition 2: Let J be a subset of the field $GF(q)$ with n distinct elements. For a polynomial $f(x) \in GF(q)[x]$ of degree less than n, the DFT of $f(x)$ over J, denoted by $\mathrm{DFT}_J(f(x))$, computes the vector

$$(f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{n-1})),$$

where $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ represent n distinct elements in J. □

The DFT over J is an isomorphism that maps the vector space $GF(q)^n$ to $GF(q)^n$ (Recall that the set $\{f(x) \mid f(x) \in GF(q)[x], \deg(f(x)) < n\}$ can be regarded as the vector space $GF(q)^n$). It should be noted that the DFT depends on the set J as well as the order of its elements $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$. Without loss of generality, we shall assume that every subset of $GF(q)$ has a fixed order throughout this paper, unless otherwise specified. In particular, $J = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ and $GF(q) = \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$. Under this assumption, DFT_J is unique.

Definition 3: Given a vector $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}) \in GF(q)^n$, the IDFT of $\boldsymbol{\gamma}$ over J, denoted by $\mathrm{IDFT}_J(\boldsymbol{\gamma})$, computes the polynomial

$$f(x) = \sum_{i=0}^{n-1} \gamma_i \frac{\prod_{j \neq i}(\alpha_i - \alpha_j)^{-1} \prod_{j=0}^{n-1}(x - \alpha_j)}{x - \alpha_i}. \quad (1)$$

□

It is straightforward to see that the IDFT over J is the inverse map of the DFT over J by the Lagrange interpolation. So IDFT_J is an isomorphism from $GF(q)^n$ to $GF(q)^n$ too.

We are now ready to define the generalized inverse discrete Fourier transform.

Definition 4: Let J be a subset of $GF(q)$ with n distinct elements. Given a polynomial $T(x) \in GF(q)[x]$, the ***generalized inverse discrete Fourier transform*** (GIDFT) with respect to J and $T(x)$ is a map from $GF(q)^n$ to $GF(q)[x]$ defined by

$$\Phi_{J,T}(\boldsymbol{\gamma}) = \sum_{i=0}^{n-1} \gamma_i \frac{T(x) - T(\alpha_i)}{x - \alpha_i}, \quad (2)$$

where $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ are n distinct elements in J and $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}) \in GF(q)^n$.

Since $(x - \alpha_i) | (T(x) - T(\alpha_i))$, $\Phi_{J,T}(\boldsymbol{\gamma})$ is a polynomial in $GF(q)[x]$ of degree less than $\deg(T(x))$ for any vector $\boldsymbol{\gamma} \in GF(q)^n$. Furthermore, it is easy to verify that, for any vectors $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2 \in GF(q)^n$ and any scalar $\beta \in GF(q)$,

$$\begin{aligned} &\Phi_{J,T}(\boldsymbol{\gamma}_1 + \boldsymbol{\gamma}_2) = \Phi_{J,T}(\boldsymbol{\gamma}_1) + \Phi_{J,T}(\boldsymbol{\gamma}_2) \\ \text{and } &\Phi_{J,T}(\beta \cdot \boldsymbol{\gamma}_1) = \beta \cdot \Phi_{J,T}(\boldsymbol{\gamma}_1), \end{aligned}$$

where $\cdot$ denotes the scalar multiplication. Hence, $\Phi_{J,T}$ is a homomorphism from $GF(q)^n$ to $GF(q)^{\deg T(x)}$ (As we discussed before, the set $\{f(x) \mid f(x) \in GF(q)[x], \deg(f(x)) < \deg(T(x))\}$ can be regarded as $GF(q)^{\deg(T(x))}$).

Compared with the IDFT, the GIDFT is a homomorphism rather than an isomorphism. However, when $\deg(T(x))$ is equal to n, $\Phi_{J,T}$ becomes an isomorphism.

Theorem 1: If $\deg(T(x)) = n$, then $\Phi_{J,T}$ is an isomorphism.

Proof: When $\deg(T(x)) = n$, it is clear that the domain and range of $\Phi_{J,T}$ have the same dimension. Hence, to prove the claim, it suffices to prove that $\Phi_{J,T}(\boldsymbol{\gamma}) = 0$ if and only if $\boldsymbol{\gamma} = \mathbf{0}$, where $\mathbf{0}$ denotes the zero vector.

If we write $T(x) = \sum_{j=0}^{n} T_j x^j$, we have

$$\begin{aligned} T(x) - T(\alpha_i) &= \sum_{j=0}^{n} T_j (x^j - \alpha_i^j) \\ &= \sum_{j=1}^{n} T_j (x - \alpha_i) \sum_{l=0}^{j-1} x^{j-1-l} (\alpha_i)^l. \end{aligned}$$

This leads to

$$\frac{T(x) - T(\alpha_i)}{x - \alpha_i} = \sum_{j=1}^{n} T_j \sum_{l=0}^{j-1} x^{j-1-l} \alpha_i^l.$$

It follows that

$$\begin{aligned} \Phi_{J,T}(\boldsymbol{\gamma}) &= \sum_{i=0}^{n-1} \gamma_i \frac{T(x) - T(\alpha_i)}{x - \alpha_i} \\ &= \sum_{i=0}^{n-1} \gamma_i \sum_{j=1}^{n} T_j \sum_{l=0}^{j-1} x^{j-1-l} \alpha_i^l \\ &= \sum_{j=1}^{n} T_j \sum_{l=0}^{j-1} x^{j-1-l} \sum_{i=0}^{n-1} \gamma_i \alpha_i^l \\ &= \sum_{j=1}^{n} T_j \sum_{l=0}^{j-1} x^{j-1-l} a_l, \end{aligned} \quad (3)$$

where $a_l = \sum_{i=0}^{n-1} \gamma_i \alpha_i^l$.

Now suppose $\Phi_{J,T}(\boldsymbol{\gamma}) = 0$ for some $\boldsymbol{\gamma}$. The coefficient of x^b in $\Phi_{J,T}(\boldsymbol{\gamma})$ is equal to

$$\sum_{j=b+1}^{n} T_j a_{j-1-b}.$$

So the coefficient of x^{n-1} in $\Phi_{J,T}(\boldsymbol{\gamma})$ is $T_n a_0$. Since $T_n \neq 0$, $a_0 = 0$. Assume that $a_0, a_1, \ldots, a_{h-1} = 0$. The coefficient of x^{n-h-1} in $\Phi_{J,T}(\boldsymbol{\gamma})$ satisfies

$$\begin{aligned} &\sum_{j=n-h}^{n} T_j a_{j-n+h} \\ &= T_{n-h} a_0 + T_{n-h+1} a_1 + \cdots + T_{n-1} a_{h-1} + T_n a_h \\ &= 0, \end{aligned}$$

which implies $a_h = 0$. By induction on h, we have $a_l = 0$ for all $l = 0, 1, \ldots, n-1$. According to the definition of a_l, the vector $(a_0, a_1, \ldots, a_{n-1})$ is related to $(\gamma_0, \gamma_1, \ldots, \gamma_{n-1})$ via a Vandermonde matrix. Hence, we can conclude that $(\gamma_0, \gamma_1, \ldots, \gamma_{n-1}) = \mathbf{0}$.

Conversely, if $(\gamma_0, \gamma_1, \ldots, \gamma_{n-1}) = \mathbf{0}$, it is straightforward to see that $\Phi_{J,T}(\boldsymbol{\gamma}) = 0$. This completes the proof. □

Comparing the IDFT (1) with the GIDFT (2), if we take $T(x) = \prod_{i=0}^{n-1}(x - \alpha_i)$ and omit the scalars $\prod_{j \neq i}(\alpha_i - \alpha_j)^{-1}$, they have the same form. This partly explains the term *generalized*. Furthermore, if we let $T(x) = \prod_{\alpha \in GF(q)}(x - \alpha)$, then $\Phi_{J,T}(\boldsymbol{\gamma})$ is equal to $\mathrm{IDFT}_{GF(q)}((\boldsymbol{\gamma}, \mathbf{0}))$, where $(\boldsymbol{\gamma}, \mathbf{0})$

denotes a concatenation of $\boldsymbol{\gamma}$ and a zero vector $\mathbf{0}$ of length $q-n$.

Lemma 1: Let

$$T(x) = \prod_{\alpha \in GF(q)} (x - \alpha).$$

Then $\Phi_{J,T}(\boldsymbol{\gamma})$ is equal to $\text{IDFT}_{GF(q)}((\boldsymbol{\gamma}, \mathbf{0}))$. Furthermore, if we write $f(x) = \Phi_{J,T}(\boldsymbol{\gamma})$, then

$$f(\alpha_i) = \gamma_i, \text{ for all } \alpha_i \in J.$$

Proof: One can write the field $GF(q)$ as

$$\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}, \alpha_n, \ldots, \alpha_{q-1}\},$$

where $\alpha_0, \alpha_1, \ldots, \alpha_{n-1} \in J$. For every $\boldsymbol{\gamma} \in GF(q)^n$, we can construct a vector $(\boldsymbol{\gamma}, \mathbf{0})$ from $\boldsymbol{\gamma}$ by concatenating a zero vector of length $q-n$. Since $T(\alpha) = 0$ for all $\alpha \in GF(q)$ and $J \subseteq GF(q)$, we have

$$f(x) = \sum_{i=0}^{n-1} \gamma_i \frac{\prod_{\alpha \in GF(q)} (x - \alpha)}{x - \alpha_i}.$$

As the multiplication of all nonzero elements in $GF(q)$ is equal to 1,

$$\prod_{\substack{\alpha \in GF(q) \\ \alpha \neq \alpha_i}} (\alpha_i - \alpha) = 1.$$

It follows that

$$\begin{aligned} f(x) &= \sum_{i=0}^{n-1} \gamma_i \frac{\prod_{\substack{\alpha \in GF(q) \\ \alpha \neq \alpha_i}} (\alpha_i - \alpha)^{-1} \prod_{\alpha \in GF(q)} (x - \alpha)}{x - \alpha_i} \\ &= \sum_{i=0}^{n-1} \gamma_i \frac{\prod_{\substack{\alpha \in GF(q) \\ \alpha \neq \alpha_i}} (\alpha_i - \alpha)^{-1} \prod_{\alpha \in GF(q)} (x - \alpha)}{x - \alpha_i} \\ &\quad + \sum_{i=n}^{q-1} 0 \frac{\prod_{\substack{\alpha \in GF(q) \\ \alpha \neq \alpha_i}} (\alpha_i - \alpha)^{-1} \prod_{\alpha \in GF(q)} (x - \alpha)}{x - \alpha_i} \\ &= \text{IDFT}_{GF(q)}((\boldsymbol{\gamma}, \mathbf{0})). \end{aligned}$$

Finally, it is easy to check that $f(\alpha_i) = \gamma_i$ for all $\alpha_i \in J$ by substituting α_i into $f(x)$. This completes the proof. □

We have shown that for a specific polynomial $T(x) = \prod_{\alpha \in GF(q)} (x - \alpha)$, the GIDFT can be related to the IDFT. In the following, we shall prove that, for a general $T(x)$, the GIDFT can also be related to the IDFT. More precisely, there exists an invertible linear transformation between the coefficients of $\Phi_{J,T}(\boldsymbol{\gamma})$ and the high-degree coefficients of $\text{IDFT}_{GF(q)}((\boldsymbol{\gamma}, \mathbf{0}))$. Hereafter, the notation $\Phi_{J,q}$ will denote the GIDFT with respect to J and $\prod_{\alpha \in GF(q)} (x - \alpha)$. Furthermore, since there are q elements in $GF(q)$, we shall assume that $T(x)$ has degree at most q.

Theorem 2: Let $t = \deg(T(x)) \leq q$, $f(x) = \Phi_{J,T}(\boldsymbol{\gamma})$ and $f_q(x) = \Phi_{J,q}(\boldsymbol{\gamma})$. Let

$$f(x) = \sum_{j=0}^{t-1} f_j x^j, f_q(x) = \sum_{j=0}^{q-1} f_j^{'} x^j.$$

Then there exists a $t \times t$ invertible matrix F such that

$$(f_0, f_1, \ldots, f_{t-1}) = (f_{q-t}^{'}, f_{q-t+1}^{'}, \ldots, f_{q-1}^{'}) F.$$

In other words, the coefficients of $\Phi_{J,T}(\boldsymbol{\gamma})$ can be related to the high-degree coefficients of $\Phi_{J,q}(\boldsymbol{\gamma})$ by an invertible linear transformation, whose matrix with respect to the standard basis is equal to F.

Proof: We write the elements of $GF(q)$ in a fixed order as

$$\{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}.$$

Let $\rho(x) = \prod_{i=0}^{t-1} (x - \alpha_i)$. This gives

$$\prod_{\alpha \in GF(q)} (x - \alpha) = \rho(x)\theta(x),$$

where $\theta(x) = \prod_{i=t}^{q-1} (x - \alpha_i)$. It follows that

$$\frac{\prod_{\alpha \in GF(q)} (x - \alpha)}{x - \alpha_i} = \frac{\rho(x)\theta(x)}{x - \alpha_i}. \tag{4}$$

Given the polynomial

$$\frac{\rho(x) - \rho(\alpha_i)}{x - \alpha_i},$$

by polynomial addition, there is $\eta(x)$ such that

$$\frac{\prod_{\alpha \in GF(q)} (x - \alpha)}{x - \alpha_i} = \frac{\rho(x) - \rho(\alpha_i)}{x - \alpha_i}\theta(x) + \eta(x). \tag{5}$$

Subtracting (4) from (5) yields

$$\frac{(x - \alpha_i)\eta(x) - \rho(\alpha_i)\theta(x)}{x - \alpha_i} = 0.$$

If $\rho(\alpha_i) = 0$, $\eta(x) = 0$. If $\rho(\alpha_i) \neq 0$, one must have $\deg(\eta(x)) < \deg(\theta(x))$. So in either case, $\frac{\rho(x) - \rho(\alpha_i)}{x - \alpha_i}$ is the quotient of $\frac{\prod_{\alpha \in GF(q)} (x - \alpha)}{x - \alpha_i}$ divided by $\theta(x)$. By multiplying the scalars γ_i and summing over i, one can conclude that the GIDFT with respect to J and $\rho(x)$, say $\Phi_{J,\rho}(\boldsymbol{\gamma})$, is the quotient of $f_q(x)$ divided by $\theta(x)$.

Let $f_\rho(x) = \Phi_{J,\rho}(\boldsymbol{\gamma})$. If we write

$$f_\rho(x) = \sum_{j=0}^{t-1} f_j^* x^j,$$

since $\theta(x)$ is monic and the polynomial division is linear, there must exist a $t \times t$ lower triangular matrix $F^{'}$ such that

$$(f_0^*, f_1^*, \ldots, f_{t-1}^*) = (f_{q-t}^{'}, f_{q-t+1}^{'}, \ldots, f_{q-1}^{'}) F^{'}.$$

More precisely, if we write

$$\theta(x) = x^{q-t} + \theta_{q-t-1} x^{q-t-1} + \cdots + \theta_1 x + \theta_0,$$

then $F^{'}$ can be written as (6). All elements on the main diagonal of $F^{'}$ are equal to one. Clearly, $F^{'}$ is invertible.

It remains to show that there is an invertible linear transformation which maps $(f_0^*, f_1^*, \ldots, f_{t-1}^*)$ to $(f_0, f_1, \ldots, f_{t-1})$. By analogy with (3), we can write

$$f(x) = \sum_{j=1}^{t} T_j \sum_{l=0}^{j-1} x^{j-1-l} a_l, f_\rho(x) = \sum_{j=1}^{t} \rho_j \sum_{l=0}^{j-1} x^{j-1-l} a_l,$$

where $a_l = \sum_{i=0}^{n-1} \gamma_i \alpha_i^l$. The coefficients of x^b in $f(x)$ and $f_\rho(x)$ are

$$\sum_{j=b+1}^{t} T_j a_{j-1-b} \text{ and } \sum_{j=b+1}^{t} \rho_j a_{j-1-b},$$

$$F^{'} = \begin{pmatrix} 1 & & & & & \\ -\theta_{q-t-1} & 1 & & & & \\ \theta_{q-t-1}^2 - \theta_{q-t-2} & -\theta_{q-t-1} & 1 & & & \\ \vdots & & & \ddots & & \\ & & \iddots & & 1 & \\ & & & & -\theta_{q-t-1} & 1 \\ * & & & \cdots & \theta_{q-t-1}^2 - \theta_{q-t-2} & -\theta_{q-t-1} & 1 \end{pmatrix}. \tag{6}$$

respectively. If $f(x) = 0$, one can obtain that $a_l = 0$ for $l = 0, 1, \ldots, t-1$ by a similar method in Theorem 1. This leads to $f_\rho(x) = 0$. Conversely, $f_\rho(x) = 0$ also implies $f(x) = 0$. Hence, both $\Phi_{J,T}$ and $\Phi_{J,\rho}$ are linear transformations from $GF(q)^n$ to $GF(q)^t$, and they have the same kernel, say $\ker(\Phi_{J,\rho})$. In addition, $a_l = 0$ for all $l = 0, 1, \ldots, t-1$ implies that

$$\sum_{i=0}^{n-1} \gamma_i \alpha_i^l = 0, \ l = 0, 1, \ldots, t-1,$$

which tells us that $\boldsymbol{\gamma}^T$ belongs to the nullspace of an $t \times n$ Vandermonde matrix

$$\begin{pmatrix} \alpha_0^0 & \alpha_1^0 & \ldots & \alpha_{n-1}^0 \\ \alpha_0^1 & \alpha_1^1 & \ldots & \alpha_{n-1}^1 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_0^{t-1} & \alpha_1^{t-1} & \ldots & \alpha_{n-1}^{t-1} \end{pmatrix}.$$

If $n \geq t$, the dimension of $\ker(\Phi_{J,\rho})$ is equal to $n - t$, which means that both $\Phi_{J,T} \circ \xi$ and $\Phi_{J,\rho} \circ \xi$ are isomorphisms from $GF(q)^n/(\ker(\Phi_{J,\rho}))$ to $GF(q)^t$, where $\xi : GF(q)^n \to GF(q)^n/(\ker(\Phi_{J,\rho}))$ is the canonical map and $\circ$ denotes the composition of maps. Therefore, there must exist an isomorphism ζ such that $\Phi_{J,T} \circ \xi = \zeta \circ \Phi_{J,\rho} \circ \xi$.

If $n < t$, $\ker(\Phi_{J,\rho}) = \{\mathbf{0}\}$. Both $\Phi_{J,T}$ and $\Phi_{J,\rho}$ are injective. So there is an isomorphism ζ satisfying $\Phi_{J,T} = \zeta \circ \Phi_{J,\rho}$.

So in either case, an invertible matrix F^* exists which satisfies

$$(f_0, f_1, \ldots, f_{t-1}) = (f_0^*, f_1^*, \ldots, f_{t-1}^*)F^*.$$

The proof is then complete when we take $F = F^{'}F^*$. □

Theorem 2 tells us that there exists a linear transformation that connects the coefficients of $\Phi_{J,T}(\boldsymbol{\gamma})$ and the high-degree coefficients of $\Phi_{J,q}(\boldsymbol{\gamma})$. Recall that we have shown that $\Phi_{J,q}(\boldsymbol{\gamma})$ is equivalent to $\text{IDFT}_{GF(q)}((\boldsymbol{\gamma}, \mathbf{0}))$ in Lemma 1. So for every $T(x)$, the corresponding GIDFT is related to $\text{IDFT}_{GF(q)}$. This also explains the term *generalized*.

We have discussed several properties of the GIDFT and illustrated the connection between the GIDFT and the IDFT for general J and $T(x)$. In the following, we shall assume that J has an algebraic structure, such as being a coset of some subgroup. An important property will be derived, which is essential to the derivation of the fast generalized inverse discrete Fourier transform.

Lemma 2: If J is a coset of some subgroup of $GF(q)$, say K, and

$$T(x) = \prod_{\beta \in K} (x - \beta),$$

then for $\boldsymbol{\gamma} \in GF(q)^n$, we have

$$\Phi_{J,T}(\boldsymbol{\gamma}) = \text{IDFT}_J(\boldsymbol{\delta} \cdot \boldsymbol{\gamma}),$$

where $\boldsymbol{\delta} = (\delta_0, \delta_1, \ldots, \delta_{n-1}) \in GF(q)^n$ and $\cdot$ represents the component-wise multiplication. In particular, if K is an additive subgroup, then the components δ_i are equal for all possible indices i.

Proof: By definition, we have

$$\Phi_{J,T}(\boldsymbol{\gamma}) = \sum_{i=0}^{n-1} \gamma_i \frac{T(x) - T(\alpha_i)}{x - \alpha_i}$$

and

$$\text{IDFT}_J(\boldsymbol{\gamma}) = \sum_{i=0}^{n-1} \gamma_i \frac{\prod_{j \neq i} (\alpha_i - \alpha_j)^{-1} \prod_{j=0}^{n-1} (x - \alpha_j)}{x - \alpha_i}.$$

We first show that $T(\alpha_i)$ is a constant for all $\alpha_i \in J$. If K is additive, there exists $\upsilon \in GF(q)$ such that $\alpha_i = \upsilon + \beta_i$ for all indices i. Then

$$\begin{aligned} T(\alpha_i) &= \prod_{j=0}^{n-1} (\alpha_i - \beta_j) \\ &= \prod_{j=0}^{n-1} (\upsilon + \beta_i - \beta_j), \end{aligned}$$

which is equal to the multiplication of all elements in J since K is closed under addition. So $T(\alpha_i)$ is a constant. If K is multiplicative, $T(x) = x^n - 1$. There is $\upsilon \in GF(q)$ such that $\alpha_i = \upsilon \beta_i$ for all indices i. Then

$$T(\alpha_i) = (\upsilon \beta_i)^n - 1 = \upsilon^n - 1.$$

So in either case, $T(\alpha_i)$ is a constant for all $\alpha_i \in J$.

As $T(\alpha_i)$ is a constant, the polynomial $T(x) - T(\alpha_i)$ has $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ as its roots. Hence, $T(x) - T(\alpha_i)$ and $\prod_{j=0}^{n-1}(x - \alpha_j)$ have the same set of roots. Since they have the same degree and are both monic, we can conclude that

$$T(x) - T(\alpha_i) = \prod_{j=0}^{n-1} (x - \alpha_j).$$

Let $\delta_i = \prod_{j \neq i} (\alpha_i - \alpha_j)$. It is obvious that

$$\Phi_{J,T}(\boldsymbol{\gamma}) = \text{IDFT}_J(\boldsymbol{\delta} \cdot \boldsymbol{\gamma}).$$

If K is an additive subgroup, for every $\alpha_i \in J$, we have $\alpha_i = \upsilon + \beta_i$ for some $\upsilon \in GF(q)$ and $\beta_i \in K$. It follows that

$$\prod_{j\neq i}(\alpha_i - \alpha_j) = \prod_{j\neq i}(\upsilon + \beta_i - \upsilon - \beta_j) = \prod_{j\neq i}(\beta_i - \beta_j),$$

which is equal to the multiplication of all nonzero elements in K. Hence, we have $\delta_i = \delta_j$ for all possible i, j. The proof is complete. □

Corollary 1: If J is a coset of some subgroup of $GF(q)$, say K, and

$$T(x) = \prod_{\beta\in K}(x - \beta),$$

then we have

$$\mathrm{DFT}_J(\Phi_{J,T}(\boldsymbol{\gamma})) = \boldsymbol{\delta}\cdot\boldsymbol{\gamma}.$$

Proof: The assertion is true since the IDFT_J is the inverse map of the DFT_J. □

We have studied the GIDFT and its properties. Since $\Phi_{J,T}$ is a homomorphism from $GF(q)^n$ to $GF(q)^t$, where $n = |J|$ and $t = \deg(T(x))$, a straightforward computation of $\Phi_{J,T}(\boldsymbol{\gamma})$ costs $O(nt)$ field operations. In the next section, we shall propose a fast method to compute $\Phi_{J,T}(\boldsymbol{\gamma})$, called the fast generalized inverse discrete Fourier transform, whose complexity is $O(q\log(t) + t^2)$, based on the properties presented in this section.

III. Fast Generalized Inverse Discrete Fourier Transform

This section proposes a fast algorithm for computing the GIDFT over J and $T(x)$, say $\Phi_{J,T}(\boldsymbol{\gamma})$. We shall refer to this algorithm as the *fast generalized inverse discrete Fourier transform* (FGIDFT). The FGIDFT has a computational complexity of $O(q\log(t) + t^2)$, where q is the cardinality of the field and $t = \deg(T(x))$.

The FGIDFT is built on fast algorithms of the DFT and IDFT over $GF(q)$, which are usually called the fast Fourier transform (FFT) and inverse fast Fourier transform (IFFT). So a constraint of the FGIDFT is that $GF(q)$ should contain a subgroup of order 2^μ such that $2^\mu \geq t$ (Fortunately, this is the only constraint of the FGIDFT).

If $q = 2^m$ for some positive integer $m \geq \mu$, this constraint is obviously satisfied and the underlying subgroup is additive, which is the most commonly encountered case in practice. In such a case, an additive FFT, also called the *Lin–Chung–Han fast Fourier transform* (LCH-FFT), computes the DFT over the underlying additive subgroup or its coset in log-linear time. Its inverse map, called the *Lin–Chung–Han inverse fast Fourier transform* (LCH-IFFT), also runs in log-linear time. If q is not a power of 2, the constraint will be satisfied if and only if $2^\mu \mid (q-1)$, where the underlying subgroup is multiplicative. In this setting, the FFT and IFFT algorithms that compute the DFT and IDFT over the multiplicative subgroup or its coset also exist, operating in log-linear time. These algorithms are usually referred to as the multiplicative FFT and multiplicative IFFT.

It should be mentioned that the LCH-FFT and LCH-IFFT are built on a specific basis for $GF(q)[x]/(x^q - x)$, called the *Lin–Chung–Han basis* (LCH-basis), say $\mathbb{X} = \{X_0(x), X_1(x), \ldots, X_{q-1}(x)\}$ (We shall discuss the LCH-basis in the next section). Likewise, the multiplicative FFT and IFFT are built on the standard basis of $GF(q)[x]/(x^q - x)$, that is, $\{x^0, x^1, \ldots, x^{q-1}\}$. In subsequent content of this section, reference to an additive subgroup indicates that the LCH-basis is chosen, while reference to a multiplicative subgroup indicates that the standard basis is chosen. This ensures that the FFT and IFFT are available.

Lemma 3: Assume that $GF(q)$ contains a subgroup of order 2^μ such that $2^\mu \geq t$. Let ν be the smallest integer such that $2^\nu \geq t$. Then $GF(q)$ contains a subgroup of order 2^ν. Furthermore, we have $2^\nu = O(t)$.

Proof: By assumption, $\nu \leq \mu$. Denote the subgroup of order 2^μ in $GF(q)$ by K. If K is multiplicative, then K is a cyclic group since any multiplicative subgroup of $GF(q)$ is cyclic. Since 2^ν divides 2^μ, the group K contains an element of order 2^ν. This element generates a cyclic subgroup of K of order 2^ν. If K is additive, we have $q = 2^m$ for some m. Since $GF(2^m)$ is a vector space over $GF(2)$, K can be viewed as a subspace over $GF(2)$ whose dimension is μ. Evidently, one can choose ν elements from a basis of K, and these elements span a vector space over $GF(2)$. This new vector space is a subgroup of $GF(2^m)$ and its order equals 2^ν. In either case, $GF(q)$ contains a subgroup of order 2^ν.

Since ν is the smallest integer satisfying $2^\nu \geq t$, one must have $2^{\nu-1} < t$. So we can conclude that

$$t \leq 2^\nu < 2t,$$

which implies that $2^\nu = O(t)$. □

By Lemma 3, we can assume that, without loss of generality, μ is the smallest integer such that $2^\mu \geq t$. Let K denote the subgroup of order 2^μ contained in $GF(q)$. We assume that K is in a fixed order, that is,

$$K = \{\beta_0, \beta_1, \ldots, \beta_{2^\mu-1}\},$$

which satisfies that the subset $\{\beta_0, \beta_1, \ldots, \beta_{2^\tau-1}\}$ is a subgroup of K for all $0 \leq \tau \leq \mu$, and that for all indices $0 \leq j < 2^\tau$ and $0 \leq l < 2^{\mu-\tau}$, $\beta_{j+l\cdot 2^\tau}$ is equal to $\upsilon_l + \beta_j$ or $\upsilon_l \cdot \beta_j$ for some scalar υ_l, depending on K is additive or multiplicative. Evidently, such an order always exists.

For any τ in the range $0 \leq \tau \leq \mu$ and $0 \leq l < 2^{\mu-\tau}$, let the notation $M_{(l,\tau)}$ denote the subset of K

$$M_{(l,\tau)} = \{\beta_{l\cdot 2^\tau}, \beta_{l\cdot 2^\tau+1}, \ldots, \beta_{l\cdot 2^\tau+2^\tau-1}\}.$$

Clearly, $M_{(0,\tau)}$ is a subgroup of K.

Proposition 1: Given τ in the range $0 \leq \tau \leq \mu$, for every $0 \leq l < 2^{\mu-\tau}$, the set $M_{(l,\tau)}$ is a coset of the subgroup $M_{(0,\tau)}$.

Proof: The claim is obviously true given the assumption about K. □

Let

$$\kappa(x) = \prod_{j=0}^{2^\mu-1}(x - \beta_j) \text{ and } \rho(x) = \prod_{j=0}^{t-1}(x - \beta_j).$$

The procedure of computing $\Phi_{J,T}(\boldsymbol{\gamma})$ can be described by

$$\boldsymbol{\gamma} \to \Phi_{J,\kappa}(\boldsymbol{\gamma}) \to \Phi_{J,\rho}(\boldsymbol{\gamma}) \to \Phi_{J,T}(\boldsymbol{\gamma}).$$

In the following sections, we will discuss these steps in turn.

A. Computation of $\Phi_{J,\kappa}(\boldsymbol{\gamma})$

Recall that we have assumed that $J = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ and that $GF(q) = \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$. Then for every $\boldsymbol{\gamma} \in GF(q)^n$,

$$\begin{aligned}\Phi_{J,\kappa}(\boldsymbol{\gamma}) &= \sum_{i=0}^{n-1} \gamma_i \frac{\kappa(x)-\kappa(\alpha_i)}{x-\alpha_i} \\ &= \sum_{i=0}^{n-1} \gamma_i \frac{\kappa(x)-\kappa(\alpha_i)}{x-\alpha_i} + \sum_{i=n}^{q-1} 0 \frac{\kappa(x)-\kappa(\alpha_i)}{x-\alpha_i} \\ &= \sum_{i=0}^{q-1} \gamma_i \frac{\kappa(x)-\kappa(\alpha_i)}{x-\alpha_i},\end{aligned}$$

where $\gamma_i = 0$ for every $i \geq n$. This gives

$$\Phi_{J,\kappa}(\boldsymbol{\gamma}) = \Phi_{GF(q),\kappa}((\boldsymbol{\gamma}, \mathbf{0})),$$

where $\mathbf{0}$ is a zero vector of length $q-n$. Let the notation $\bar{\boldsymbol{\gamma}}$ denote the vector $(\boldsymbol{\gamma}, \mathbf{0})$.

Given a vector $\bar{\boldsymbol{\gamma}} \in GF(q)^q$ and a subset of $GF(q)$, say $W = \{\upsilon_0, \upsilon_1, \ldots, \upsilon_{|W|-1}\}$, $\bar{\boldsymbol{\gamma}}_W$ denotes the vector

$$\bar{\boldsymbol{\gamma}}_W = (\bar{\gamma}_{j_0}, \bar{\gamma}_{j_1}, \ldots, \bar{\gamma}_{j_{|W|-1}})$$

in which $j_l = i$ if and only if $\upsilon_l = \alpha_i$. Recall that we have assumed that $GF(q) = \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$.

The field $GF(q)$ can be partitioned into cosets of K. More precisely,

$$GF(q) = \begin{cases} \bigcup_W W, & \text{if } K \text{ is additive} \\ (\bigcup_W W) \bigcup \{0\}, & \text{if } K \text{ is multiplicative,} \end{cases}$$

where W are cosets of K. This gives

$$\Phi_{GF(q),\kappa}(\bar{\boldsymbol{\gamma}}) = \sum_W \Phi_{W,\kappa}(\bar{\boldsymbol{\gamma}}_W) + I_K \cdot \Phi_{\{0\},\kappa}(\bar{\boldsymbol{\gamma}}_{\{0\}}), \quad (7)$$

where $I_K = 0$ if K is additive and $I_K = 1$ otherwise. According to Lemma 2, each component $\Phi_{W,\kappa}(\bar{\boldsymbol{\gamma}}_W)$ can be computed by

$$\text{IDFT}_W(\boldsymbol{\delta} \cdot \bar{\boldsymbol{\gamma}}_W),$$

where the vector $\boldsymbol{\delta}$ can be precomputed. By our assumption, there exists an IFFT algorithm to compute the IDFT over every coset W of K within $O(2^\mu \log(2^\mu))$ operations. So the complexity of computing $\Phi_{W,\kappa}(\bar{\boldsymbol{\gamma}}_W)$ is $O(2^\mu \log(2^\mu))$. Since there are $\lfloor \frac{q}{2^\mu} \rfloor$ cosets of K and the complexity of a single summation is $O(2^\mu)$, hence computing the summation $\sum_W \Phi_{W,\kappa}(\bar{\boldsymbol{\gamma}}_W)$ has a complexity of

$$O(\lfloor \frac{q}{2^\mu} \rfloor \cdot 2^\mu \log(2^\mu) + (\lfloor \frac{q}{2^\mu} \rfloor - 1) \cdot 2^\mu) = O(q \log(2^\mu)).$$

Finally, as there is only one element in the set $\{0\}$, computing the component $\Phi_{\{0\},\kappa}(\bar{\boldsymbol{\gamma}}_{\{0\}})$ is straightforward within $O(1)$ operations. Hence, we can conclude that the computation of $\Phi_{J,\kappa}(\boldsymbol{\gamma})$ has complexity $O(q \log(2^\mu))$.

B. Computation of $\Phi_{J,\rho}(\boldsymbol{\gamma})$

We now turn to the computation of $\Phi_{J,\rho}(\boldsymbol{\gamma})$ from $\Phi_{J,\kappa}(\boldsymbol{\gamma})$.

Lemma 4: For every $\boldsymbol{\gamma} \in GF(q)^n$, $\Phi_{J,\rho}(\boldsymbol{\gamma})$ is the quotient of $\Phi_{J,\kappa}(\boldsymbol{\gamma})$ divided by $\prod_{j=t}^{2^\mu-1}(x-\beta_j)$.

Proof: Since $\kappa(x) = \rho(x)\prod_{j=t}^{2^\mu-1}(x-\beta_j)$, for every $\alpha_i \in J$, using a similar method in Theorem 2, one can prove that $\frac{\rho(x)-\rho(\alpha_i)}{x-\alpha_i}$ is the quotient of $\frac{\kappa(x)-\kappa(\alpha_i)}{x-\alpha_i}$ divided by $\prod_{j=t}^{2^\mu-1}(x-\beta_j)$. Then by multiplying the scalars and summing over i, one can obtain the claim. □

Lemma 4 tells us that

$$\Phi_{J,\kappa}(\boldsymbol{\gamma}) = \Phi_{J,\rho}(\boldsymbol{\gamma}) \prod_{j=t}^{2^\mu-1}(x-\beta_j) + \eta(x),$$

where $\deg(\eta(x)) < 2^\mu - t$. Let $f_\kappa(x) = \Phi_{J,\kappa}(\boldsymbol{\gamma})$ and let $f_\rho(x) = \Phi_{J,\rho}(\boldsymbol{\gamma})$. This gives

$$f_\kappa(x) = f_\rho(x) \prod_{j=t}^{2^\mu-1}(x-\beta_j) + \eta(x). \quad (8)$$

So $\eta(\beta_j) = f_\kappa(\beta_j)$ for $j = t, t+1, \ldots, 2^\mu - 1$. Since $\deg(\eta(x)) < 2^\mu - t$, these evaluations determine the $\eta(x)$ uniquely. Once $\eta(x)$ is known, we further have

$$f_\rho(\beta_j) = (f_\kappa(\beta_j) - \eta(\beta_j))(\prod_{l=t}^{2^\mu-1}(\beta_j - \beta_l))^{-1},$$

for $j = 0, 1, \ldots, t-1$. Likewise, since $\deg(f_\rho(x)) < t$, these evaluations determine the $f_\rho(x)$ uniquely too. So computing the $\eta(x)$ and $f_\rho(x)$ can be regarded as doing the IDFT over $\{\beta_t, \beta_{t+1}, \ldots, \beta_{2^\mu-1}\}$ and over $\{\beta_0, \beta_1, \ldots, \beta_{t-1}\}$, respectively. However, unless $t = 2^\mu$, the orders of these two sets are not powers of two, which means that we cannot make use of the FFT and IFFT straightforwardly. Instead, we reformulate the computations of $\eta(x)$ and $f_\rho(x)$ as the GIDFT problems, which are subject to a degree constraint and then solve them.

Computing the $\eta(x)$ and $f_\rho(x)$ can be reformulated as Problem 1 and Problem 2, respectively.

Problem 1: Given a set $M_{(l,\tau)}$ and a vector $(f(\beta_{l\cdot 2^\tau}), f(\beta_{l\cdot 2^\tau+1}), \ldots, f(\beta_{l\cdot 2^\tau+2^\tau-1}))$ in which

$$f(\beta_{l\cdot 2^\tau+\epsilon}), f(\beta_{l\cdot 2^\tau+\epsilon+1}), \ldots, f(\beta_{l\cdot 2^\tau+2^\tau-1})$$

are known, computing the IDFT over $M_{(l,\tau)}$ under the constraint that $\deg(f(x)) < 2^\tau - \epsilon$.

Problem 2: Given a set $M_{(l,\tau)}$ and a vector $(f(\beta_{l\cdot 2^\tau}), f(\beta_{l\cdot 2^\tau+1}), \ldots, f(\beta_{l\cdot 2^\tau+2^\tau-1}))$ in which

$$f(\beta_{l\cdot 2^\tau}), f(\beta_{l\cdot 2^\tau+1}), \ldots, f(\beta_{l\cdot 2^\tau+\epsilon-1})$$

are known, computing the IDFT over $M_{(l,\tau)}$ under the constraint that $\deg(f(x)) < \epsilon$.

Note that these two problems always have unique solutions since they correspond to two systems of linear equations whose matrices are Vandermonde matrices. According to Lemma 2, under a component-wise vector multiplication, the IDFT over $M_{(l,\tau)}$ can be regarded as the GIDFT over $M_{(l,\tau)}$ and $\kappa_\tau(x) = \prod_{j=0}^{2^\tau-1}(x-\beta_j)$. Thus, Problem 1 and 2 can be regarded as Problem 3 and 4, respectively:

Problem 3: Given a set $M_{(l,\tau)}$ and a vector $\hat{\boldsymbol{\gamma}} = (\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{2^\tau-1})$ in which

$$\hat{\gamma}_\epsilon, \hat{\gamma}_{\epsilon+1}, \ldots, \hat{\gamma}_{2^\tau-1}$$

are known, computing $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})$ under the constraint that $\deg(\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})) < 2^\tau - \epsilon$.

Problem 4: Given a set $M_{(l,\tau)}$ and a vector $\hat{\boldsymbol{\gamma}} = (\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{2^\tau-1})$ in which

$$\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{\epsilon-1}$$

are known, computing $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})$ under the constraint that $\deg(\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})) < \epsilon$.

Algorithm 1 and 2 provide solutions for Problem 3 and 4, respectively. Before proving the correctness of them, we need the following lemmas.

Algorithm 1 Algorithm for Solving Problem 3

Input: A set $M_{(l,\tau)}$, and a vector $\hat{\boldsymbol{\gamma}} = (\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{2^\tau-1})$ in which $\hat{\gamma}_\epsilon, \hat{\gamma}_{\epsilon+1}, \ldots, \hat{\gamma}_{2^\tau-1}$ are known.
Output: $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})$ such that $\deg(\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})) < 2^\tau - \epsilon$.

if $\tau = 0$ **then**
  if $\epsilon = 0$ **then**
    $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) = \hat{\gamma}_0$.
  else
    $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) = 0$.
  end if
else
  Let $\boldsymbol{\sigma}_0 = (\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{2^{\tau-1}-1})$ and let $\boldsymbol{\sigma}_1 = (\hat{\gamma}_{2^{\tau-1}}, \hat{\gamma}_{2^{\tau-1}+1}, \ldots, \hat{\gamma}_{2^\tau-1})$.
  if $\epsilon \geq 2^{\tau-1}$ **then**
    Call Algorithm 1 with $M_{(2l+1,\tau-1)}$ and $\boldsymbol{\sigma}_1$ in which $\hat{\gamma}_\epsilon, \hat{\gamma}_{\epsilon+1}, \ldots, \hat{\gamma}_{2^\tau-1}$ are known, and obtain $\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)$.
    Let $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) = (\varrho_{(2l+1,\tau-1)} - \varrho_{(2l,\tau-1)})\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)$.
  else
    Compute $\theta(x) = \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)$.
    Compute $\boldsymbol{\sigma}_2$ such that $\theta(x) = \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_2)$.
    Let $\boldsymbol{\varsigma} = \boldsymbol{\sigma}_0 + \boldsymbol{\sigma}_2$.
    Call Algorithm 1 with $M_{(2l,\tau-1)}$ and $\boldsymbol{\varsigma}$ in which $\varsigma_\epsilon, \varsigma_{\epsilon+1}, \ldots, \varsigma_{2^{\tau-1}-1}$ are known, and obtain $\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})$.
    Let $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) = \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})\kappa_{\tau-1}(x) - \varrho_{(2l+1,\tau-1)}\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma}) + (\varrho_{(2l,\tau-1)} - \varrho_{(2l+1,\tau-1)})\theta(x)$.
  end if
end if
return $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})$.

Algorithm 2 Algorithm for Solving Problem 4

Input: A set $M_{(l,\tau)}$, and a vector $\hat{\boldsymbol{\gamma}} = (\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{2^\tau-1})$ in which $\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{\epsilon-1}$ are known.
Output: $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})$ such that $\deg(\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})) < \epsilon$.

if $\tau = 0$ **then**
  if $\epsilon = 0$ **then**
    $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) = 0$.
  else
    $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) = \hat{\gamma}_0$.
  end if
else
  Let $\boldsymbol{\sigma}_0 = (\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{2^{\tau-1}-1})$ and $\boldsymbol{\sigma}_1 = (\hat{\gamma}_{2^{\tau-1}}, \hat{\gamma}_{2^{\tau-1}+1}, \ldots, \hat{\gamma}_{2^\tau-1})$.
  if $\epsilon \leq 2^{\tau-1}$ **then**
    Call Algorithm 2 with $M_{(2l,\tau-1)}$ and $\boldsymbol{\sigma}_0$ in which $\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{\epsilon-1}$ are known, and obtain $\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0)$.
    Let $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) = (-\varrho_{(2l+1,\tau-1)} + \varrho_{(2l,\tau-1)})\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0)$.
  else
    Compute $\theta(x) = \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0)$.
    Compute $\boldsymbol{\sigma}_2$ such that $\theta(x) = \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_2)$.
    Let $\boldsymbol{\varsigma} = \boldsymbol{\sigma}_1 + \boldsymbol{\sigma}_2$.
    Call Algorithm 2 with $M_{(2l+1,\tau-1)}$ and $\boldsymbol{\varsigma}$ in which $\varsigma_0, \varsigma_1, \ldots, \varsigma_{\epsilon-2^{\tau-1}-1}$ are known, and obtain $\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})$.
    Let $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) = \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})\kappa_{\tau-1}(x) - \varrho_{(2l,\tau-1)}\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma}) + (\varrho_{(2l,\tau-1)} - \varrho_{(2l+1,\tau-1)})\theta(x)$.
  end if
end if
return $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})$.

Lemma 5: For $0 \leq \tau \leq \mu$ and $0 \leq l < 2^{\mu-\tau}$, the evaluations $\kappa_\tau(\beta_{l\cdot 2^\tau+j})$ are equal to each other for all $0 \leq j < 2^\tau$.

Proof: For all $0 \leq j < 2^\tau$, $\beta_{l\cdot 2^\tau+j} \in M_{(l,\tau)}$. If K is additive, by Proposition 1, there is a scalar υ_l such that $\beta_{l\cdot 2^\tau+j} = \beta_j + \upsilon_l$ and $\beta_{l\cdot 2^\tau+u} = \beta_u + \upsilon_l$. Since $\kappa_\tau(x) = \prod_{h=0}^{2^\tau-1}(x - \beta_h)$,

$$\begin{aligned}\kappa_\tau(\beta_{l\cdot 2^\tau+j}) &= \prod_{h=0}^{2^\tau-1}(\beta_{l\cdot 2^\tau+j} - \beta_h)\\ &= \prod_{h=0}^{2^\tau-1}(\upsilon_l + \beta_j - \beta_h)\\ &= \prod_{h'=0}^{2^\tau-1}(\upsilon_l + \beta_u - \beta_{h'})\\ &= \kappa_\tau(\beta_{l\cdot 2^\tau+u}),\end{aligned}$$

where the third equality follows from the fact that the subgroup $\{\beta_0, \beta_1, \ldots, \beta_{2^\tau-1}\}$ is closed under addition. This proves the claim when K is additive.

If K is multiplicative, $\kappa_\tau(x) = x^{2^\tau} - 1$ since every subgroup $M_{(0,\tau)}$ is cyclic. By Proposition 1, there is a scalar υ_l such

that $\beta_{l\cdot 2^\tau+j} = \beta_j \cdot \upsilon_l$ and $\beta_{l\cdot 2^\tau+u} = \beta_u \cdot \upsilon_l$. It follows that

$$\begin{aligned}\kappa_\tau(\beta_{l\cdot 2^\tau+j}) &= (\beta_j \cdot \upsilon_l)^{2^\tau} - 1\\ &= \beta_j^{2^\tau} \cdot \upsilon_l^{2^\tau} - 1\\ &= \beta_u^{2^\tau} \cdot \upsilon_l^{2^\tau} - 1\\ &= \kappa_\tau(\beta_{l\cdot 2^\tau+u}),\end{aligned}$$

where the third equality holds since $\beta_j^{2^\tau} = \beta_u^{2^\tau} = 1$. The proof is then complete. □

Lemma 5 tells us that the evaluations of $\kappa_\tau(x)$ at the elements of $M_{(l,\tau)}$ are equal to each other. For simplicity, we denote this value by $\varrho_{(l,\tau)}$, that is,

$$\varrho_{(l,\tau)} = \kappa_\tau(\beta_{l\cdot 2^\tau}).$$

Lemma 6: For $0 < \tau \le \mu$ and $0 \le l < 2^{\mu-\tau}$,

$$\kappa_\tau(x) - \varrho_{(l,\tau)} = (\kappa_{\tau-1}(x) - \varrho_{(2l,\tau-1)})(\kappa_{\tau-1}(x) - \varrho_{(2l+1,\tau-1)}). \tag{9}$$

Proof: According to Lemma 5, we have

$$\kappa_\tau(\beta_{l\cdot 2^\tau+j}) - \varrho_{(l,\tau)} = 0$$

for all indices $0 \le j < 2^\tau$. On the other hand, if $0 \le j < 2^{\tau-1}$,

$$\begin{aligned}\kappa_{\tau-1}(\beta_{l\cdot 2^\tau+j}) - \varrho_{(2l,\tau-1)} &= \kappa_{\tau-1}(\beta_{2l\cdot 2^{\tau-1}+j}) - \varrho_{(2l,\tau-1)}\\ &= 0\end{aligned}$$

by Lemma 5. Likewise, if $2^{\tau-1} \le j < 2^\tau$,

$$\begin{aligned}&\kappa_{\tau-1}(\beta_{l\cdot 2^\tau+j}) - \varrho_{(2l+1,\tau-1)}\\ &= \kappa_{\tau-1}(\beta_{(2l+1)\cdot 2^{\tau-1}+(j-2^{\tau-1})}) - \varrho_{(2l,\tau-1)}\\ &= 0.\end{aligned}$$

So the monic polynomials on both sides of (9) have the same evaluations at 2^τ elements of $M_{l,\tau}$. Moreover, since the degrees of these two polynomials are both equal to 2^τ, we can conclude that they are the same. This proves the claim. □

Theorem 3: Algorithm 1 is correct and its complexity is $O(2^\tau \log(2^\tau))$.

Proof: We prove the assertion by induction on l and τ.

Assume that $\tau = 0$. The vector $\hat{\boldsymbol{\gamma}} = (\hat{\gamma}_0)$. If $\epsilon = 0$, $\kappa_0(x) = x - \beta_0$. It follows that

$$\begin{aligned}\Phi_{M_{(l,0)},\kappa_0}(\hat{\boldsymbol{\gamma}}) &= \hat{\gamma}_0 \frac{x - \beta_0 - \beta_l + \beta_0}{x - \beta_l}\\ &= \hat{\gamma}_0.\end{aligned}$$

If $\epsilon > 0$, $2^\tau - \epsilon \le 0$. The only solution is the zero polynomial. In either case, Algorithm 1 is correct.

Now suppose that Algorithm 1 is correct for $0, 1, \ldots, \tau - 1$ and every possible l. Let $\boldsymbol{\sigma}_0 = (\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{2^{\tau-1}-1})$ and $\boldsymbol{\sigma}_1 = (\hat{\gamma}_{2^{\tau-1}}, \hat{\gamma}_{2^{\tau-1}+1}, \ldots, \hat{\gamma}_{2^\tau-1})$. We have

$$\begin{aligned}&\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})\\ &= \sum_{j=0}^{2^\tau-1} \hat{\gamma}_j \frac{\kappa_\tau(x) - \kappa_\tau(\beta_{l\cdot 2^\tau+j})}{x - \beta_{l\cdot 2^\tau+j}}\\ &= \sum_{j=0}^{2^\tau-1} \hat{\gamma}_j \frac{\kappa_\tau(x) - \varrho_{(l,\tau)}}{x - \beta_{l\cdot 2^\tau+j}} \quad \textit{(by Lemma 5)}\\ &= \sum_{j=0}^{2^\tau-1} \hat{\gamma}_j \frac{(\kappa_{\tau-1}(x) - \varrho_{(2l,\tau-1)})(\kappa_{\tau-1}(x) - \varrho_{(2l+1,\tau-1)})}{x - \beta_{l\cdot 2^\tau+j}}\\ &\qquad\qquad \textit{(by Lemma 6)}\\ &= \sum_{j=0}^{2^{\tau-1}-1} \hat{\gamma}_j \frac{\kappa_{\tau-1}(x) - \varrho_{(2l,\tau-1)}}{x - \beta_{l\cdot 2^\tau+j}}(\kappa_{\tau-1}(x) - \varrho_{(2l+1,\tau-1)})\\ &\quad + \sum_{j=0}^{2^{\tau-1}-1} \hat{\gamma}_{j+2^{\tau-1}} \frac{\kappa_{\tau-1}(x) - \varrho_{(2l+1,\tau-1)}}{x - \beta_{l\cdot 2^\tau+2^{\tau-1}+j}}\\ &\quad \cdot (\kappa_{\tau-1}(x) - \varrho_{(2l,\tau-1)})\\ &= \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0)(\kappa_{\tau-1}(x) - \varrho_{(2l+1,\tau-1)})\\ &\quad + \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)(\kappa_{\tau-1}(x) - \varrho_{(2l,\tau-1)})\\ &= (\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) + \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1))\kappa_{\tau-1}(x)\\ &\quad - \varrho_{(2l+1,\tau-1)}\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0)\\ &\quad - \varrho_{(2l,\tau-1)}\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)\end{aligned} \tag{10}$$

According to the definitions, the degrees of $\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0)$ and $\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)$ are less than $2^{\tau-1}$, and $\deg(\kappa_{\tau-1}(x)) = 2^{\tau-1}$. So the high-degree coefficients of $\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})$ are totally determined by

$$(\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) + \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1))\kappa_{\tau-1}(x).$$

If $\epsilon \ge 2^{\tau-1}$, $\deg(\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})) < 2^\tau - \epsilon \le 2^{\tau-1}$. We must have

$$\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) + \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1) = 0.$$

It follows that

$$\begin{aligned}&\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})\\ &= (\varrho_{(2l+1,\tau-1)} - \varrho_{(2l,\tau-1)})\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1).\end{aligned}$$

Since $\epsilon \ge 2^{\tau-1}$, $\hat{\gamma}_\epsilon, \hat{\gamma}_{\epsilon+1}, \ldots, \hat{\gamma}_{2^\tau-1}$ are known in $\boldsymbol{\sigma}_1$. By induction, calling Algorithm 1 with $M_{(2l+1,\tau-1)}$ and $\boldsymbol{\sigma}_1$ gives $\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)$ whose degree is less than $2^\tau - \epsilon$. This implies that $\deg(\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})) < 2^\tau - \epsilon$, which satisfies the degree constraint. This proves the case when $\epsilon \ge 2^{\tau-1}$.

If $\epsilon < 2^{\tau-1}$, the vector $\boldsymbol{\sigma}_1$ is known. Let $\theta(x) = \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)$. By Theorem 1, $\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}$ is an isomorphism. So there exists $\boldsymbol{\sigma}_2$ such that

$$\theta(x) = \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_2).$$

Then by (10), we have

$$\begin{aligned}
&\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) \\
&= (\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) + \Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1))\kappa_{\tau-1}(x) \\
&\quad - \varrho_{(2l+1,\tau-1)}\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) \\
&\quad - \varrho_{(2l,\tau-1)}\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1) \\
&= (\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) + \theta(x))\kappa_{\tau-1}(x) \\
&\quad - \varrho_{(2l+1,\tau-1)}\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) - \varrho_{(2l,\tau-1)}\theta(x) \\
&= (\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) + \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_2))\kappa_{\tau-1}(x) \\
&\quad - \varrho_{(2l+1,\tau-1)}\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) - \varrho_{(2l,\tau-1)}\theta(x) \\
&= (\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0 + \boldsymbol{\sigma}_2))\kappa_{\tau-1}(x) \\
&\quad - \varrho_{(2l+1,\tau-1)}\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) - \varrho_{(2l,\tau-1)}\theta(x).
\end{aligned}$$

Let $\boldsymbol{\varsigma} = \boldsymbol{\sigma}_0 + \boldsymbol{\sigma}_2$. Since the components $\hat{\gamma}_\epsilon, \hat{\gamma}_{\epsilon+1}, \ldots, \hat{\gamma}_{2^{\tau-1}-1}$ of $\boldsymbol{\sigma}_0$ are known and the vector $\boldsymbol{\sigma}_2$ is known, the components $\varsigma_\epsilon, \varsigma_{\epsilon+1}, \ldots, \varsigma_{2^{\tau-1}-1}$ of $\boldsymbol{\varsigma}$ are known. Then by induction, calling Algorithm 1 with $M_{(2l,\tau-1)}$ and $\boldsymbol{\varsigma}$ returns $\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})$ whose degree is less than $2^{\tau-1}-\epsilon$. Because

$$\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma}) = \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0 + \boldsymbol{\sigma}_2),$$

it follows that

$$\begin{aligned}
&\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_0) \\
&= \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma}) - \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1) \\
&= \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma}) - \theta(x).
\end{aligned}$$

Hence, we can obtain that

$$\begin{aligned}
&\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}}) \\
&= \Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})\kappa_{\tau-1}(x) \\
&\quad - \varrho_{(2l+1,\tau-1)}\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma}) \\
&\quad + (\varrho_{(2l+1,\tau-1)} - \varrho_{(2l,\tau-1)})\theta(x).
\end{aligned}$$

Obviously, since $\deg(\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})) < 2^{\tau-1} - \epsilon$, $\deg(\Phi_{M_{(l,\tau)},\kappa_\tau}(\hat{\boldsymbol{\gamma}})) < 2^\tau - \epsilon$. This proves that Algorithm 1 is correct for the case $\epsilon < 2^{\tau-1}$.

It remains to analyze the computational complexity. Clearly, if $\tau = 0$, the complexity is $O(1)$. Suppose that $\tau > 0$. The complexity of Step 11 in Algorithm 1 is $O(2^\tau)$ since it is a scalar multiplication. The complexity of Step 13 is $O(2^\tau \log(2^\tau))$ since it can be done by executing a scalar multiplication and an IFFT, according to Lemma 2. Likewise, Step 14 has complexity $O(2^\tau \log(2^\tau))$ by Corollary 1. Step 15 is a vector addition whose complexity is $O(2^\tau)$. Steps 10 and 16 are recursively calls of themselves with input scale $\tau - 1$. Now the only thing left is to determine the complexity of Step 17. Recall that

$$\kappa_{\tau-1}(x) = \prod_{j=0}^{2^{\tau-1}-1}(x - \beta_j),$$

whose zeros are exactly the elements in the subgroup $M_{(0,\tau-1)}$. If $M_{(0,\tau-1)}$ is additive, then the LCH-basis

$$\{X_0(x), X_1(x), \ldots, X_{q-1}(x)\}$$

is chosen. Let

$$\begin{aligned}
&\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma}) \\
&= h_{2^{\tau-1}-1}X_{2^{\tau-1}-1}(x) + h_{2^{\tau-1}-2}X_{2^{\tau-1}-2}(x) \\
&\quad + \cdots + h_0X_0(x).
\end{aligned}$$

Since $\kappa_{\tau-1}(x) = X_{2^{\tau-1}}(x)$ and $X_j(x) \cdot X_{2^{\tau-1}}(x) = X_{2^{\tau-1}+j}(x)$ if $j < 2^{\tau-1}$ (These properties can be seen in [11]), we have

$$\begin{aligned}
&\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})\kappa_{\tau-1}(x) \\
&= h_{2^{\tau-1}-1}X_{2^\tau-1}(x) + h_{2^{\tau-1}-2}X_{2^\tau-2}(x) \\
&\quad + \cdots + h_0X_{2^{\tau-1}}(x).
\end{aligned}$$

If $M_{(0,\tau-1)}$ is multiplicative, then the standard basis $\{x^0, x^1, \ldots, x^{q-1}\}$ is chosen. We have $\kappa_{\tau-1}(x) = x^{2^{\tau-1}} - 1$ since $M_{(0,\tau-1)}$ is cyclic. Let

$$\begin{aligned}
&\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma}) \\
&= h_{2^{\tau-1}-1}x^{2^{\tau-1}-1} + h_{2^{\tau-1}-2}x^{2^{\tau-1}-2} + \cdots + h_0.
\end{aligned}$$

It follows that

$$\begin{aligned}
&\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})\kappa_{\tau-1}(x) \\
&= h_{2^{\tau-1}-1}x^{2^\tau-1} + h_{2^{\tau-1}-2}x^{2^\tau-2} + \cdots + h_0x^{2^{\tau-1}} \\
&\quad - h_{2^{\tau-1}-1}x^{2^{\tau-1}-1} - h_{2^{\tau-1}-2}x^{2^{\tau-1}-2} - \cdots - h_0.
\end{aligned}$$

In either case, given the coordinate vector of the polynomial $\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})$ with respect to the corresponding basis, the coordinate vector of

$$\Phi_{M_{(2l,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\varsigma})\kappa_{\tau-1}(x),$$

which is the first term on the right side in Step 17, can be obtained by at most 1 vector addition (When analyzing computational complexity, we treat field subtraction as field addition since they have the same complexity in implementation). In addition, the second and third terms on the right side of Step 17 are scalar multiplications. Finally, two vector additions are needed to combine these three terms. Hence, the complexity of Step 17 is $O(2^\tau)$.

Hence, if $\epsilon \geq 2^{\tau-1}$, the operations needed by Algorithm 1 are

$$P(\tau) = P(\tau - 1) + O(2^\tau).$$

If $\epsilon < 2^{\tau-1}$,

$$P(\tau) = O(2^\tau \log(2^\tau)) + P(\tau - 1) + O(2^\tau).$$

In both cases,

$$P(\tau) = P(\tau - 1) + O(2^\tau \log(2^\tau)).$$

Hence, the complexity of Algorithm 1 is $O(2^\tau \log(2^\tau))$. This completes the proof. □

Theorem 4: Algorithm 2 is correct and its complexity is $O(2^\tau \log(2^\tau))$.

Proof: The proof is similar to that of Theorem 3. We omit it here. □

With the help of Algorithms 1 and 2, we can obtain $\eta(x)$ first and then compute $f_\rho(x) = \Phi_{J,\rho}(\boldsymbol{\gamma})$.

Remarks: In the above disucssion, we transform Problems 1 and 2 to Problems 3 and 4, and then solve them by the

GIDFT. Since Problems 1 and 2 can be viewed as the IDFT with inputs

$$(f(\beta_{l\cdot 2^\tau+\epsilon}), f(\beta_{l\cdot 2^\tau+\epsilon+1}), \ldots, f(\beta_{l\cdot 2^\tau+2^\tau-1}))$$

and

$$(f(\beta_{l\cdot 2^\tau}), f(\beta_{l\cdot 2^\tau+1}), \ldots, f(\beta_{l\cdot 2^\tau+\epsilon-1})),$$

respectively, another method for solving them can also be derived. We briefly discuss this strategy for Problem 1 and compare it with Algorithms 1 (The discussion is analogous for Problem 2).

Let

$$M_0 = \{\beta_{l\cdot 2^\tau+\epsilon}, \beta_{l\cdot 2^\tau+\epsilon+1}, \ldots, \beta_{l\cdot 2^\tau+2^\tau-1}\},$$
$$M_1 = \{\beta_{l\cdot 2^\tau}, \beta_{l\cdot 2^\tau+1}, \ldots, \beta_{l\cdot 2^\tau+\epsilon-1}\}.$$

It follows that $M_0 \cap M_1 = \emptyset$ and $M_0 \cup M_1 = M_{(l,\tau)}$. Define the polynomial $z_{M_1}(x)$ with respect to M_1 by

$$z_{M_1}(x) = \prod_{j=0}^{\epsilon-1}(x - \beta_{l\cdot 2^\tau+j}).$$

Assume that another coset $M_{(l',\tau)}$ exists which is different from $M_{(l,\tau)}$. Then Algorithm 3 presents a method which also solves Problem 1. Note that if the cosets $M_{(l,\tau)}$, $M_{(l',\tau)}$ and the parameter ϵ are fixed, then the polynomial $z_{M_1}(x)$, as well as the evaluations $z_{M_1}(\beta_{l\cdot 2^\tau+j})$ and the inverse $z_{M_1}(\beta_{l'\cdot 2^\tau+j})^{-1}$, can be computed in advance.

Algorithm 3 Algorithm for Solving Problem 1 with the Aid of $z_{M_1}(x)$

Input: M_0, evaluations $f(\beta_{l\cdot 2^\tau+j}), z_{M_1}(\beta_{l\cdot 2^\tau+j}), j = \epsilon, \epsilon+1, \ldots, 2^\tau-1$ and $z_{M_1}(\beta_{l'\cdot 2^\tau+j})^{-1}, j = 0, 1, \ldots, 2^\tau-1$.
Output: $f(x) = \mathrm{IDFT}_{M_0}((f(\beta_{l\cdot 2^\tau+\epsilon}), f(\beta_{l\cdot 2^\tau+\epsilon+1}), \ldots, f(\beta_{l\cdot 2^\tau+2^\tau-1})))$.
1: Construct the vector $\hat{\boldsymbol{\gamma}}$ in which $\hat{\gamma}_j = 0$ if $0 \le j < \epsilon$ and $\hat{\gamma}_j = f(\beta_{l\cdot 2^\tau+j}) \cdot z_{M_1}(\beta_{l\cdot 2^\tau+j})$ if $\epsilon \le j < 2^\tau$.
2: Compute $\theta(x) = \mathrm{IDFT}_{M_{(l,\tau)}}(\hat{\boldsymbol{\gamma}})$.
3: Evaluate $\theta(x)$ at the coset $M_{(l',\tau)}$.
4: Compute the vector $\boldsymbol{\sigma}$ in which $\sigma_j = \theta(\beta_{l'\cdot 2^\tau+j}) \cdot z_{M_1}(\beta_{l'\cdot 2^\tau+j})^{-1}$ for $j = 0, 1, \ldots, 2^\tau-1$.
5: Compute $f(x) = \mathrm{IDFT}_{M_{(l',\tau)}}(\boldsymbol{\sigma})$.
6: **return** $f(x)$.

The computational complexity of Algorithm 3 is $O(2^\tau \log(2^\tau))$, which has the same order with that of Algorithms 1. However, Algorithm 1 costs fewer field operations compared with Algorithm 3 since it exploits the relationship between the GIDFT over coset and over sub-coset. As shown in (10), the high-degree part of the GIDFT over the coset $M_{(l,\tau)}$ is determined by the GIDFT over two sub-cosets $M_{(2l,\tau-1)}$ and $M_{(2l+1,\tau-1)}$. So once we have obtained $\Phi_{M_{(2l+1,\tau-1)},\kappa_{\tau-1}}(\boldsymbol{\sigma}_1)$, the original problem can be reduced to a sub-problem, as we shown in the proof of Theorem 3. Therefore, a complexity reduction can be achieved compared with Algorithm 3. The following example further supports this conclusion.

Example 1: The prime field $GF(41)$ contains a multiplicative subgroup of order 8 since $8 \mid (41-1)$, which can be written as

$$K = \{\beta_0, \beta_1, \ldots, \beta_7\} = \{1, 40, 32, 9, 27, 14, 3, 38\}.$$

Thus, 8-point FFT and IFFT are available in $GF(41)$. Given the evaluations $f(\beta_2), f(\beta_3), \ldots, f(\beta_7)$, one can compute $f(x)$ by either Algorithm 1 or 3. In this setting, simulation results show that Algorithm 1 requires 33 field additions and 50 field multiplications, while Algorithm 3 needs 72 field additions and 107 field multiplications (As we discussed before, we regard field subtraction and field addition as the same operation). □

C. Computation of $\Phi_{J,T}(\boldsymbol{\gamma})$

Finally, we compute $\Phi_{J,T}(\boldsymbol{\gamma})$ from $\Phi_{J,\rho}(\boldsymbol{\gamma})$. Theorem 2 tells us that both $\Phi_{J,T}(\boldsymbol{\gamma})$ and $\Phi_{J,\rho}(\boldsymbol{\gamma})$ are related to the high-degree coefficients of $\Phi_{J,q}(\boldsymbol{\gamma})$. Since $\deg(T(x)) = \deg(\rho(x)) = t$, we can conclude that there exists a $t \times t$ invertible matrix which connects the coefficients of $\Phi_{J,T}(\boldsymbol{\gamma})$ and $\Phi_{J,\rho}(\boldsymbol{\gamma})$. Note that if $T(x)$ is fixed, this matrix can be precomputed. Hence, one can obtain $\Phi_{J,T}(\boldsymbol{\gamma})$ from $\Phi_{J,\rho}(\boldsymbol{\gamma})$ through a linear transformation whose complexity is at most $O(t^2)$.

A complete description of computing $\Phi_{J,T}(\boldsymbol{\gamma})$ is shown in Algorithm 4.

Algorithm 4 Fast Generalized Inverse Discrete Fourier Transform (FGIDFT)

Input: $J \subseteq GF(q)$ with cardinality n, $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}) \in GF(q)^n$ and $T(x) \in GF(q)[x]$ of degree t.
Output: $\Phi_{J,T}(\boldsymbol{\gamma})$.
1: Let $\bar{\boldsymbol{\gamma}} = (\boldsymbol{\gamma}, \mathbf{0})$.
2: Compute $f_\kappa(x) = \Phi_{GF(q),\kappa}(\bar{\boldsymbol{\gamma}})$ by (7).
3: Evaluate $f_\kappa(x)$ at the set K by FFT.
4: Given the evaluations $\eta(\beta_j) = f_\kappa(\beta_j)$ for $j = t, t+1, \ldots, 2^\mu-1$, transform Problem 1 to Problem 3 by a component-wise multiplication, and then solve Problem 3 to obtain $\eta(x)$.
5: Compute the evaluations $f_\rho(\beta_j)$ by (8) for $j = 0, 1, \ldots, t-1$.
6: Given the evaluations $f_\rho(\beta_j)$ for $j = 0, 1, \ldots, t-1$, transform Problem 2 to Problem 4 by a component-wise multiplication, and then solve Problem 4 to obtain $f_\rho(x)$.
7: Apply a linear transformation to $f_\rho(x)$ to get $\Phi_{J,T}(\boldsymbol{\gamma})$.
8: **return** $\Phi_{J,T}(\boldsymbol{\gamma})$.

Theorem 5: Algorithm 4 computes $\Phi_{J,T}(\boldsymbol{\gamma})$ and its complexity is at most $O(q\log(t) + t^2)$.

Proof: The correctness of Algorithm 4 can be verified from the preceding discussion in this section. We have shown that the complexity of Step 2 is $O(q\log(2^\mu))$, and the complexity of both Step 4 and Step 6 are $O(2^\mu \log(2^\mu))$. Moreover, it is easy to see that Steps 3 and 5 have complexity $O(2^\mu \log(2^\mu))$. Finally, the complexity of Step 1 is $O(1)$, and the complexity

of Step 7 is at most $O(t^2)$. Hence, the complexity of Algorithm 4 is at most

$$\begin{aligned}&O(q\log(2^\mu))+O(2^\mu\log(2^\mu))+O(2^\mu)+O(1)+O(t^2)\\&=O(q\log(2^\mu)+t^2).\end{aligned}$$

Recall that we have assumed that μ is the smallest integer such that $2^\mu \geq t$. Then by Lemma 3, $2^\mu = O(t)$. Hence, the total complexity of Algorithm 4 is $O(q\log(t)+t^2)$. □

We now provide an example to illustrate the FGIDFT.

Example 2: Let the underlying field be $GF(41)$. 8-point FFT and IFFT are available in this field. We can write

$$\begin{aligned}GF(41)=\{&1,40,32,9,27,14,3,38,2,39,23,18,13,28,\\&6,35,4,37,5,36,26,15,12,29,7,34,19,22,\\&25,16,21,20,8,33,10,31,11,30,24,17,0\}.\end{aligned}$$

The multiplicative subgroup of order 8 is

$$K=\{\beta_0,\beta_1,\ldots,\beta_7\}=\{1,40,32,9,27,14,3,38\}.$$

Let

$$\begin{aligned}J=\{&1,40,32,9,27,14,3,38,2,39,23,18,13,28,\\&6,35,4,37,5,36,26,15,12,29,7,34,19,22,\\&25,16,21,20,8,33,10,31,11,30,24,17\}\end{aligned}$$

and let

$$T(x)=x^6.$$

We compute the GIDFT over J and $T(x)$ by the FGIDFT.

Given the vector

$$\begin{aligned}\boldsymbol{\gamma}=(&12,12,35,38,34,11,38,4,35,11,24,7,26,\\&39,18,11,18,38,13,1,18,21,5,36,4,36,\\&22,11,11,38,40,38,26,33,14,4,17,27,0,36),\end{aligned}$$

we first compute $f_\kappa(x)=\Phi_{GF(q),\kappa}(\bar{\boldsymbol{\gamma}})$ by (7), where $\bar{\boldsymbol{\gamma}}=(\boldsymbol{\gamma},0)$, and we get

$$f_\kappa(x)=x^7+39x^6+13x^5+14x^4+7x^3+6x^2+34x+40.$$

Next, we evaluate $f_\kappa(x)$ at K by the FFT and yield

$$(f(\beta_0),f(\beta_1),\ldots,f(\beta_7))=(31,3,27,32,6,25,36,37).$$

By (8), we have

$$\eta(\beta_6)=36,\eta(\beta_7)=37.$$

Calling Algorithm 1 with inputs $M_{(0,3)}=K$ and $\hat{\boldsymbol{\gamma}}=(\hat{\gamma}_0,\hat{\gamma}_1,\ldots,\hat{\gamma}_7)$ in which $\hat{\gamma}_6=36,\hat{\gamma}_7=37$ are known, we then obtain

$$\eta(x)=34x+16.$$

Evaluating $\eta(x)$ at K by the FFT and computing the evaluations $f_\rho(\beta_0),f_\rho(\beta_1),\ldots,f_\rho(\beta_5)$ by (8), one has

$$(f_\rho(\beta_0),f_\rho(\beta_1),\ldots,f_\rho(\beta_5))=(28,23,38,29,6,10).$$

We call Algorithm 2 with inputs $M_{(0,3)}=K$ and $\hat{\boldsymbol{\gamma}}=(\hat{\gamma}_0,\hat{\gamma}_1,\ldots,\hat{\gamma}_7)$ in which $\hat{\gamma}_0=28,\hat{\gamma}_1=23,\ldots,\hat{\gamma}_5=10$ are known, and obtain

$$f_\rho(x)=x^5+39x^4+22x^3+37x^2+11.$$

Finally, the matrix that connects the coefficients of $f_\rho(x)$ and $\Phi_{J,T}(\boldsymbol{\gamma})$ is

$$\begin{pmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\32&0&1&0&0&0\\0&32&0&1&0&0\\0&0&32&0&1&0\\0&0&0&32&0&1\end{pmatrix}.$$

Multiplying the coordinate vector $(11,0,37,22,39,1)$ of $f_\rho(x)$ on the right by the above matrix, we can obtain the coordinate vector of $\Phi_{J,T}(\boldsymbol{\gamma})$, which is $(6,7,14,13,39,1)$. This yields

$$\Phi_{J,T}(\boldsymbol{\gamma})=x^6+39x^5+13x^4+14x^3+7x+6.$$

□

IV. Additive Fast Generalized Inverse Discrete Fourier Transform

This section focuses on the additive FGIDFT over $GF(2^m)$, since binary extension fields $GF(2^m)$ are commonly used in practice. In Section VI, the additive FGIDFT shall be used to decode binary Goppa codes. We first briefly review the concepts of the LCH-basis, LCH-FFT, and LCH-IFFT. Then we derive the additive FGIDFT.

A. LCH-Basis, LCH-FFT and LCH-IFFT

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

$$\omega_j=j_0v_0+j_1v_1+\cdots+j_{m-1}v_{m-1},0\leq j<2^m, \quad (11)$$

where $(j_0,j_1,\ldots,j_{m-1})$ denotes the binary representation of the integer j. For any integer $0\leq\tau\leq m$, there is a corresponding subspace of $GF(2^m)$, denoted by V_τ, which is spanned by the set $\{v_0,v_1,\ldots,v_{\tau-1}\}$ (V_0 is defined as the zero space). For each subspace V_τ, the corresponding subspace polynomial $s_\tau(x)$ is defined by

$$s_\tau(x)=\prod_{j=0}^{2^\tau-1}(x-\omega_j),0\leq\tau\leq m.$$

Clearly, the subspace polynomial $s_\tau(x)$ is monic and it satisfies that

$$s_\tau(\omega_j)=0$$

for all $\omega_j\in V_\tau$. The subspace polynomial $s_\tau(x)$ is a linearized polynomial, which implies that

$$s_\tau(\omega_j+\omega_l)=s_\tau(\omega_j)+s_\tau(\omega_l)$$

for all $\omega_j,\omega_l\in GF(2^m)$. More detailed discussions can be found in [12].

For any $0\leq j<2^m$, construct a polynomial

$$X_j(x)=s_0(x)^{j_0}s_1(x)^{j_1}\cdots s_{m-1}(x)^{j_{m-1}}.$$

So $X_j(x)$ is monic and $\deg(X_j(x))=j$. It follows that the set $\mathbb{X}=\{X_0(x),X_1(x),\ldots,X_{2^m-1}(x)\}$ forms a basis of vector space $GF(2^m)[x]/(x^{2^m}-x)$ over $GF(2^m)$. Let

$$\bar{X}_j(x)=X_j(x)/p_j,0\leq j<2^m,$$

where $p_j = s_0(v_0)^{j_0} s_1(v_1)^{j_1} \cdots s_{m-1}(v_{m-1})^{j_{m-1}}$. It is easy to check that the set $\bar{\mathbb{X}} = \{\bar{X}_0(x), \bar{X}_1(x), \ldots, \bar{X}_{2^m-1}(x)\}$ is also a basis of $GF(2^m)[x]/(x^{2^m} - x)$ over $GF(2^m)$. As both $\mathbb{X}$ and $\bar{\mathbb{X}}$ are bases, any polynomial $f(x) \in GF(2^m)[x]/(x^{2^m} - x)$ can be represented with respect to $\mathbb{X}$ or $\bar{\mathbb{X}}$. The sets $\mathbb{X}$ and $\bar{\mathbb{X}}$ are called the Lin–Chung–Han basis (LCH-basis). Throughout the paper, we shall use them interchangeably.

For a polynomial $f(x) \in GF(2^m)[x]/(x^{2^m} - x)$ of degree less than 2^τ, given its coordinate vector $\bar{\mathbf{f}} = (\bar{f}_0, \bar{f}_1, \ldots, \bar{f}_{2^\tau-1})$ with respect to $\bar{\mathbb{X}}$ and a scalar $\varepsilon \in GF(2^m)$, the Lin–Chung–Han fast Fourier transform (LCH-FFT) computes the vector

$$\mathbf{F} = (f(\omega_0 + \varepsilon), f(\omega_1 + \varepsilon), \ldots, f(\omega_{2^\tau-1} + \varepsilon))$$

within $O(2^\tau \log(2^\tau))$ field operations, which is denoted by

$$\mathbf{F} = \mathrm{FFT}_{\bar{\mathbb{X}}}(\bar{\mathbf{f}}, \tau, \varepsilon).$$

Its inverse transform, called the Lin–Chung–Han inverse fast Fourier transform (LCH-IFFT), is written as

$$\bar{\mathbf{f}} = \mathrm{IFFT}_{\bar{\mathbb{X}}}(\mathbf{F}, \tau, \varepsilon),$$

whose complexity is also $O(2^\tau \log(2^\tau))$. For completeness, detailed descriptions of $\mathrm{FFT}_{\bar{\mathbb{X}}}$ and $\mathrm{IFFT}_{\bar{\mathbb{X}}}$ are shown in Algorithms 5 and 6, respectively. For more discussions, please refer to [4].

Algorithm 5 $\mathrm{FFT}_{\bar{\mathbb{X}}}$ [13]

Input: $\bar{\mathbf{f}} = (\bar{f}_0, \bar{f}_1, \ldots, \bar{f}_{2^\tau-1})$, τ, ε.
Output: $(f(\omega_0 + \varepsilon), f(\omega_1 + \varepsilon), \ldots, f(\omega_{2^\tau-1} + \varepsilon))$.

if $\tau = 0$ **then**
  return $\bar{f}_0$
end if
for $l = 0, 1, \ldots, 2^{\tau-1} - 1$ **do**
  $a_l^{(0)} = \bar{f}_l + \dfrac{s_{\tau-1}(\varepsilon)}{s_{\tau-1}(v_{\tau-1})} \bar{f}_{l+2^{\tau-1}}$
  $a_l^{(1)} = a_l^{(0)} + \bar{f}_{l+2^{\tau-1}}$
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 $\mathbf{A}_0 = \mathrm{FFT}_{\bar{\mathbb{X}}}(\mathbf{a}^{(0)}, \tau - 1, \varepsilon)$, $\mathbf{A}_1 = \mathrm{FFT}_{\bar{\mathbb{X}}}(\mathbf{a}^{(1)}, \tau - 1, v_{\tau-1} + \varepsilon)$
return $(\mathbf{A}_0, \mathbf{A}_1)$

Algorithm 6 $\mathrm{IFFT}_{\bar{\mathbb{X}}}$ [13]

Input: $\mathbf{F} = (f(\omega_0 + \varepsilon), f(\omega_1 + \varepsilon), \ldots, f(\omega_{2^\tau-1} + \varepsilon)), \tau, \varepsilon$
Output: $\bar{\mathbf{f}}$ such that $\mathbf{F} = \mathrm{FFT}_{\bar{\mathbb{X}}}(\bar{\mathbf{f}}, \tau, \varepsilon)$

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

B. Additive Fast Generalized Inverse Discrete Fourier Transform

Recall that our goal is computing $\Phi_{J,T}(\boldsymbol{\gamma})$, where J is a subset of $GF(2^m)$ with cardinality n, $T(x) \in GF(2^m)[x]$ and $\boldsymbol{\gamma} \in GF(2^m)^n$. Let $t = \deg(T(x))$ and let μ be the smallest integer such that $2^\mu \geq t$.

Lemma 7: The subspace

$$V_\mu = \{\omega_0, \omega_1, \ldots, \omega_{2^\mu-1}\}$$

is the desired subgroup K that defined in Section III. Furthermore, for $0 \leq \tau \leq \mu$, the subspace polynomial $s_\tau(x) = \kappa_\tau(x)$.

Proof: Obviously, V_μ is an additive subgroup of $GF(2^m)$ whose order is equal to 2^μ. Furthermore, one can verify that $V_\tau = \{\omega_0, \omega_1, \ldots, \omega_{2^\tau-1}\}$ is a subgroup of V_μ and that for all indices $0 \leq j < 2^\tau$ and $0 \leq l < 2^{\mu-\tau}$,

$$\omega_{j+l\cdot 2^\tau} = \omega_{l\cdot 2^\tau} + \omega_j.$$

Hence, V_μ is exactly the desired subgroup K. Finally, the assertion that $s_\tau(x) = \kappa_\tau(x)$ can be easily verified. □

As discussed in Section III, the procedure of computing $\Phi_{J,T}(\boldsymbol{\gamma})$ can be described by

$$\boldsymbol{\gamma} \to \Phi_{J,s_\mu}(\boldsymbol{\gamma}) \to \Phi_{J,\rho}(\boldsymbol{\gamma}) \to \Phi_{J,T}(\boldsymbol{\gamma}),$$

where $\rho(x) = \prod_{j=0}^{t-1}(x - \omega_j)$.

According to (7), the polynomial $\Phi_{J,s_\mu}(\boldsymbol{\gamma})$ can be obtained by

$$\Phi_{J,s_\mu}(\boldsymbol{\gamma}) = \sum_W \Phi_{W,s_\mu}(\bar{\boldsymbol{\gamma}}_W),$$

where W is the coset of V_μ and $\bar{\boldsymbol{\gamma}}_W$ is the sub-vector of $\bar{\boldsymbol{\gamma}} = (\boldsymbol{\gamma}, \mathbf{0})$ that corresponds to W. By the definition of ω_j, we have

$$\omega_{j+l\cdot 2^\mu} = \omega_{l\cdot 2^\mu} + \omega_j,$$

for all $0 \leq j < 2^\mu$ and $0 \leq l < 2^{m-\mu}$, this means that

$$V_\mu + \omega_{l\cdot 2^\mu}$$

is a coset of V_μ for every $0 \leq l < 2^{m-\mu}$. It follows that

$$\Phi_{J,s_\mu}(\boldsymbol{\gamma}) = \sum_l \Phi_{V_\mu+\omega_{l\cdot 2^\mu},s_\mu}(\bar{\boldsymbol{\gamma}}_{V_\mu+\omega_{l\cdot 2^\mu}}).$$

Each of the components in this summation can be computed by the corollary below.

Corollary 2: The coordinate vector $\Phi_{V_\mu+\varepsilon,s_\mu}(\bar{\boldsymbol{\gamma}}_{V_\mu+\varepsilon})$ with respect to $\bar{\mathbb{X}}$ is given by

$$\mathrm{IFFT}_{\bar{\mathbb{X}}}(\delta \cdot \bar{\boldsymbol{\gamma}}_{V_\mu+\varepsilon}, \mu, \varepsilon),$$

where the scalar $\delta = \prod_{j=1}^{2^\mu-1} \omega_j$.

Proof: The assertion follows from Lemma 2. □

Hence, the computation of $\Phi_{J,s_\mu}(\boldsymbol{\gamma})$ can be written as

$$\sum_l \mathrm{IFFT}_{\bar{\mathbb{X}}}(\delta \cdot \bar{\boldsymbol{\gamma}}_{V_\mu+\omega_{l\cdot 2^\mu}}, \mu, \omega_{l\cdot 2^\mu}). \tag{12}$$

We now turn to the computation of $\Phi_{J,\rho}(\boldsymbol{\gamma})$ from $\Phi_{J,s_\mu}(\boldsymbol{\gamma})$.

Corollary 3: $\Phi_{J,\rho}(\boldsymbol{\gamma})$ is the quotient of $\Phi_{J,s_\mu}(\boldsymbol{\gamma})$ divided by $\prod_{j=t}^{2^\mu-1}(x-\omega_j)$.

Proof: The claim is true according to Lemma 4. □

Let $f_\rho(x) = \Phi_{J,\rho}(\boldsymbol{\gamma})$ and $f_{s_\mu}(x) = \Phi_{J,s_\mu}(\boldsymbol{\gamma})$. We use division with remainder to write

$$f_{s_\mu}(x) = f_\rho(x)\prod_{j=t}^{2^\mu-1}(x-\omega_j) + \eta(x),$$

where $\deg(\eta(x)) < 2^\mu - t$. It follows that

$$\eta(\omega_j) = f_{s_\mu}(\omega_j)$$

for $j = t, t+1, \ldots, 2^\mu - 1$. Since $\deg(\eta(x)) < 2^\mu - t$, then it can be determined uniquely based on these evaluations. Once $\eta(x)$ is obtained, one can compute the evaluations of $f_\rho(x)$ at points $\omega_0, \omega_1, \ldots, \omega_{t-1}$ by

$$f_\rho(\omega_l) = \left(\prod_{j=t}^{2^{\mu-1}}(\omega_l - \omega_j)\right)^{-1}(f_{s_\mu}(\omega_l) - \eta(\omega_l)), \qquad (13)$$

$l = 0, 1, \ldots, t-1$. Because $\deg(f_\rho(x)) < t$, then $f_\rho(x)$ can also be determined by these evaluations uniquely.

Recall that we have shown that the problems of computing $\eta(x)$ and $f_\rho(x)$ can be regards as computing the GIDFT under a degree constraint. More precisely, computing $\eta(x)$ is equivalent to finding

$$\Phi_{V_\mu, s_\mu(x)}(\hat{\boldsymbol{\gamma}})$$

under the constraint $\deg(\Phi_{V_\mu,s_\mu(x)}(\hat{\boldsymbol{\gamma}})) < 2^\mu - \epsilon$, where

$$\hat{\boldsymbol{\gamma}} = (\hat{\gamma}_0, \hat{\gamma}_1, \ldots, \hat{\gamma}_{2^\mu-1}) \qquad (14)$$

in which $\hat{\gamma}_j = \delta^{-1}\cdot\eta(\omega_j)$ for $j = t, t+1, \ldots, 2^\mu-1$. Likewise, $f_\rho(x)$ is equal to

$$\Phi_{V_\mu, s_\mu(x)}(\tilde{\boldsymbol{\gamma}})$$

under the constraint $\deg(\Phi_{V_\mu,s_\mu(x)}(\tilde{\boldsymbol{\gamma}})) < t$, where

$$\tilde{\boldsymbol{\gamma}} = (\tilde{\gamma}_0, \tilde{\gamma}_1, \ldots, \tilde{\gamma}_{2^\mu-1}) \qquad (15)$$

in which $\tilde{\gamma}_j = \delta^{-1}\cdot f_\rho(\omega_j)$ for $j = 0, 1, \ldots, t-1$. Therefore, the problems of computing $\eta(x)$ and $f_\rho(x)$ can be solved by Algorithms 1 and 2 straightforwardly.

Finally, $\Phi_{J,T}(\boldsymbol{\gamma})$ can be obtained from $\Phi_{J,\rho}(\boldsymbol{\gamma}) = f_\rho(x)$ by a linear transformation.

A detailed description of the additive FGIDFT is shown in Algorithm 7.

Corollary 4: Algorithm 7 computes $\Phi_{J,T}(\boldsymbol{\gamma})$ and its complexity is $O(2^m\log(t) + t^2)$.

Proof: This corollary follows from Theorem 5. □

V. Decoding Generalized Reed–Solomon Codes

In this section, we apply the FGIDFT to decode GRS codes. The main contribution of this section is as follows.

Theorem 6: For an (n, k) GRS code over $GF(q)$, let μ be the smallest integer such that $2^\mu \geq n-k$. If $GF(q)$ contains a subgroup of order 2^μ and n is proportional to q, then there exists a decoding algorithm whose complexity is $O(n\log(n-k) + (n-k)\log^2(n-k))$.

Algorithm 7 Additive Fast Generalized Inverse Discrete Fourier Transform

Input: A subset J with cardinality n, a vector $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, \ldots, \gamma_{n-1})$ and a polynomial $T(x)$ of degree t.
Output: $\Phi_{J,T}(\boldsymbol{\gamma})$.

1: Let $\bar{\boldsymbol{\gamma}} = (\boldsymbol{\gamma}, \mathbf{0})$.
2: Compute $f_{s_\mu}(x) = \Phi_{J,s_\mu}(\boldsymbol{\gamma})$ by (12).
3: Evaluate $f_{s_\mu}(x)$ at the set V_μ by $\mathrm{FFT}_{\mathbb{X}}$.
4: Given $\eta(\omega_j) = f_{s_\mu}(\omega_j)$ for $j = t, t+1, \ldots, 2^\mu-1$, call Algorithm 3 with input $\hat{\gamma}$ (14) and $\epsilon = t$ to obtain $\eta(x)$.
5: Compute $f_\rho(\omega_j)$ for $j = 0, 1, \ldots, t-1$ by (13).
6: Given $f_\rho(\omega_j)$ for $j = 0, 1, \ldots, t-1$, call Algorithm 4 with input $\tilde{\gamma}$ (15) and $\epsilon = t$ to obtain $f_\rho(x)$.
7: Apply a linear transformation to $f_\rho(x)$ to get $\Phi_{J,T}(\boldsymbol{\gamma})$.
8: **return** $\Phi_{J,T}(\boldsymbol{\gamma})$.

Before proving this theorem, we first introduce some background. In subsequent content of this section, we assume that $GF(q)$ contains a subgroup K of order 2^μ and that $2^\mu \geq n-k$. As before, this implies that the FFT and IFFT are available.

A. Generalized Reed–Solomon Codes

Let n and k be integers satisfying $0 < k \leq n \leq q$. Let $\mathcal{L} = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ be n distinct elements in $GF(q)$ and let $\mathbf{w} = (w_0, w_1, \ldots, w_{n-1})$ whose elements are all nonzero in $GF(q)$. The generalized Reed–Solomon (GRS) code $GRS_k(\mathcal{L}, \mathbf{w})$ consists of all vectors

$$(w_0 f(\alpha_0), w_1 f(\alpha_1), \ldots, w_{n-1} f(\alpha_{n-1})), \qquad (16)$$

where $f(x) \in GF(q)[x]$ satisfying $\deg(f(x)) < k$. $GRS_k(\mathcal{L}, \mathbf{w})$ code is an (n, k, d) code over $GF(q)$, where $d = n-k+1$ is the minimum distance.

The dual code of $GRS_k(\mathcal{L}, \mathbf{w})$ is also a GRS code. The parity check matrix of $GRS_k(\mathcal{L}, \mathbf{w})$ can be written as

$$H = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_0 & \alpha_1 & \ldots & \alpha_{n-1} \\ \vdots & \vdots & \ldots & \vdots \\ \alpha_0^{n-k-1} & \alpha_1^{n-k-1} & \ldots & \alpha_{n-1}^{n-k-1} \end{pmatrix} \cdot \begin{pmatrix} u_0 & & & \\ & u_1 & & \\ & & \ldots & \\ & & & u_{n-1} \end{pmatrix}. \qquad (17)$$

where $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) \in GF(q)^n$ is a vector whose components are all nonzero. It can be shown that

$$u_i = w_i^{-1}\left(\prod_{0\leq j\leq n-1, j\neq i}(\alpha_i - \alpha_j)\right)^{-1}.$$

B. Decoding Based on the FGIDFT

For a codeword of $GRS_k(\mathcal{L}, \mathbf{w})$, say $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$, the received vector can be written

as

$$\begin{aligned}\mathbf{y} &= (y_0, y_1, \ldots, y_{n-1}) \\ &= (c_0, c_1, \ldots, c_{n-1}) + (e_0, e_1, \ldots, e_{n-1}) \\ &= \mathbf{c} + \mathbf{e},\end{aligned}$$

where $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$ is the error pattern in which $e_i \neq 0$ implies an error occurs at position i. The error location set E is defined as

$$E = \{i | e_i \neq 0, i = 0, 1, \ldots, n-1\}.$$

Given a polynomial $T(x) \in GF(q)[x]$ of degree $n-k$, the *generalized syndrome* with respect to the received vector $\mathbf{y}$ is defined by

$$\mathbf{S}(x) = \sum_{i=0}^{n-1} y_i u_i \frac{T(x) - T(\alpha_i)}{x - \alpha_i}. \tag{18}$$

The concept of generalized syndromes was initially proposed in [14] for illustrating the relationship between various key equations of RS codes. We generalize this concept to GRS codes here.

Remark: Comparing (18) with (2), one can see that they have the same form. However, the GIDFT is defined in a more general manner, and its properties are also investigated in this paper. In particular, the FGIDFT, which is a fast algorithm for computing the GIDFT, is proposed in this paper.

Lemma 8 ([14]): The syndrome $\mathbf{S}(x)$ satisfies that

$$\mathbf{S}(x) = \sum_{i \in E} e_i u_i \frac{T(x) - T(\alpha_i)}{x - \alpha_i}.$$

Proof: The proof can be found in [14]. However, we present it in Appendix A for self-containment. □

Define the error locator polynomial by

$$\lambda(x) = \prod_{i \in E} (x - \alpha_i).$$

Lemma 9: The generalized syndrome $\mathbf{S}(x)$ and the error locator polynomial $\lambda(x)$ are related by the key equation

$$\mathbf{S}(x)\lambda(x) = \theta(x)T(x) + z(x), \tag{19}$$

where

$$\begin{aligned}\theta(x) &= \sum_{i \in E} e_i u_i \prod_{\substack{j \in E \\ j \neq i}} (x - \alpha_j), \\ z(x) &= -\sum_{i \in E} e_i u_i T(\alpha_i) \prod_{\substack{j \in E \\ j \neq i}} (x - \alpha_j).\end{aligned}$$

Proof: The proof can also be found in [14]. Likewise, we provide it in Appendix B for self-containment. □

The decoding procedure can be divided into four phases: 1) Computing the generalized syndrome $\mathbf{S}(x)$; 2) Solving the key equation (19) to find $\lambda(x)$ and $\theta(x)$; 3) Finding the roots of $\lambda(x)$ to determine the error locations; 4) Computing the corresponding error values. We shall describe these phases in turn.

1) Computing the Generalized Syndrome: It is straightforward to see that computing the generalized syndrome (18) is equivalent to computing the GIDFT over $\mathcal{L}$ and $T(x)$, where $\boldsymbol{\gamma} = (y_0 u_0, y_1 u_1, \ldots, y_{n-1} u_{n-1})$. Thus, we can make use of the FGIDFT in Section III to obtain the generalized syndrome. Since any $T(x)$ of degree $n-k$ can be used to define the generalized syndrome, we fix

$$T(x) = \prod_{j=0}^{n-k-1} (x - \beta_j),$$

where $\beta_j \in K$. In other words, $T(x)$ is equal to $\rho(x)$ that appeared in Section III. There are two reasons for making such a choice. On one hand, Step 7 in Algorithm 4 is no longer needed since the linear transformation is the identity map when $T(x) = \rho(x)$. On the other hand, a fast key equation solver exists when using this $T(x)$. Clearly, with the aid of the FGIDFT, computing the generalized syndrome costs $O(q \log(n-k))$ operations.

2) Solving the Key Equation: When $T(x) = \prod_{j=0}^{n-k-1}(x - \beta_j)$, the key equation (19) is an interpolation problem. The Euclidean algorithm [7], the modular approach [15], and the Welch–Berlekamp algorithm [16] are capable of solving it. Furthermore, assuming the FFT and IFFT are available, the fast modular approach proposed in [5] solves this interpolation problem within $O((n-k)\log^2(n-k))$ operations. It should be noted that, although the fast modular approach in [5] requires that $n-k$ is a power of 2. This constraint could be removed straightforwardly.

3) Finding the Error Locations: The error locations can be determined by finding the roots of the error locator polynomial $\lambda(x)$. Straightforwardly, one can evaluate $\lambda(x)$ at each coset of K (and on $\{0\}$ when K is multiplicative). Since $\deg(\lambda(x)) < 2^\mu$, the complexity of evaluating $\lambda(x)$ at each coset is $O(2^\mu \log(2^\mu))$. Hence, the complexity of finding the error locations is $O(q \log(2^\mu))$.

4) Computing the Error Values: For an error location $i \in E$, the corresponding error value is computed by

$$e_i = \frac{\theta(\alpha_i)}{u_i \lambda'(\alpha_i)},$$

where $\lambda'(x)$ denotes the formal derivative of $\lambda(x)$. Since there are two polynomial evaluations plus one multiplication and one inverse, the complexity of computing the error values is approximately two times of that of finding the error locations. This implies that the complexity is $O(q \log(2^\mu))$.

We are now ready to prove Theorem 6.

Proof of Theorem 6: Since $GF(q)$ contains a subgroup K of order 2^μ, the FFT and IFFT are available. So, according to the above discussion, the total complexity for decoding an (n, k) GRS code is

$$\begin{aligned}&O(q\log(n-k)) + O((n-k)\log^2(n-k)) \\ &+ O(q\log(2^\mu)) + O(q\log(2^\mu)) \\ &= O(q\log(n-k)) + (n-k)\log^2(n-k)).\end{aligned}$$

Since we assume that n is proportional to q, we can conclude that there exists a decoding algorithm for an (n, k) GRS code,

whose complexity is $O(n\log(n-k))+(n-k)\log^2(n-k))$. □

Example 3: Consider a $(40, 34)$ RS code defined over $GF(41)$. Let the sets J, K and the vector $\boldsymbol{\gamma}$ be the same as in Example 2. If $\boldsymbol{\gamma}$ is the received vector, according to the above discussion, $f_\rho(x)$ is exactly the syndrome corresponding to $\boldsymbol{\gamma}$, i.e.,

$$\mathbf{S}(x) = f_\rho(x) = x^5 + 39x^4 + 22x^3 + 37x^2 + 11.$$

The key equation is

$$\mathbf{S}(x)\lambda(x) = \theta(x)\prod_{j=0}^{5}(x-\beta_j) + z(x),$$

which can be solved by the fast modular approach and we obtain

$$\lambda(x) = 8x^3 + 3x^2 + 33x + 38, \theta(x) = 8x^2 + 28x + 8.$$

The roots of $\lambda(x)$ are $1, 40$ and 15, which are the error locators. Since $1, 40$ and 15 are the first, second and twenty-second elements of J, the error locations are $1, 2$ and 22. The error values are obtained by substituting $1, 40$ and 15 into

$$\frac{\theta(x)}{\lambda'(x)} = \frac{8x^2+28x+8}{24x^2+6x+33},$$

which are $2, 7$ and 33. Hence, the decoding result of $\boldsymbol{\gamma}$ is

$$\begin{aligned}&(10, 5, 35, 38, 34, 11, 38, 4, 35, 11, 24, 7, 26,\\ &39, 18, 11, 18, 38, 13, 1, 18, 29, 5, 36, 4, 36,\\ &22, 11, 11, 38, 40, 38, 26, 33, 14, 4, 17, 27, 0, 36).\end{aligned}$$

□

C. Compared with Previous Works

Based on the FGIDFT, we have proposed a decoding algorithm for GRS codes whose computational complexity is $O(n\log(n-k))+(n-k)\log^2(n-k))$. In [4] and [5], decoding algorithms for RS codes defined over $GF(2^m)$ were proposed which also achieve $O(n\log(n-k))+(n-k)\log^2(n-k))$ complexity. Compared with them, a key difference is that the proposed algorithm no longer requires the field characteristic to be 2. Indeed, when decoding RS codes over $GF(2^m)$, the syndromes that appeared in [4] and [5], though defined in a different way, are essentially the GIDFT over $GF(2^m)$ and $T(x) = \prod_{j=0}^{n-k-1}(x-\beta_j)$, where $\beta_j \in K$. This implies that their methods can be seen as special cases of the proposed decoding framework. Furthermore, the derivation of the key equation here is more transparent than those in [4] and [5], as every component of the key equation can be displayed explicitly, as shown in Lemma 9.

VI. Decoding Binary Goppa Codes

This section is devoted to decoding for Goppa codes. Goppa codes are of great significance because they achieve the Gilbert–Varshamov bound and serve as the basis for the McEliece cryptosystem, an important post-quantum cryptography scheme. This section mainly focuses on separable binary Goppa codes, which are the most commonly encountered in practice (Note that the techniques discussed here can also be applied to non-separable Goppa codes). In this section, we propose a fast decoding algorithm for separable binary Goppa codes that utilizes the additive FGIDFT presented in Section IV. We also provide complexity comparisons between the new algorithm and others in the literature for decoding practical Goppa codes.

A. Decoding Algorithm for Separable Binary Goppa Codes

Let n be an integer satisfying $0 < n \le 2^m$ and let $L = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ be n distinct elements in field $GF(2^m)$. Choose a polynomial $G(x) \in GF(2^m)[x]$ such that $G(\alpha_i) \neq 0$ for all $\alpha_i \in L$, where $G(x)$ is called the Goppa polynomial. For any vector $\mathbf{a} = (a_0, a_1, \ldots, a_n)$ over $GF(2)$, we associate a rational function

$$R_{\mathbf{a}}(x) = \sum_{i=0}^{n-1}\frac{a_i}{x-\alpha_i}.$$

The Goppa code $\Gamma(L, G)$ consists of all vectors $\mathbf{a} \in GF(2)^n$ such that

$$R_{\mathbf{a}}(x) \equiv 0 \pmod{G(x)}.$$

Let $r = \deg(G(x))$. The code $\Gamma(L, G)$ is capable of correcting $\lfloor r/2 \rfloor$ errors. Its parity check matrix can be written as:

$$H = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_0 & \alpha_1 & \ldots & \alpha_{n-1} \\ \vdots & \vdots & \ldots & \vdots \\ \alpha_0^{r-1} & \alpha_1^{r-1} & \ldots & \alpha_{n-1}^{r-1} \end{pmatrix} \cdot \begin{pmatrix} G(\alpha_0)^{-1} & & & \\ & G(\alpha_1)^{-1} & & \\ & & \ldots & \\ & & & G(\alpha_{n-1})^{-1} \end{pmatrix}. \quad (20)$$

If $G(x)$ has no multiple zeros, we say that $\Gamma(L, G)$ is separable, and if $G(x)$ is irreducible, we say that $\Gamma(L, G)$ is irreducible. Hence, an irreducible Goppa code is separable. A separable Goppa code $\Gamma(L, G)$ is equivalent to $\Gamma(L, \bar{G})$, where $\bar{G}(x) = G(x)^2$. Note that, by definition, we call $G(x)$ the Goppa polynomial of this separable Goppa code even though $\bar{G}(x)$ generates the same code as $G(x)$. Since $\deg(\bar{G}(x)) = 2r$, a separable Goppa code $\Gamma(L, G)$ is capable of correcting r errors. In order to have enough syndromes to decode up to r errors, we decode $\Gamma(L, \bar{G})$ instead of $\Gamma(L, G)$ for separable Goppa codes.

For a codeword $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1}) \in \Gamma(L, \bar{G})$, the received vector can be written as

$$\begin{aligned}\mathbf{y} &= (y_0, y_1, \ldots, y_{n-1})\\ &= (a_0, a_1, \ldots, a_{n-1}) + (e_0, e_1, \ldots, e_{n-1})\\ &= \mathbf{a} + \mathbf{e},\end{aligned}$$

where $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$ is the error pattern in which $e_i \neq 0$ implies an error occurs at position i. The error location set E is defined as $E = \{i | e_i \neq 0, i = 0, 1, \ldots, n-1\}$. Given the received vector $\mathbf{y}$, the original syndrome for $\Gamma(L, G)$ (see [17] or [8] for details) is defined by

$$R_{\mathbf{y}}(x) \bmod G(x) = \sum_{i=0}^{n-1} y_i G(\alpha_i)^{-1}\frac{G(x)-G(\alpha_i)}{x-\alpha_i}.$$

This is equivalent to the GIDFT of $(y_0G(\alpha_0)^{-1}, y_1G(\alpha_1)^{-1}, \ldots, y_{n-1}G(\alpha_{n-1})^{-1})$ with respect to L and $G(x)$. However, in order to derive a fast decoding algorithm, we choose another specific $T(x)$ to define the syndrome for $\Gamma(L, \bar{G})$, as we will show below.

Let

$$T(x) = \prod_{j=0}^{2r-1}(x - \omega_j).$$

The corresponding syndrome of $\mathbf{y}$ is defined by

$$\mathbf{S}(x) = \sum_{i=0}^{n-1} y_i \bar{G}(\alpha_i)^{-1} \frac{T(x) - T(\alpha_i)}{x - \alpha_i}. \tag{21}$$

Define the error locator polynomial

$$\lambda(x) = \prod_{i \in E}(x - \alpha_i)$$

according to the error location set E. The key equation of $\Gamma(L, \bar{G})$ is then given by

$$\mathbf{S}(x)\lambda(x) = \theta(x)T(x) + z(x), \tag{22}$$

where

$$\theta(x) = \sum_{i \in E} e_i \bar{G}(\alpha_i)^{-1} \prod_{\substack{j \in E \\ j \neq i}}(x - \alpha_j),$$
$$z(x) = \sum_{i \in E} e_i \bar{G}(\alpha_i)^{-1} T(\alpha_i) \prod_{\substack{j \in E \\ j \neq i}}(x - \alpha_j).$$

Now, the procedure of decoding Goppa codes can be summarized as follows:

1) Compute the syndrome $\mathbf{S}(x)$;
2) Solve the key equation (22) to obtain $\lambda(x)$;
3) Find the roots of $\lambda(x)$ to determine the error locations.

Note that for binary Goppa codes, the error values do not need to be computed.

Clearly, the generalized syndrome (21) is equivalent to compute

$$\Phi_{L,T}(\boldsymbol{\gamma}),$$

where $\boldsymbol{\gamma} = (y_0\bar{G}(\alpha_0)^{-1}, y_1\bar{G}(\alpha_1)^{-1}, \ldots, y_{n-1}\bar{G}(\alpha_{n-1})^{-1})$. Hence, Algorithm 7 can be used to compute it efficiently.

The key equation (22)

$$\mathbf{S}(x)\lambda(x) = \theta(x) \prod_{j=0}^{2r-1}(x - \omega_j) + z(x) \tag{23}$$

is an interpolation problem. The fast modular approach (see [5] for details) is capable of efficiently solving this key equation.

Once the error locator polynomial $\lambda(x)$ has been found, we can compute the roots of $\lambda(x)$ by

$$\mathrm{FFT}_{\bar{\mathbb{X}}}(\bar{\lambda}, \mu, \omega_{l \cdot 2^\mu}), 0 \leq l < 2^{m-\mu}, \tag{24}$$

where $\bar{\lambda}$ represents the coordinate vector of $\lambda(x)$ with respect to $\bar{\mathbb{X}}$ and μ is the smallest integer such that $2^\mu \geq \deg(\bar{G}(x))$.

A complete description of the fast decoding algorithm for Goppa codes is presented in Algorithm 8. Its complexity is $O(n\log(r) + r\log^2(r))$ when n is proportional to 2^m.

We now provide an example to illustrate the decoding procedure of binary Goppa codes.

Algorithm 8 Fast Decoding Algorithm for Binary Separable Goppa Codes

Input: Received vector $\mathbf{y} = \mathbf{c} + \mathbf{e}$.
Output: The codeword $\mathbf{c}$.
1: Compute the vector $\boldsymbol{\gamma}$ in which $\gamma_i = y_i\bar{G}(\alpha_i)^{-1}$.
2: Compute the generalized syndrome polynomial $\Phi_{L,T}(\boldsymbol{\gamma})$ by Algorithm 7.
3: Solve the key equation (23) by the fast modular approach to get $\lambda(x)$.
4: Find the error locations by (24).
5: **return** $\mathbf{y} - \mathbf{e}$.

Example 4: Consider the field $GF(2^4) = GF(2)[x]/(x^4 + x + 1)$ and an irreducible polynomial $G(x) = x^3 + x + 1$ over it. The set $\{v_0, v_1, v_2, v_3\}$, where $v_i = x^i$, is a basis of $GF(2^4)$ over $GF(2)$. Let the set L be

$$\{\omega_0, \omega_1, \ldots, \omega_{15}\},$$

where ω_j is defined by (11). Then $\Gamma(L, G)$ is a $(16, 4)$ Goppa code whose generator matrix is

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Since $G(x)$ is irreducible, we have $\Gamma(L, G) = \Gamma(L, \bar{G})$, where $\bar{G}(x) = G(x)^2 = x^6 + x^2 + 1$.

Suppose the all-zero codeword is transmitted and the received vector $\mathbf{y}$ is

$$(0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0),$$

which implies that the errors occur at positions $1, 6$ and 14. We apply Algorithm 8 to decode the received vector $\mathbf{y}$.

Since

$$\bar{G}(\omega_1)^{-1} = \omega_1, \bar{G}(\omega_6)^{-1} = \omega_6, \bar{G}(\omega_{14})^{-1} = \omega_7,$$

it follows that

$$\boldsymbol{\gamma} = (0, \omega_1, 0, 0, 0, 0, \omega_6, 0, 0, 0, 0, 0, 0, 0, \omega_7, 0).$$

Let $T(x) = \prod_{j=0}^{5}(x - \omega_j)$. The generalized syndrome polynomial $\Phi_{L,T}(\boldsymbol{\gamma})$ can be obtained using Algorithm 7, which is

$$\begin{aligned} \mathbf{S}(x) &= \Phi_{L,T}(\boldsymbol{\gamma}) \\ &= \omega_{10}\bar{X}_4(x) + \omega_8\bar{X}_3(x) + \omega_{12}\bar{X}_2(x) \\ &\quad + \omega_8\bar{X}_1(x) + \omega_{14}\bar{X}_0(x). \end{aligned}$$

Note that we use the basis $\bar{\mathbb{X}}$ in this example. The key equation is

$$\mathbf{S}(x)\lambda(x) = \theta(x) \prod_{j=0}^{5}(x - \omega_j) + z(x).$$

Solving this equation by the fast modular approach, we obtain

$$\lambda(x) = \omega_5\bar{X}_3(x) + \omega_{14}\bar{X}_2(x) + \omega_3\bar{X}_1(x) + \omega_3\bar{X}_0(x).$$

Next, the roots of $\lambda(x)$ can be computed using (24); they are ω_1, ω_6 and ω_{14}. Finally, we flip the received vector $\mathbf{y}$ at positions $1, 6$ and 14 and obtain the all-zero codeword, which is the transmitted codeword.

B. Complexity Comparisons

Complexity comparisons between the proposed algorithm and techniques in the literature are provided in Table I and II, when decoding Goppa codes. The comparisons are made by counting the field operations required to correct a received vector, where the number of errors equals the error correction capability. The code parameters are chosen from the post-quantum cryptography scheme that has been submitted to NIST; see [9] for more details. The code in Table I is defined over $GF(2^{12}) = GF(2)[x]/(x^{12}+x^3+1)$, which is of length 3488. The Goppa polynomial is $y^{64}+y^3+y+x$, and its error correction capability is equal to 64. The code in Table II is defined over $GF(2^{13}) = GF(2)[x]/(x^{13}+x^4+x^3+x+1)$, which is of length 8192. The Goppa polynomial is $y^{128}+y^7+y^2+y+1$ and its error correction capability is equal to 128.

For clarity, we describe the decoding procedures for Methods 1 and 2 that are listed in Tables I and II. Let H denote the parity check matrix of $\Gamma(L,G)$, which is shown in (20), and let H_G denote the parity check matrix of $\Gamma(L,\bar{G})$ which can be written as

$$H_G = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_0 & \alpha_1 & \dots & \alpha_{n-1} \\ \vdots & \vdots & \dots & \vdots \\ \alpha_0^{2r-1} & \alpha_1^{2r-1} & \dots & \alpha_{n-1}^{2r-1} \end{pmatrix} \cdot \begin{pmatrix} G(\alpha_0)^{-2} & & & \\ & G(\alpha_1)^{-2} & & \\ & & \ddots & \\ & & & G(\alpha_{n-1})^{-2} \end{pmatrix}.$$

Let $\boldsymbol{\gamma}$ denote the received vector.

Method 1 first computes the syndrome $\mathbf{S}(x)$ by $\boldsymbol{\gamma}H_G^T$, where H_G^T denotes the transpose of H_G. The syndrome $\mathbf{S}(x)$ and the error locator polynomial $\lambda(x)$ satisfies

$$\mathbf{S}(x)\lambda(x) \equiv z(x) \pmod{x^{2r}}.$$

This key equation is solved by the extended Euclidean algorithm, yielding $\lambda(x)$. Finally, the roots of $\lambda(x)$, whose reciprocals are the error locators, are computed by the Chien search. Recall that the computation of error values is not needed since the code is binary. Method 1 can be found in [8, Chapter 12].

Method 2, which is usually called the Patterson algorithm, first computes the syndrome $\mathbf{S}(x)$ by $\boldsymbol{\gamma}H^T$. The syndrome $\mathbf{S}(x)$ and the error locator polynomial $\lambda(x)$ satisfies the key equation:

$$\mathbf{S}(x)\lambda(x) \equiv \lambda^{'}(x) \pmod{G(x)},$$

where $\lambda^{'}(x)$ is the formal derivative of $\lambda(x)$. Since $G(x)$ is irreducible, one can find $f(x)$ by the extended Euclidean algorithm such that $\mathbf{S}(x)f(x) = 1 \bmod G(x)$. Next, we apply a linear transformation on $f(x)+x$ to obtain $d(x)$ such that

$$d(x)^2 = (f(x)+x) \bmod G(x).$$

Solving the equation

$$d(x)\theta(x) \equiv z(x) \pmod{G(x)}$$

by the extended Euclidean algorithm, then the error locator polynomial is given by

$$\lambda(x) = x\theta(x)^2 + z(x)^2.$$

Finally, the roots of $\lambda(x)$, which are the error locators, are computed by the Chien search. The Patterson algorithm is given in Algorithm 4 of [6].

The comparisons in Table I and II show that the proposed algorithm reduces the number of field operations by 76% and 88% compared to the two traditional methods, respectively.

VII. Comments and Open Problems

This section provides some comments, along with several open problems. The FGIDFT requires that $GF(q)$ contains a subgroup whose order is a power of 2, which ensures that the FFT and IFFT are available. Nevertheless, the FFT and IFFT algorithms are available if the order of a subgroup is smooth, i.e., the order is a product of small primes. For example, for a subgroup whose order is $O(3^m)$, the FFT and IFFT for this subgroup exist. If we let the degree of $T(x)$ be equal to the order of the subgroup, one can similarly derive the FGIDFT as in Section III. However, if $\deg(T(x))$ is not equal to the order, it is not clear yet how to design fast algorithms for Problems 3 and 4. This will lead to a code parameter constraint when applying the FGIDFT to decoding algorithms. Hence, how to design the FGIDFT for more general cases is still open.

The FGIDFT is built on an additive FFT or a multiplicative FFT of $GF(q)$. In [18], a new kind of FFT was proposed, which is based on the automorphism groups of rational function fields, called the Galois theory FFT (G-FFT). A generalization of the FGIDFT, which utilizes the G-FFT, remains an open problem.

In Section V, we presented decoding algorithms for GRS codes. Indeed, the proposed algorithm works for any code whose parity matrix has the form (17), no matter how the code is defined. Naturally, the question is whether the encoding can utilize the FGIDFT or not. The answer is yes for GRS codes if we choose the parity positions carefully. Let $\boldsymbol{\gamma}$ be a codeword. Then the corresponding syndrome

$$\Phi_{J,T}(\boldsymbol{\gamma}) = 0.$$

If we let the parity positions be $\beta_0, \beta_1, \dots, \beta_{t-1}$, according to the discussion in Section III, we have

$$\begin{aligned} &\Phi_{J,\kappa}(\boldsymbol{\gamma}) \\ &= \Phi_{GF(q),\kappa}(\bar{\boldsymbol{\gamma}}) \\ &= \sum_{W} \Phi_{W,\kappa}(\bar{\boldsymbol{\gamma}}_W) + I_K \cdot \Phi_{\{0\},\kappa}(\bar{\boldsymbol{\gamma}}_{\{0\}}) \\ &= \Phi_{K,\kappa}(\bar{\boldsymbol{\gamma}}_K) + \sum_{\substack{W \\ W \neq K}} \Phi_{W,\kappa}(\bar{\boldsymbol{\gamma}}_W) + I_K \cdot \Phi_{\{0\},\kappa}(\bar{\boldsymbol{\gamma}}_{\{0\}}). \end{aligned}$$

In the last equality, the second and third terms can be computed since $\bar{\boldsymbol{\gamma}}_W$ and $\bar{\boldsymbol{\gamma}}_{\{0\}}$ are known for $W \neq K$. Denote their summation by $\theta(x)$. Recall that $\Phi_{K,\kappa}$ is an isomorphism, which implies that there is a vector $\boldsymbol{\sigma}$ such that $\theta(x) = \Phi_{K,\kappa}(\boldsymbol{\sigma})$. It follows that

$$\begin{aligned} \Phi_{J,\kappa}(\boldsymbol{\gamma}) &= \Phi_{K,\kappa}(\bar{\boldsymbol{\gamma}}_K) + \theta(x) \\ &= \Phi_{K,\kappa}(\bar{\boldsymbol{\gamma}}_K + \boldsymbol{\sigma}). \end{aligned}$$

TABLE I
COMPLEXITY COMPARISON WHEN DECODING GOPPA CODE OF LENGTH 3488 AND OF CORRECTION CAPABILITY 64.

Components	Method 1 [8, Chapter 12]			Method 2 [6, Algorithm 4]			Proposed decoding		
	Mul.	Add.	Div.	Mul.	Add.	Div.	Mul.	Add.	Div.
Syndrome	0	446,464	0	0	223,232	0	18,528	33,792	128
Key equation	16,705	20,673	129	16,506	25,127	189	30,704	41,256	0
Chien search	226,720	226,720	3,488	226,720	226,720	0	14,336	28,736	0
Total	243,425	693,857	3,617	243,226	475,079	189	63,568	103,784	128

TABLE II
COMPLEXITY COMPARISON WHEN DECODING GOPPA CODE OF LENGTH 8192 AND OF CORRECTION CAPABILITY 128.

Components	Method 1 [8, Chapter 12]			Method 2 [6, Algorithm 4]			Proposed decoding		
	Mul.	Add.	Div.	Mul.	Add.	Div.	Mul.	Add.	Div.
Syndrome	0	2,096,896	0	0	1,048,576	0	42,496	76,032	256
Key equation	66,177	82,305	257	65,786	101,447	381	73,712	101,608	0
Chien search	1,056,639	1,056,639	8,191	1,056,768	1,056,768	0	32,768	65,664	0
Total	1,122,816	3,235,840	8,448	1,122,554	2,206,791	381	148,976	243,304	256

Since $\Phi_{J,T}(\boldsymbol{\gamma}) = 0$, the degree of $\Phi_{J,\kappa}(\boldsymbol{\gamma})$ is less than $2^\mu - t$ by Lemma 4. Furthermore, the components in $\bar{\boldsymbol{\gamma}}_K + \boldsymbol{\sigma}$ that corresponds to $\beta_t, \beta_{t+1}, \ldots, \beta_{2^\mu-1}$ are known. So the last equality can be regarded as an instance of Problem 3. Therefore, Algorithm 1 solves it and we get $\Phi_{J,\kappa}(\boldsymbol{\gamma})$ and of course $\bar{\boldsymbol{\gamma}}_K + \boldsymbol{\sigma}$. Evidently, $\bar{\boldsymbol{\gamma}}_K$ will be obtained. This completes the encoding. It is straightforward to see that the complexity of encoding is $O(q \log(t))$. Notably, if the parity positions are chosen arbitrarily, deriving a fast encoding algorithm is still an open issue.

An irreducible Goppa code $\Gamma(L, G)$ whose Goppa polynomial $G(x)$ has degree r is capable of correcting r errors. However, the decoding algorithm presented in Section VI corrects r errors for $\Gamma(L, G)$ by decoding its equivalent codeword in $\Gamma(L, \bar{G})$, where $\bar{G}(x) = G(x)^2$. Deriving a more efficient algorithm for irreducible Goppa codes remains an open problem.

VIII. CONCLUSION

In this work, we generalize the concept of the IDFT to the GIDFT. The properties of the GIDFT are investigated. Furthermore, we propose the FGIDFT, which computes the GIDFT efficiently when $GF(q)$ contains a desired subgroup. Next, we apply the FGIDFT to decode GRS codes and prove that for any (n, k) GRS code, if $GF(q)$ contains a subgroup of order $2^\mu \geq n - k$ and n is proportional to q, then there exists a fast decoding algorithm whose complexity is $O(n \log(n-k) + (n-k)\log^2(n-k))$. We also describe the additive FGIDFT for $GF(2^m)$ in detail and derive a new decoding algorithm for binary Goppa codes based on the additive FGIDFT. Compared with other known methods, the new one is superior in terms of computational complexity. Complexity comparisons show that the new algorithm reduces the number of field operations by $76\% - 88\%$ when decoding Goppa codes used in post-quantum cryptography.

APPENDIX A
PROOF OF LEMMA 8

Proof: This proof can be found in [14]. However, we repeat the proof here to make this paper self-contained.

A codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ satisfies $H\mathbf{c}^T = 0$. It follows that

$$\sum_{i=0}^{n-1} c_i u_i \alpha_i^l = 0, \text{ for } l = 0, 1, \ldots, r-1.$$

If we write $T(x) = \sum_{j=0}^{n-k} T_j x^j$, we have

$$\begin{aligned} T(x) - T(\alpha_i) &= \sum_{j=0}^{n-k} T_j (x^j - \alpha_i^j) \\ &= \sum_{j=1}^{n-k} T_j (x - \alpha_i) \sum_{l=0}^{j-1} x^{j-1-l} (\alpha_i)^l. \end{aligned}$$

This leads to

$$\frac{T(x) - T(\alpha_i)}{x - \alpha_i} = \sum_{j=1}^{n-k} T_j \sum_{l=0}^{j-1} x^{j-1-l} \alpha_i^l.$$

Hence, one has

$$\begin{aligned} \mathbf{S}(x) &= \sum_{i=0}^{n-1} y_i u_i \sum_{j=1}^{n-k} T_j \sum_{l=0}^{j-1} x^{j-1-l} \alpha_i^l \\ &= \sum_{j=1}^{n-k} T_j \sum_{l=0}^{j-1} x^{j-1-l} \sum_{i=0}^{n-1} y_i u_i \alpha_i^l \\ &= \sum_{j=1}^{n-k} T_j \sum_{l=0}^{j-1} x^{j-1-l} \sum_{i=0}^{n-1} (c_i + e_i) u_i \alpha_i^l \\ &= \sum_{j=1}^{n-k} T_j \sum_{l=0}^{j-1} x^{j-1-l} \sum_{i \in E} e_i u_i \alpha_i^l \\ &= \sum_{i \in E} e_i u_i \sum_{j=1}^{n-k} T_j \sum_{l=0}^{j-1} x^{j-1-l} \alpha_i^l \\ &= \sum_{i \in E} e_i u_i \frac{T(x) - T(\alpha_i)}{x - \alpha_i}. \end{aligned}$$

□

APPENDIX B
PROOF OF LEMMA 9

Proof: According to Lemma 8, we have

$$\begin{aligned}
&\mathbf{S}(x)\lambda(x)\\
&= \sum_{i\in E} e_i u_i \frac{T(x)-T(\alpha_i)}{x-\alpha_i} \prod_{j\in E}(x-\alpha_j)\\
&= \sum_{i\in E} e_i u_i T(x) \prod_{\substack{j\in E\\ j\neq i}}(x-\alpha_j) - \sum_{i\in E} e_i u_i T(\alpha_i) \prod_{\substack{j\in E\\ j\neq i}}(x-\alpha_j)\\
&= T(x)\sum_{i\in E} e_i u_i \prod_{\substack{j\in E\\ j\neq i}}(x-\alpha_j) - \sum_{i\in E} e_i u_i T(\alpha_i) \prod_{\substack{j\in E\\ j\neq i}}(x-\alpha_j).
\end{aligned}$$

By taking

$$\theta(x) = \sum_{i\in E} e_i u_i \prod_{\substack{j\in E\\ j\neq i}}(x-\alpha_j),$$

$$z(x) = -\sum_{i\in E} e_i u_i T(\alpha_i) \prod_{\substack{j\in E\\ j\neq i}}(x-\alpha_j),$$

the proof is complete. □

REFERENCES

[1] S. Gao, "A new algorithm for decoding Reed–Solomon codes," in *Communications, Information and Network Security*. Boston, MA: Springer US, 2003, pp. 55–68.

[2] J. Justesen, "On the complexity of decoding Reed–Solomon codes (corresp.)," *IEEE Trans. Inf. Theory*, vol. 22, no. 2, pp. 237–238, Mar. 1976.

[3] 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.

[4] 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.

[5] N. Tang and Y. S. Han, "A new decoding method for Reed–Solomon codes based on FFT and modular approach," *IEEE Trans. Commun.*, vol. 70, no. 12, pp. 7790–7801, Dec. 2022.

[6] N. Patterson, "The algebraic decoding of Goppa codes," *IEEE Trans. Inf. Theory*, vol. 21, no. 2, pp. 203–207, Mar. 1975.

[7] 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, Jan. 1975.

[8] F. J. MacWilliams and N. J. A. Sloane, "Alternant, Goppa and other generalized BCH codes," in *The Theory of Error-Correcting Codes*, vol. 16. Amsterdam, New York, Oxford: North-Holland Publishing Company, 1977, pp. 365–368.

[9] D. J. Bernstein *et al.*, "Classic McEliece: conservative code-based cryptography," Oct. 2022. [Online]. Available: https://classic.mceliece.org/

[10] R. E. Blahut, "Codes based on the Fourier transform," in *Algebraic Codes for Data Transmission*. New York: Cambridge University Press, 2003, pp. 131–132.

[11] S.-J. Lin, T. Y. Al-Naffouri, and Y. S. Han, "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.

[12] O. Ore, "On a special class of polynomials," *Trans. Amer. Math. Soc.*, vol. 35, no. 3, pp. 559–584, Jul. 1933.

[13] S.-J. Lin, W.-H. Chung, and Y. S. Han, "Novel polynomial basis and its application to Reed–Solomon erasure codes," in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, Philadelphia, PA, USA, Oct. 2014, pp. 316–325.

[14] K. Araki and I. Fujita, "Generalized syndrome polynomials for decoding Reed–Solomon codes," *IEICE Trans. Fundamentals*, vol. 75, no. 8, pp. 1026–1029, Aug. 1992.

[15] 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.

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

[17] V. D. Goppa, "A new class of linear correcting codes," *Problemy Peredachi Informatsii*, vol. 6, no. 3, pp. 24–30, 1970.

[18] S. Li and C. Xing, "Fast Fourier transform via automorphism groups of rational function fields," in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Alexandria, Virginia, USA, Jan. 2024, pp. 3836–3859.

Nianqi Tang received the Ph.D. degree in communication and information systems from Xidian University, China, in 2019. From 2019 to 2023, he was a Senior Engineer with Huawei Technologies Co., Ltd. From 2023 to 2025, he was an assistant professor with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China. Since 2025, he has served as an algorithm researcher at Theo End (Shenzhen) Computing Technology Co., Ltd. His research interests include error control coding, data compression, and artificial intelligence.

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 the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respec tively, 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 with the School of Computer and Information Science, Syracuse University, from 1992 to 1993. From 1993 to 1997, he was an Associate Professor with 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 with the Department of Electrical Engineering, University of Hawaii at Manoa, Honolulu, HI, USA, from June 2001 to October 2001; the SUPRIA Visiting Research Scholar with the Department of Electrical Engineering and Computer Science and the CASECenter, Syracuse University, from September 2002 to January 2004 and from July 2012 to June 2013; and a Visiting Scholar with the Department of Electrical and Computer Engineering, The University of Texas at Austin, 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 with the Department of Electrical Engineering, National Taiwan University of Science and Technology. He has been a Chair Professor at the National Taipei University since February 2015. From February 2017 to February 2021, he was with the School of Electrical Engineering and Intelligentization, Dongguan University of Technology, China. He is currently with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China. His research interests include error-control coding, wireless networks, and security. He was a Winner of the 1994 Syracuse University Doctoral Prize. One of his papers won the prestigious 2013 ACM CCS Test-of-Time Award in Cybersecurity.

Chao Chen received the Ph.D. degree in communication and information systems from Xidian University, China, in 2010. From 2010 to 2014, he was an Engineer with China Academy of Space Technology (CAST), Xi'an. From 2014 to 2015, he was a Postdoctoral Fellow with Institute of Network Coding (INC), The Chinese University of Hong Kong. He is currently an Associate Professor with the State Key Laboratory of Integrated Services Networks (ISN), Xidian University, China. His research interests include channel coding and source coding.

Danyang Pei received the B.S. degree in Electronic Information from Beijing University of Posts and Telecommunications in 2023, and the M.S. degree in Electronic Information from the University of Electronic Science and Technology of China in 2026. Her research interests include error control coding and wireless communications.