A Group-Type Distributed Coded Computation Scheme Based on a Gabidulin Code

Koki Kazama
Waseda University
Email: kokikazama@aoni.waseda.jp

Toshiyasu Matsushima
Waseda University
Email: toshimat@waseda.jp

***Abstract*—We focus on a distributed coded computation scheme for matrix multiplication. In this system, the product matrix is encoded and decoded through the overall system to correct errors in computation. We propose a group-type distributed coded computation scheme, for one example, a scheme based on a Gabidulin code, and evaluate the computation time complexity and the error-correcting capability of the overall system.**

The full version of this paper is in [1].

I. Introduction

We focus on a distributed computation scheme in which errors are corrected in computing multiplication of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ on a finite field $\mathbb{F}_q$. In the system of a distributed computation scheme, the main computer (master) partitions the matrix and distributes them to multiple computers (workers), and (2) workers perform parallel computing. This system has the advantage of decreasing the computation time complexity, while this has the disadvantage of increasing the possibility of occurring errors in computation. To eliminate the advantage, a distributed coded computation scheme (DCC) [2], [3], [4], [5] uses an error-correcting code (ECC) to correct errors in distributed computation. On performance evaluation on DCCs, (a) computation time complexity (CTC) and (b) error-correcting capability of the overall system are important criteria while they are a tradeoff. Considering the reason to construct DCCs, we would like to propose the DCC which can compute $\boldsymbol{AB}$ more efficiently than the stand-alone scheme (SA), of which the system computes $\boldsymbol{AB}$ solely. Moreover, the system of the proposed scheme can correct some errors.

In this paper, we propose a new distributed coded computation scheme called *Group-Type Distributed Coded Computation scheme* (GDCC) and, as one example, a *GDCC based on a Gabidulin code* (GDCCG). A Gabidulin code encodes a matrix over $\mathbb{F}_q$ and correct an error matrix $\boldsymbol{E}$ if $\mathrm{rank}(\boldsymbol{E}) \leq t$, where t is a constant defined later. The key idea is that computing an inner product of two vectors over an extension field $\mathbb{F}_{q^m}$ can be decomposed to computations over $\mathbb{F}_q$, which can be parallelly performed. Using these facts, this distributed computation system performs a process over $\mathbb{F}_q$, which is equivalent to encoding a vector over $\mathbb{F}_{q^m}$ corresponding $\boldsymbol{AB}$ to a codeword over $\mathbb{F}_{q^m}$. Thus the encoder in the GDCCG system encodes all columns of $\boldsymbol{AB}$ together while the encoders in previous DCC systems encode each column of $\boldsymbol{AB}$. This enables to correct error matrices whose columns depend on each other and enables to decrease the overall CTC of the system simultaneously. The GDCCG is a little different from the scheme of [5] because more workers are used and they are equally partitioned into multiple groups and the parallel computation of matrix products is performed within each group. Thus the computation time complexity of the GDCCG is less than that of [5], while the error-correcting capability of the GDCCG is the same as that of [5].

Moreover, we evaluate (a) the CTC and (b) the error-correcting capability of the GDCC in detail and show the advantages as follows. In the evaluation on (a), we evaluate the CTC of the GDCCG and the SA and we show the condition of parameters (the number of workers and groups) in which the GDCC is superior to the SA. We cannot generally evaluate (a) on GDCC because the CTC of the decoding algorithm is depending on the code. Thus we evaluate (a) only on GDCCG. We define the CTC of a DCC system as the number of four arithmetic operations over $\mathbb{F}_q$ in parallel computing of each worker and decoding of the master. To evaluate the number of operations of the encoder and the decoder using a Gabidulin code over $\mathbb{F}_{q^m}$ in the GDCCG system, we first show how many times it is necessary when the operation of $\mathbb{F}_{q^m}$ is decomposed into the operation of $\mathbb{F}_q$, and then we count them. In the evaluation on (b), we show what error matrices the GDCC system and the GDCCG system can correct, respectively. Specifically, we show that the GDCCG system can correct an error matrix if $\mathrm{rank}(\boldsymbol{E}) \leq t$, where t is a certain constant.

II. Notations

For two integers m and n which satisfies $m \leq n$, $[m, n]$ denotes $[m, n] \coloneqq \{\, m, m+1, \ldots, n \,\}$. $[n]$ denotes $[1, n]$. All vectors are column vectors except specifically noted. $\boldsymbol{E}^\top$ is the transpotent of a matrix $\boldsymbol{A}$. $\mathbb{F}_q$ is a finite field with q elements, where q is a power of a prime number. $\mathbb{F}_q^{n\times b}$ denotes the set of all $n \times b$ matrices over $\mathbb{F}_q$, and $\mathbb{F}_q^n \coloneqq \mathbb{F}_q^{n\times 1}$. $\boldsymbol{e}_{\cdot j} \in \mathbb{F}_q^n$ denotes j-th column of a matrix $\boldsymbol{E} \in \mathbb{F}_q^{n\times b}$ $\boldsymbol{e}_{i\cdot} \in \mathbb{F}_q^b$ denotes the transpotent of i-th row vector. Thus the matrix $\boldsymbol{E}$ is $(\boldsymbol{e}_{\cdot 1}, \ldots, \boldsymbol{e}_{\cdot b}) = (\boldsymbol{e}_{1\cdot}, \ldots, \boldsymbol{e}_{n\cdot})^\top$. For any set $A \subset [n]$, we define a matrix $\boldsymbol{G}_{A\cdot}^\top \in \mathbb{F}_q^{|A|\times k_{\mathrm{A}}}$ as a matrix constructed from all $i(\in A)$-th row $\boldsymbol{g}_{i\cdot}^\top \in \mathbb{F}_q^{1\times k_{\mathrm{A}}}$ of the matrix $\boldsymbol{G}$. ? denotes the symbol representing an erasure or decoding failure. We

This research is supported in part by Grant-in-Aid JP17K06446 for Scientific Research (C).

formally define the sum and difference of a and ? as ? for any $a \in \mathbb{F}_q$.

Definition 2.1 ($\boldsymbol{v}$, $\boldsymbol{f}^s$): $v_1, \ldots, v_m \subset \mathbb{F}_{q^m}$ are linearly independent over $\mathbb{F}_q$ and $v_i = v_1^{q^{i-1}}$ for any $i \in [m]$, i.e. $\{v_1, \ldots, v_m\}$ is a normal basis [6] over $\mathbb{F}_{q^m}$. We define $\boldsymbol{v}$ as a vector $(v_1, \ldots, v_m)$. For any $s \in \mathbb{N}$, a linear homomorphism $\boldsymbol{f}^s \colon \mathbb{F}_{q^m}^s \to \mathbb{F}_q^{s \times m}$ over $\mathbb{F}_q$ outputs a matrix $\boldsymbol{X} \in \mathbb{F}_q^{s \times m}$ which satisfies $\boldsymbol{x} = \boldsymbol{X}\boldsymbol{v}$ from the input $\boldsymbol{x} \in \mathbb{F}_{q^m}^s$. When $s = 1$, $f^1 \coloneqq \boldsymbol{f}^s$.

III. Computation Time Complexity

We explain the problem setting, especially the definition of computation time complexity, of distributed coded computation schemes in this paper. In this paper, q is a power of 2. Positive integers $n, k_{\mathrm{A}}, k_{\mathrm{B}}, l$ satisfy $2 \leq k_{\mathrm{A}} < n, 2 \leq k_{\mathrm{B}}$ and $2 \leq l$. m is $\max\{k_{\mathrm{B}}, n\}$.

The schemes in this paper compute $\boldsymbol{AB} \in \mathbb{F}_q^{k_{\mathrm{A}} \times k_{\mathrm{B}}}$. In this paper, we consider schemes for computing a multiplication of two matrices, $\boldsymbol{A} \in \mathbb{F}_q^{k_{\mathrm{A}} \times l}$ and $\boldsymbol{B} \in \mathbb{F}_q^{l \times k_{\mathrm{B}}}$. One value of $\boldsymbol{A}$ is input to the master only once, and the master store this value. On the other hand, many values of $\boldsymbol{B}$ are input to the master many times, and each time the master attempts to compute the value of the matrix $\boldsymbol{AB}$.

The most simple scheme is as follows.

Definition 3.1: We define a *stand-alone scheme* (SA) as a scheme in which the master computes $\boldsymbol{AB}$ solely from the input $\boldsymbol{A}, \boldsymbol{B}$.

We would like to compute $\boldsymbol{AB}$ more efficiently than the SA system, i.e. to construct a computing system whose CTC is less than that of the SA system.

Definition 3.2: We define *computation time complexity* (CTC) of a process as the number of four arithmetic operations over $\mathbb{F}_q$ which the system performs in the process from input to output. A subtraction, an addiction, a multiplication, and an inversion are equally treated as one operation in the evaluation of the CTC. *CTC of the system* is the overall CTC from input $\boldsymbol{B}$ to output $\hat{\boldsymbol{AB}}$. [1]

Proposition 3.1: CTC of the SA system is $k_{\mathrm{A}} k_{\mathrm{B}}(2l-1)$.

IV. Relation between ECCs and Previous DCCs

As a basis for the proposition, we explain the correspondence in principle between previous DCCs and ECCs in [5]. Specifically, an encoding of error-correcting codes corresponds to an encoding of $\boldsymbol{AB}$ into the system's output result $\Pi(\boldsymbol{AB})$. GDCCG encodes all columns of $\boldsymbol{AB}$ together.

This system encodes $\boldsymbol{AB}$ to $\Pi(\boldsymbol{AB})$ using a function $\Pi \colon \mathbb{F}_q^{k_{\mathrm{A}} \times k_{\mathrm{B}}} \to \mathcal{C}(\subset \mathbb{F}_q^{n \times b})$, and decode a matrix $\boldsymbol{Y} = \Pi(\boldsymbol{AB}) + \boldsymbol{E}$, where $\boldsymbol{E} \coloneqq (\boldsymbol{e}_{1\cdot}, \ldots, \boldsymbol{e}_{n\cdot})^\top$. $\Pi(\boldsymbol{AB})$ is a matrix whose i-th row is $\boldsymbol{g}_{i\cdot}^\top \boldsymbol{AB}$ for any $i \in [n]$. $\boldsymbol{AB}$, $\Pi(\boldsymbol{AB})$, $\boldsymbol{E}$, $\boldsymbol{Y}$, $\boldsymbol{G}$, Π, ψ and $\mathcal{C}$ correspond with information, codeword, error ,recieved word, generator matrix, encoder, decoder, code in ECCs, respectively. We call them product matrix, *codeword (matrix)*, *error (matrix)*, *recieved matrix*, *generator matrix*, *encoder*, *decoder*, *code*, respectively.

[1] $\hat{\boldsymbol{AB}}$ may not be $\boldsymbol{AB}$ by errors.

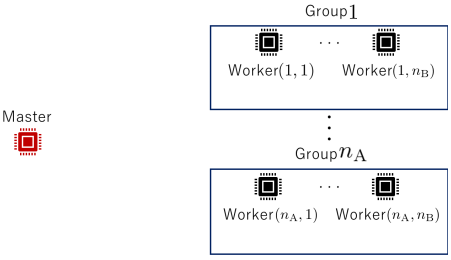

Fig. 1. the master and the workers

We propose a scheme which $\Pi(\boldsymbol{AB})$ is a codeword $\boldsymbol{f}^n(\boldsymbol{GA}(\boldsymbol{B}, \boldsymbol{0}_{l \times (m-k_{\mathrm{B}})})\boldsymbol{v})$ of a code $\mathcal{C}(\subset \mathbb{F}_q^{n \times m})$. We explain more details after. In particular, the scheme whose matrix-encoder Π is a Gabidulin encoder can correct low-rank error matrices $\boldsymbol{E}$, which cannot be corrected by previous schemes. The workers in this system are partitioned into multiple groups for shortening the computational time complexity of the system.

V. The Proposed Scheme

In this section, we propose a computation scheme called *GDCC*. Moreover, we propose a scheme called *GDCC with a Gabidulin code* (GDCCG) as an example of a GDCC.

First, we define symbols. We define $\tilde{\mathcal{C}} \subset \mathbb{F}_{q^m}^n$ as an (n, k_{A}) linear code over $\mathbb{F}_{q^m}$. This code has a generator matrix $\boldsymbol{G} \in \mathbb{F}_{q^m}^{n \times k_{\mathrm{A}}}$ of a canonical systematic encoder. $\mathcal{C} \coloneqq \boldsymbol{f}^n(\tilde{\mathcal{C}})(\subset \mathbb{F}_q^{n \times m})$ is a linear code over $\mathbb{F}_q$. The decoder of $\mathcal{C}$ is $\psi \colon \mathbb{F}_q^{n \times m} \to \mathcal{C} \cup \{?\}$. We define a set $\mathcal{E}$ $(\subset \mathbb{F}_q^{n \times m})$ as the set of all error matrices which can be corrected rightly, i.e. for any $\boldsymbol{E} \in \mathcal{E}$ and $\boldsymbol{C} \in \mathcal{C}$, $\psi(\boldsymbol{C} + \boldsymbol{E}) = \boldsymbol{C}$.

Lemma 5.1: Any codeword of a linear code $\mathcal{C} = \boldsymbol{f}^n(\tilde{\mathcal{C}})$ over $\mathbb{F}_q$ is $\boldsymbol{f}^n(\boldsymbol{GMv})$ for some matrix $\boldsymbol{M} \in \mathbb{F}_q^{k_{\mathrm{A}} \times m}$.

Proof : A codeword an (n, k_{A}) linear code $\tilde{\mathcal{C}} \subset \mathbb{F}_{q^m}^n$ over $\mathbb{F}_{q^m}$ is $\boldsymbol{GMv}$ for some matrix $\boldsymbol{M} \in \mathbb{F}_q^{k_{\mathrm{A}} \times m}$. □

A. GDCC

We propose a GDCC $((\pi_{ij} | i \in [n_{\mathrm{A}}], j \in [n_{\mathrm{B}}]), \boldsymbol{G}, \psi)$ when $q, k_{\mathrm{A}}, k_{\mathrm{B}}, l, n, n_{\mathrm{A}}$ and n_{B} are given. In this scheme, we use the master and $n_{\mathrm{A}} n_{\mathrm{B}}$ workers which are partitioned into multiple groups, where $n_{\mathrm{A}} \in \mathbb{N}$ devides n and $n_{\mathrm{B}} \in \mathbb{N}$ devides m. All workers are equally partitioned into n groups. The j-th worker of the i-th group is called *a worker* $(i, j) \in [n_{\mathrm{A}}] \times [n_{\mathrm{B}}]$ (Figure 1). We assume that errors occur only when workers compute something. The master has the matrix $\boldsymbol{G} \in \mathbb{F}_{q^m}^{n \times k_{\mathrm{A}}}$ mentioned above and a function ψ. A function π_{ij} are stored in each worker $(i, j) \in [n_{\mathrm{A}}] \times [n_{\mathrm{B}}]$.

In GDCC, when the matrix $\boldsymbol{A}$ is input to the master, then the master and all workers perform *preprocess*. For any time when the matrix $\boldsymbol{B}$ is input to the master, all workers perform *Computing Process*, and then the master performs *Decoding Process*. The result of Decoding Process of the master is $\hat{\boldsymbol{AB}} \in \mathbb{F}_q^{k_{\mathrm{A}} \times k_{\mathrm{B}}}$, which is an estimated results of $\boldsymbol{AB}$. See the details in below (Figure 2).

[$\langle$*Preprocess*$\rangle$] (Figure 3) The master encodes $\boldsymbol{A}$ to

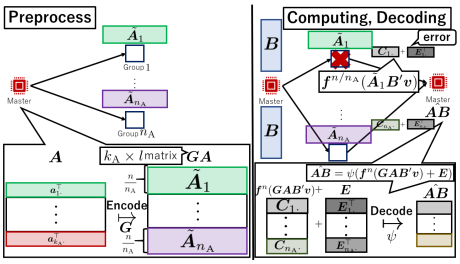

Fig. 2. the flow of the DCC

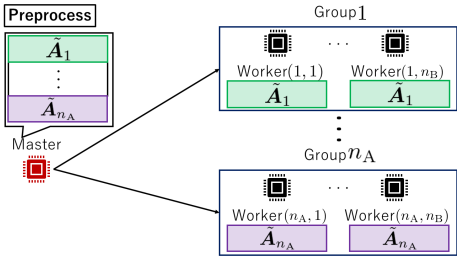

Fig. 3. Preprocess

$$\boldsymbol{GA} = \begin{pmatrix} \boldsymbol{G}^{\top}_{[1,n/n_{\mathrm{A}}]\cdot}\boldsymbol{A} \\ \vdots \\ \boldsymbol{G}^{\top}_{[(n_{\mathrm{A}}-1)(n/n_{\mathrm{A}})+1,n]\cdot}\boldsymbol{A} \end{pmatrix} \in \mathbb{F}_{q^m}^{n\times l}. \tag{1}$$

The master store $\boldsymbol{G}^{\top}_{[(i-1)(n/n_{\mathrm{A}})+1,i(n/n_{\mathrm{A}})]\cdot}\boldsymbol{A} \in \mathbb{F}_{q^m}^{(n/n_{\mathrm{A}})\times l}$ in all workers of each group $i \in [n_{\mathrm{A}}]$. Then, the workers in the i-th group store the j'-th symbol of a matrix $f^1(\boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot l'}v_{m'}) \in \mathbb{F}_q^{1\times m}$ for all $l' \in [l]$, $m' \in [m]$, $i' \in [(i-1)(n/n_{\mathrm{A}})+1, i(n/n_{\mathrm{A}})]$ and $j' \in [(j-1)(m/n_{\mathrm{B}})+1, j(m/n_{\mathrm{B}})]$. We define this as $\tilde{a}_{i'l'm'j'} \in \mathbb{F}_q$. $\boldsymbol{a}_{\cdot l'} \in \mathbb{F}_q^{k_{\mathrm{A}}}$ is the l'-th column of the matrix $\boldsymbol{A}$. $\tilde{a}_{i'l'm'j}$ can be computed from $\boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{A} = (\boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot 1}, \ldots, \boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot l})$.

[⟨*Computing Process*⟩] (Figure 4) The master sends the matrix $\boldsymbol{B}$ to all workers. Hereafter, we define $\boldsymbol{B}' = \boldsymbol{B}$ if $n \le k_{\mathrm{B}}$, and $\boldsymbol{B}' = (\boldsymbol{B}, \boldsymbol{0})$ if $n > k_{\mathrm{B}}$, where $\boldsymbol{0}$ is an $l \times (m - k_{\mathrm{B}})$ zero matrix over $\mathbb{F}_q$. Each worker (i,j) computes the $(j-1)(m/n_{\mathrm{B}})+1, \ldots, j(m/n_{\mathrm{B}})$-th columns (we think them as *j-th block*) of an $(m/n_{\mathrm{B}}) \times m$ matrix $\boldsymbol{C}_{i\cdot} := \boldsymbol{f}^{m/n_{\mathrm{B}}}\left(\boldsymbol{G}_{[(i-1)(m/n_{\mathrm{B}})+1,i(m/n_{\mathrm{B}})]\cdot}\boldsymbol{AB'v}\right)$. This computation can be done by computing $\sum_{l'\in[l]}\sum_{m'\in[m]}\tilde{a}_{i'l'm'j'}b_{l'm'}$ from $\boldsymbol{B}$ for any $(i',j') \in [(i-1)(m/n_{\mathrm{B}})+1, i(m/n_{\mathrm{B}})] \times [(j-1)(m/n_{\mathrm{B}})+1, j(m/n_{\mathrm{B}})]$. See proposition 5.1.

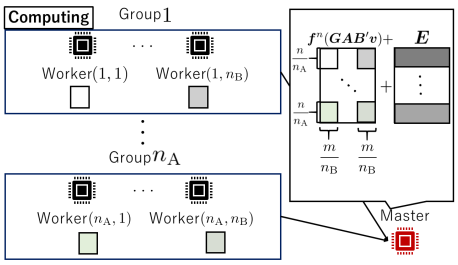

Fig. 4. Computing Process

For any $(i,j) \in [n_{\mathrm{A}}] \times [n_{\mathrm{B}}]$, we define the correct computing result of any worker (i,j) as $\pi_{ij}(\boldsymbol{g}_i^{\top}\boldsymbol{A}, \boldsymbol{B})$. we define *product function* as the function $\pi_{ij}\colon \mathbb{F}_{q^m}^{(n/n_{\mathrm{A}})\times l} \times \mathbb{F}_q^{l\times k_{\mathrm{B}}} \to \mathbb{F}_q$ which computes the $(j-1)(m/n_{\mathrm{B}})+1, \ldots, j(m/n_{\mathrm{B}})$-th columns of the matrix $\boldsymbol{C}_{i\cdot}$ from $\boldsymbol{G}^{\top}_{[(i-1)(n/n_{\mathrm{A}})+1,i(n/n_{\mathrm{A}})]\cdot}\boldsymbol{A} \in \mathbb{F}_{q^m}^{(n/n_{\mathrm{A}})\times l}$ and $\boldsymbol{B}$. We assume that the error $e_{i'j'} \in \mathbb{F}_q$ occurs in computing the j'-th symbol of $f^1\left(\boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{AB'v}\right) \in \mathbb{F}_q^{1\times m}$. The master recieves a matrix $\boldsymbol{Y} := \boldsymbol{f}^n(\boldsymbol{GAB'v}) + \boldsymbol{E}$, where $\boldsymbol{E}$ is a matrix whose (i',j')-th entry is $e_{i'j'}$ for any $(i',j') \in [n] \times [m]$. This matrix is constructed from all output results of all workers.

[⟨*Decoding Process*⟩] The master gets $\psi(\boldsymbol{Y})$ from the matrix $\boldsymbol{Y}$ with a function $\psi : \mathbb{F}_q^{n\times m} \to \mathcal{C} \cup \{?\}$, where a symbol $? \notin \mathcal{C}$ represents a fact that the master cannot get an estimated matrix $\hat{\boldsymbol{AB}}$. Since the generator matrix is a generator matrix of a canonical systematic encoder, if $\psi(\boldsymbol{Y}) \in \mathcal{C}$, the master gets $\hat{\boldsymbol{AB}}$ from $\psi(\boldsymbol{Y})$.

Assumption 5.1: We do not include CTC of the preprocess in the evaluation on (a) since $\boldsymbol{A}$ is input only once. Moreover, we do not include any communication time between workers and the master.

Proposition 5.1: For any $(i',j') \in [(i-1)(m/n_{\mathrm{B}})+1, i(m/n_{\mathrm{B}})] \times [(j-1)(m/n_{\mathrm{B}})+1, j(m/n_{\mathrm{B}})]$, the (i',j')-th entry of a matrix $\boldsymbol{f}^n(\boldsymbol{GABv})$ is $\sum_{l'\in[l]}\sum_{m'\in[k_{\mathrm{B}}]}\tilde{a}_{i'l'm'j'}b_{l'm'}$.

Proof : It is the j'-th entry of

$$\begin{aligned}
&f^1\left(\boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{AB'v}\right) \\
&= f^1\left(\begin{pmatrix} \boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot 1} & \ldots & \boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot l} \end{pmatrix}\begin{pmatrix} \sum_{m'\in[m]} b'_{1m'}v_{m'} \\ \vdots \\ \sum_{m'\in[m]} b'_{lm'}v_{m'} \end{pmatrix}\right) \\
&= f^1\left(\begin{pmatrix} \boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot 1} & \ldots & \boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot l} \end{pmatrix}\begin{pmatrix} \sum_{m'\in[k_{\mathrm{B}}]} b_{1m'}v_{m'} \\ \vdots \\ \sum_{m'\in[k_{\mathrm{B}}]} b_{lm'}v_{m'} \end{pmatrix}\right) \\
&= f^1\left(\sum_{l'\in[l]} \boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot l'}\left(\sum_{m'\in[k_{\mathrm{B}}]} b_{l'm'}v_{m'}\right)\right) \\
&= f^1\left(\sum_{l'\in[l]}\sum_{m'\in[k_{\mathrm{B}}]} b_{l'm'}\left(\boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{a}_{\cdot l'}\right)v_{m'}\right) \\
&= f^1\left(\sum_{j\in[m]}\left(\sum_{l'\in[l]}\sum_{m'\in[k_{\mathrm{B}}]}\tilde{a}_{i'l'm'j}b_{l'm'}\right)v_j\right) \\
&= \left(\sum_{l'\in[l]}\sum_{m'\in[k_{\mathrm{B}}]}\tilde{a}_{i'l'm'1}b_{l'm'}, \ldots, \sum_{l'\in[l]}\sum_{m'\in[k_{\mathrm{B}}]}\tilde{a}_{i'l'm'm}b_{l'm'}\right). \square
\end{aligned}$$

Corollary 5.1: Any worker $(i,j) \in [n_{\mathrm{A}}] \times [n_{\mathrm{B}}]$ performs $(lk_{\mathrm{B}}-1)(mn/n_{\mathrm{A}}n_{\mathrm{B}})$ additions over $\mathbb{F}_q$ and $lk_{\mathrm{B}}(mn/n_{\mathrm{A}}n_{\mathrm{B}})$ multiplications over $\mathbb{F}_q$ in Computing Process.

Proof : Proposition 5.1 shows that any worker (i,j) computes the j'-th symbol $\sum_{l'\in[l]}\sum_{m'\in[k_{\mathrm{B}}]}\tilde{a}_{i'l'm'j'}b_{l'm'}$ of $f^1(\boldsymbol{g}^{\top}_{i'\cdot}\boldsymbol{AB'v})$ for any $(i',j') \in [(i-1)(m/n_{\mathrm{B}})+1 : i(m/n_{\mathrm{B}})] \times [(j-1)(m/n_{\mathrm{B}})+1, j(m/n_{\mathrm{B}})]$ in Computing

Process. Moreover, each $\tilde{a}_{i'l'm'j'}$ is already computed in Preprocess. Thus the worker (i,j) performs $lk_{\mathrm{B}}-1$ additions over $\mathbb{F}_q$ and lk_{B} multiplications over $\mathbb{F}_q$ to compute $\sum_{l'\in[l]}\sum_{m'\in[k_{\mathrm{B}}]}\tilde{a}_{i'l'm'j'}b_{l'm'}$ from the input $\boldsymbol{B}$. The worker (i,j) performs this computation for all $(i',j') \in [(i-1)(m/n_{\mathrm{B}})+1, i(m/n_{\mathrm{B}})] \times [(j-1)(m/n_{\mathrm{B}})+1, j(m/n_{\mathrm{B}})]$. □

B. Example : GDCC Based on a Gabidulin Code

We propose a GDCC based on a Gabidulin code as an example of GDCCs.

Definition 5.1 (Gabidulin code [7]): Let $m,k\ (\in\mathbb{N})$ satisfy $m\geq n\geq k$. Let $h_1,\ldots,h_n\in\mathbb{F}_{q^m}$ be $h_i=v_i$ for any $i\in[n]$, where $\{v_1,\ldots,v_m\}$ is the optimal normal basis. Let $\boldsymbol{H}\in\mathbb{F}_{q^m}^{n\times(n-k)}$ is a matrix whose (i,j)-th entry is $h_i^{q^{j-1}}$ for any $(i,j)\in[n]\times[k]$. An (n,k) linear code $\tilde{\mathcal{C}}_{\mathrm{G}}\subset\mathbb{F}_{q^m}^n$ over $\mathbb{F}_{q^m}$ which has a parity check matrix $\boldsymbol{H}$ is called an (n,k) *Gabidulin code* over $\mathbb{F}_{q^m}$.

We assume some assumptions in this paper to construct the proposed scheme. To decompose four arithmetic operations over $\mathbb{F}_{q^m}$ to those over $\mathbb{F}_q$, we assume some conditions on q and m.

Definition 5.2 (Multiplication Table [8]): For any $i\in[m]$, we define $T_{i1},\ldots,T_{im}\in\mathbb{F}_q$ as the elements uniquely determined by $v_1v_i=\sum_{j\in[m]}T_{ij}v_j$. $\boldsymbol{T}\coloneqq(T_{ij})_{(i,j)\in[m]^2}\in\mathbb{F}_q^{m\times m}$ is called *multiplication table*. We define $C(\boldsymbol{T})\in\mathbb{Z}$ as the number of nonzero entries of $\boldsymbol{T}$. $C(\boldsymbol{T})$ is called the *complexity of the normal basis* $\{v_1,\ldots,v_m\}$.

Lemma 5.2 ([9]): An addition over $\mathbb{F}_{q^m}$ costs$m$ additions over $\mathbb{F}_q$. Moreover, an multiplication over $\mathbb{F}_{q^m}$ costs $m(C(\boldsymbol{T})+1)-m^2-1$ additions over $\mathbb{F}_q$ and $m(C(\boldsymbol{T})+m)$ multiplications over $\mathbb{F}_q$

Definition 5.3 (Optimal Normal Basis [10]): For any normal basis $\{v_1,\ldots,v_m\}$ and multicative table $\boldsymbol{T}$, $C(\boldsymbol{T})\geq 2m-1$. The normal basis $\{v_1,\ldots,v_m\}$ is called an *optimal* normal basis if it achieves this lower bound.
We set a normal basis $\{v_1,\ldots,v_m\}\in\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ an *optimal* normal basis.

Lemma 5.3 ([10]): An optimal normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ exists if and only if q and m satisfies the following. $\log_2 q$ and m are prime with each other. $2m+1$ is a prime number. A multicative group $(\mathbb{Z}/(2m+1)\mathbb{Z})^\star=(\mathbb{Z}/(2m+1)\mathbb{Z})\setminus\{0\}$ is generated from 2 and -1.

Assumption 5.2 (Assumption for the parameters): We assume the condition of Lemma 5.3.

Definition 5.4 (GDCCG): Let $\boldsymbol{G}\in\mathbb{F}_{q^m}^{n\times k_{\mathrm{A}}}$ is a canonical systematic generator matrix of a (n,k_{A}) Gabidulin code over $\mathbb{F}_{q^m}$. ψ is a bounded rank distance decoder of this code. This GDCC is called *GDCCG*.

VI. Performance Evaluations on the Proposed Schemes

We evaluate (a) the CTC of the GDCCG system and (b) the error-correcting capability of the GDCC system and the GDCCG system. We evaluate (b) on GDCC and GDCCG and show that GDCCG corrects an error matrix that previous schemes cannot correct. Moreover, we compare (a) the CTC on the stand-alone scheme (SA) and that of GDCCG and give the parameter condition when GDCCG is superior to the SA.

A. Evaluations on Error-Correcting Capabilities

We evaluate (b) the error-corrcting capability on GDCC and GDCCG.

Theorem 6.1 (evaluation on (b) on GDCC): The GDCC system computes correctly if the error matrix $\boldsymbol{E}$ is in $\mathcal{E}$.

Proof : GDCC outputs $\boldsymbol{f}^n(\boldsymbol{GAB'v})\in\mathbb{F}_q^{n\times m}$ from $\boldsymbol{A}$ and $\boldsymbol{B}$ when no error occurs in the computation. If $\boldsymbol{E}\in\mathcal{E}$, then $\psi(\boldsymbol{f}^n(\boldsymbol{GAB'v})+\boldsymbol{E})=\boldsymbol{f}^n(\boldsymbol{GAB'v})$ since $\boldsymbol{f}^n(\boldsymbol{GAB'v})\in\mathcal{C}$ from Lemma 5.1. Thus this scheme corrects all error matrices in $\mathcal{E}$. □

Theorem 6.2 (evaluation on (b) on GDCCG): The GDCCG system computes correctly if the error matrix $\boldsymbol{E}$ satisfies $\mathrm{rank}\boldsymbol{E}\leq t$.

Proof : An (n,k_{A}) Gabidulin code over $\mathbb{F}_{q^m}$ can correct an error matrix $\boldsymbol{E}$ if $\mathrm{rank}(\boldsymbol{E})\leq t$. □
This error-correcting capability is the same as that of [5].

B. Evaluations on Computation Time Complexities

1) Assumptions on Computation Time Complexities: We assume some assumptions in this paper to evaluate the CTC. We use corresponding between $\boldsymbol{a}\in\mathbb{F}_{q^m}^n$ and $\boldsymbol{f}^n(\boldsymbol{a})$ in the proposed scheme. We assume Assumption 6.1, also assumed in [8] [11] [12].

Assumption 6.1: We do not include CTC of computing $\boldsymbol{f}^n(\boldsymbol{a})\in\mathbb{F}_q^{n\times m}$ from $\boldsymbol{a}\in\mathbb{F}_{q^m}^n$ in the evaluation on (a). We do not include that of computing $(\boldsymbol{f}^n)^{-1}(\boldsymbol{A})\in\mathbb{F}_{q^m}^n$ from $\boldsymbol{A}\in\mathbb{F}_q^{n\times m}$.

Proposition 6.1: For any q, m and $a\in\mathbb{F}_{q^m}$, if $f^1(a)=(a_1,\ldots,a_m)\in\mathbb{F}_q^{1\times m}$, then $f^1(a^{q^i})=(a_{m-i+1},\ldots,a_m,a_1,a_2,\ldots,a_{m-i})$.

Definition 6.1 (cyclic shift [8]): For any q, m and a, we define i-th*cyclic shift up* as $\boldsymbol{a}^{\uparrow i}\coloneqq f(a^{q^i})$ and *cyclic shift down* $\boldsymbol{a}^{\downarrow i}\coloneqq f(a^{q^i})$.
We assume Assumption 6.2, also assumed in [8] [11] [12].

Assumption 6.2: We do not include computational time complexities of cyclic shifts up or down in the evaluation.

From these assumptions, the below facts hold.

Proposition 6.2: q and m satisfies Assumption 5.2. An addition, which is also a substraction, over $\mathbb{F}_{q^m}$ costs m additions over $\mathbb{F}_q$. A multiplication over $\mathbb{F}_{q^m}$ costs m^2-1 additions and m^2 multiplication over $\mathbb{F}_q$. An inversion over $\mathbb{F}_{q^m}$ costs $(m^2-1)(2\lfloor\log_2(m\log_2 q-1)\rfloor+2)$ additions and $m^2(2\lfloor\log_2(m\log_2 q-1)\rfloor+2)$ multiplications over $\mathbb{F}_q$.

Form Proposition 6.2, Corollary 6.1 holds. The value D is a little modified from the upper bound of the number of operations, derived in [11]. The more details are in Appendix in [1].

Corollary 6.1: q and m satisfies the condition of Lemma 5.3. We define $t=\lfloor(n-k_{\mathrm{A}})/2\rfloor$ and $d=n-k_{\mathrm{A}}+1$. Then D is an upper bound of the complexity of the decoding algorithm [11] of an (n,k_{A})Gabidulin code over $\mathbb{F}_{q^m}$. D is

$$2mnt+m^2+m-1+\frac{2}{3}(m(m-1)(2m-1)-t(t-1)(2t-1))$$

$$
\begin{aligned}
&-(t-2)(m-t)-(t-1)(m^2-m-t^2+t)\\
&+(dn+d^2-t^2+2dt+mt-n-4d-t-1)m\\
&+(dn+3d^2+3t^2-4dt+mt-n-9d+9t+5)(2m^2-1)\\
&+2t(2m^2-1)(2\lfloor \log_2(m\log_2 q-1)\rfloor). \qquad (2)
\end{aligned}
$$

2) Evaluations and A Comparison to the Stand-Alone System: We evaluate (a) on the GDCCG system in case $\mathrm{rank}\boldsymbol{E} \leq t$ using D defined in Eq. (2). The CTC of the GDCC system is the sum of the number of four arithmetic operations over $\mathbb{F}_q$ of each worker and that of Decoding Process of the master.

Theorem 6.3 (evaluation on (a) on GDCCG): Under Assumption 5.2, the sum of CTC of Computation Process of each worker and that of Decoding Process of the master in the GDCCG system is at most

$$(2k_{\mathrm{B}}l-1)(mn/n_{\mathrm{A}}n_{\mathrm{B}})+D. \qquad (3)$$

Proof : We showed from Corollary 5.1 that CTC of Computation Process of each worker $(i,j) \in [n_{\mathrm{A}}] \times [n_{\mathrm{B}}]$ is at most $(2k_{\mathrm{B}}l-1)(mn/n_{\mathrm{A}}n_{\mathrm{B}})$. The master computes $\boldsymbol{f}^n(\boldsymbol{GAB'v})$ from $\boldsymbol{f}^n(\boldsymbol{GAB'v})+\boldsymbol{E}$ with the bounded rank-distance decoder in Decoding Process. We do not include time to compute $\boldsymbol{AB}$ from $\boldsymbol{f}^n(\boldsymbol{GAB'v})$ since $\boldsymbol{G}$ is a generator matrix of a canonical systematic encoder. Thus the CTC is at most D. □

We compare the CTC of the stand-alone scheme.

Corollary 6.2 (Comparison (a) on the proposed scheme with that of the SA system): Under Assumption 5.2, if $n, n_{\mathrm{A}}, n_{\mathrm{B}}$ satisfies the below, the value of Eq.(3) is less than the CTC of the SA system.

$$l > \frac{1}{2k_{\mathrm{B}}(k_{\mathrm{A}}-(mn/n_{\mathrm{A}}n_{\mathrm{B}}))}(k_{\mathrm{A}}k_{\mathrm{B}}-(mn/n_{\mathrm{A}}n_{\mathrm{B}})+D).$$

Example 6.1: Set $n = n_{\mathrm{A}}, m = n_{\mathrm{B}}$, $q = 2$, $k_{\mathrm{A}} = 100$, $k_{\mathrm{B}} = 293$ and $l = 100000$. The value of Eq.(3) is less than the CTC of the SA system when $n(> k_{\mathrm{A}} = 100)$ satisfies $101 \leq n \leq 172$. Table 4.1 of [10] shows that $(q,m) = (2,293)$ satisfy Assumption 5.2.

Remark 6.1: Theorem 6.2 showed that (b) the error-correcting capability of the GDCCG is the same as that of [5]. However, (a) the CTC of the GDCCG is less than or equal to that of the scheme of [5]. This is because, for each $i \in [n]$, one worker in Group i computes a part of $\boldsymbol{g}_i.\boldsymbol{AB'v}$ in the GDCCG, while one worker computes all entries of $\boldsymbol{g}_i.\boldsymbol{AB'v}$ in the scheme of [5]. Since the purpose of this paper is to compare the GDCCG with the SA, we do not compare the GDCCG with the scheme of [5] in detail.

VII. Conclusion and Future Works

In this paper, we proposed and evaluated a new distributed coded computation scheme called GDCC and, as one example, GDCCG. First, we evaluated the GDCCG with (a) the computation time complexity and showed the parameter condition in which the GDCCG system is superior to the SA system. Next, we evaluated the GDCC and the GDCCG with (b) the error-correcting capability of the overall system and showed that the GDCCG system can correct $\boldsymbol{E}$ which satisfies $\mathrm{rank}(\boldsymbol{E}) \leq t$.

In future works, we would like to improve the proposed schemes. For example, if we use more efficient decoding algorithms such as [13], the GDCCG may be better concerning the CTC. For another example, the GDCCG uses more workers than the scheme of [5]. Thus we would like to propose new DCCs considering not only (a) and (b) but also communication load and the number of workers.

References

[1] K. Kazama and T Matsushima. A group-type distributed coded computation scheme based on a gabidulin code. https://onl.bz/w2NxCX5, 2022.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran. Speeding up distributed machine learning using codes. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1143–1147, 2016.

[3] K.H. Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Transactions on Computers*, Vol. C-33, No. 6, pp. 518–528, 1984.

[4] S. Dutta, V. Cadambe, and P. Grover. "short-dot": Computing large linear transforms distributedly using coded short dot products. *IEEE Transactions on Information Theory*, Vol. 65, No. 10, pp. 6171–6193, 2019.

[5] K. Kazama and T Matsushima. A coded computation method based on a gabidulin code and the evaluation of its error-correcting capability (in japanese). *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences A*, Vol. J104-A, No. 6, pp. 156–159, 2021.

[6] H. Niederreiter. *Finite Fields*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1984.

[7] E. M. Gabidulin. Theory of codes with maximum rank distance. *Problems of Information Transmission (English translation of Problemy Peredachi Informatsii)*, Vol. 21, No. 1, pp. 1–12, 1985.

[8] S. Gao, D. Panario, V. Shoup, et al. Algorithms for exponentiation in finite fields. *Journal of Symbolic Computation*, Vol. 29, No. 6, pp. 879–889, 2000.

[9] D. Silva and F. R. Kschischang. Fast encoding and decoding of gabidulin codes. In *2009 IEEE International Symposium on Information Theory (ISIT)*, pp. 2858–2862, 2009.

[10] S. Gao. *Normal bases over finite fields.* University of Waterloo Waterloo, 1993.

[11] M. Gadouleau and Zhiyuan Yan. Complexity of decoding gabidulin codes. In *2008 42nd Annual Conference on Information Sciences and Systems*, pp. 1081–1085, 2008.

[12] R. B. Venturelli and D. Silva. An evaluation of erasure decoding algorithms for gabidulin codes. In *2014 International Telecommunications Symposium (ITS)*, pp. 1–5, 2014.

[13] H. Bartz, T. Jerkovits, S. Puchinger, and J. Rosenkilde. Fast decoding of codes in the rank, subspace, and sum-rank metric. *IEEE Transactions on Information Theory*, Vol. 67, No. 8, pp. 5026–5050, 2021.

Appendix

We explain a decoding algorithm of a Gabidulin code [7] Moreover, we derive an upper bound of four arithmetic operations over $\mathbb{F}_q$. This value is based on [11], though these two are a little different.

First, we explain linearized polynomials.

Definition A.1 (linearized polynomial [6]): A polynomial $F(x) = \sum_{i\in[0,u]} f_i x^{q^i}$ over $\mathbb{F}_{q^m}$ is called a *linearized polynomial*. If $f_u \neq 0$, then we define $\deg_q F(x) := u$ is called the *q-degree* of this linearized polynomial.

The set of all linearized polynomials forms a non-commutative ring by the addition $+$ and the multiplication $*$ [6]. Specifically, the multiplication $*$ is different from that of polynomial rings.

Definition A.2 (+, $*$ [6]): Let $F(x) = \sum_{i\in[0,u]} f_i x^{q^i}$ and $G(x) = \sum_{i\in[0,v]} g_i x^{q^i}$ be linearized polynomials which satisfy $f_u g_v \neq 0$. The addition $F(x)+G(x)$ of two linearized polynomials $F(x)$ and $G(x)$ is defined as the ordinary addition of two polynomials. The multiplication $F(x)*G(x)$ is defined as $F(G(x))$.
For any $i \in [0, u+v]$, the coefficient h_i of $F(x) * G(x) = \sum_{i\in[0,u+v]} h_i x^{q^i}$ is $\sum_{j\in[0,i]} f_i g_{i-j}^{q^j}$. For any linearized polynomial $F(x)$, all roots of $F(x)$ form a linear space over F_q [6].

In the decoding algorithm, the input is a recieved word $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$ and the output is $\hat{\boldsymbol{c}} \in \mathbb{F}_{q^m}^n$. The algorithm is constructed from Step1 to Step 7. There is a big difference between our analysis and the analysis in [11], especially in Step 3. Here we define t as the bounded rank-distance $\lfloor (n-k_{\mathrm{A}})/2 \rfloor$ of the Gabidulin code, and d as the minimum rank-distance $n-k_{\mathrm{A}}+1(\geq 2)$.

1) Compute the syndrome $(s_1, \ldots, s_{d-1})^\top := \boldsymbol{H}^\top \boldsymbol{y} \in \mathbb{F}_{q^m}^{d-1}$.
   It costs $(n-1)(d-1)$ additions over $\mathbb{F}_{q^m}$ and $n(d-1)$ multiplications over $\mathbb{F}_{q^m}$.
2) Compute linearized polynomials $F(x)$ and $\Lambda(x)$ which satisfies $\deg_q F(x) < t$ and $F(x) = \Lambda(x) * S(x) \mod z^{q^{d-1}}$ from the syndrome linearized polynomial $S(x) := \sum_{i\in[n-k]} s_i x^{q^{i-1}}$ by extended Euclidian algorithm.
   We derive upper bounds here. It costs at most $(d-1)(d-2) - \frac{1}{2}t(t-1) + 2t(d-t-2)$ additions over $\mathbb{F}_{q^m}$, at most $(d-1)(d-2) - \frac{1}{2}t(t-5) + 2(d-t-1)(d-t-2)$ multiplications over $\mathbb{F}_{q^m}$, and at most t inversions[2]over $\mathbb{F}_{q^m}$.
   We explain more details later.
3) Compute t roots $E_1, \ldots, E_t \in \mathbb{F}_{q^m}$ of a linearized polynomial $\Lambda(x)$.[3].
   The number of four arithmetic operations (additions, substractions, multiplications, and inversions) over $\mathbb{F}_q$ is $\frac{2}{3}(m(m-1)(2m-1) - t(t-1)(2t-1)) - (t-2)(m-t) - (t-1)(m(m-1) - t(t-1)) + m^2 + m - 1$. The number of additions over $\mathbb{F}_{q^m}$ is tm. The number of multiplications $\mathbb{F}_{q^m}$ is $(t+1)m$. We explain more details later.
4) For any $w \in [r]$, compute $E_1 x_1^w + \cdots + E_t x_t^w = s_w$,where $x_1, \ldots, x_t \in \mathbb{F}_{q^m}$.
   It costs $\frac{3}{2}t(t-1)$ additions over $\mathbb{F}_{q^m}$, $(t+1)\left(\frac{3}{2}t-1\right)$ multiplications over $\mathbb{F}_{q^m}$, and t inversions over $\mathbb{F}_{q^m}$. The derivation is omitted since it is similar to that of [11].
5) For all $w \in [0, t-1]$, compute $\boldsymbol{Y} \in \mathbb{F}_q^{t\times n}$ which satisfies $x_w = Y_{w1}h_1 + \cdots + Y_{wn}h_n$.
   No arithmetic operations are needed since $\{h_1, \ldots, h_n\}$ is $\{v_1, \ldots, v_n\}$.
6) Compute $\boldsymbol{e} = \boldsymbol{Y}(E_1, \ldots, E_t)^\top \in \mathbb{F}_{q^m}^n$.
   The number of four arithmetic operations over $\mathbb{F}_q$ is $mn(2t-1)$ since $\boldsymbol{Y}$ is an $n \times t$ matrix over $\mathbb{F}_q$ and $(E_1, \ldots, E_t)^\top \in \mathbb{F}_{q^m}^t$ corresponds a $t \times m$ matrix over $\mathbb{F}_q$.
7) Compute $\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{e} \in \mathbb{F}_{q^m}^n$.
   It costs mn substractions over $\mathbb{F}_q$.

There is a difference between this paper and [11] as follows. In Step 3, we consider a non-probabilistic algorithm, while they consider a probabilistic algorithm in [11]. Moreover, our evaluation of the number of arithmetic operations in Step 2 is different from that of [11], mainly because our evaluation of the number of arithmetic operations to compute each coefficient of the multiplication of two linearized polynomials is different from that of [11]. Moreover, in Step 6, we show that it costs mnr multiplications over $\mathbb{F}_q$ and "$mn(t-1)$" additions over $\mathbb{F}_q$, while they showed that it costs mnr multiplications over $\mathbb{F}_q$ and "mnt" additions over $\mathbb{F}_q$ in [11]. At last, Step 7 is not written in [11].

A. Step 2

The computation in Step 2 is Step 2-1 and 2-2 as follows The input is the syndrome linearized polynomial $S(x) \in \mathbb{F}_{q^m}[x]$. The putputs are $F(x), \Lambda(x) \in \mathbb{F}_{q^m}[x]$.

Step 2-1 $F_0(x) := x^{q^{d-1}}$, $F_1(x) := S(x)$. Linearized polynomials $G_1(x), G_2(x), \ldots$ and $F_0(x), F_1(x), \ldots$ over $\mathbb{F}_{q^m}$ are recursively defined as follows. For al $i \in \{0, 1, \ldots\}$,

$$\begin{cases} F_i(x) = G_{i+1}(x) * F_{i+1}(x) + F_{i+2}(x) \\ \deg_q F_{i+2}(x) < \deg_q F_{i+1}(x) < \deg_q F_i(x). \end{cases} \tag{4}$$

We define r as the unique positive integer which satisfies $\deg_q F_{i+1}(x) < \frac{d-1}{2} \leq \deg_q F_i(x)$.

Step 2-2 $A_{-1}(x) := 0$ and $F_0(x) := x$. Linearized polynomials $A_1(x), A_2(x), \ldots$ over $\mathbb{F}_{q^m}$ are recursively defined as follows. For all $i \in \{1, 2, \ldots, r\}$,

$$A_i(x) := G_i(x) * A_{i-1}(x) + A_{i-2}(x). \tag{5}$$

Moreover, we define $\Lambda(x)$ as the monic linearized polynomial dividing $A_r(x)$ by its highest order coefficient.

We derive the computation time complexities in Step2-1 and 2-2.

Proposition A.1: It costs uv additions over $\mathbb{F}_{q^m}$ and $(u+1)(v+1)$ multiplications over $\mathbb{F}_{q^m}$ to compute a multiplication of two linearized polynomials $F(x) = \sum_{i\in[0,u]} f_i x^{q^i}$, $G(x) = \sum_{i\in[0,v]} g_i x^{q^i}$ $(f_u g_v \neq 0)$ over $\mathbb{F}_{q^m}$.

Our calculation of this number of times is different from that of [11]. The statement in [11] is that $uv - u - v - 1$ additions and uv multiplications are needed.

Proof : We enumerate the number of multiplications. We do not include the number operations to compute $g_k^{q^k}$ from the given value g_k in the evaluation from Assumption 6.2. Then it costs $(u+1)(v+1)$ multiplications to compute $f_j g_k^{q^k}$ for all $i \in [0, u], k \in [0, v]$. The number of additions are derived if

[2]An inversion is an operation of computing a^{-1} from a. This indicates that an operation of computing ab^{-1} is a combination of an inversion b^{-1} and a multilication ab^{-1}.

[3]There may be at most $t-1$ roots. For simplicity, we assume there are t roots.

the q-degree of the multiplications of linearized polynomials is substracted from the number of the multiplications over $\mathbb{F}_q$. $\square$

Proposition A.2: Let u and v ($u \leq v$) are q-degrees of linearized polynomials $F(x)$ and $G(x)$, respectively. Let $Q(x)$ and $R(x)$ be linearized polynomials which satisfy $F(x) = Q(x) * G(x) + R(x)$ and $\deg_q R(x) < \deg_q G(x)$. It costs at most $v(u-v+1)$ additions over $\mathbb{F}_{q^m}$, at most $(v+1)(u-v+1)$ multiplications over $\mathbb{F}_{q^m}$ and 1 inversion over $\mathbb{F}_{q^m}$ to compute $Q(x)$ and $R(x)$.

The proof is omitted since the result is similar to that of [11].

Proposition A.3: In the above algorithm, $r \leq t$ and $\deg_q \Lambda(x) \leq t$ hold.

Proof : For simplicity, d_i denotes $\deg_q F_i(x)$. For any $i \in [0, r]$, $d_{i+1} < d_i < d-1-i$ holds. Thus,

$$(d_{r+1} <)\frac{d-1}{2} \leq d_r \leq d-1-r. \tag{6}$$

This results $r \leq t$.

$\deg_q G_i(x) = d_{i-1} - d_i$ holds for any $i \in [r]$ from Eq.(4). On the other hand, from Eq.(5), $\deg_q A_i(x) = d_{i-1} - d_i + \deg_q A_{i-1}(x)$ holds for any $i \in [r]$. Thus $\deg_q A_i(x) = d_0 - d_i = d - 1 - d_i$ holds. This indicates that

$$\deg_q \Lambda(x) = \deg_q A_r(x) = d-1-d_r \leq d-1-\frac{d-1}{2} = \frac{d-1}{2}. \tag{7}$$

Therefore $\deg_q \Lambda(x) \leq t$ holds. $\square$

The computation in Step 2-1 costs t inversions, at most $2t(d-1) - \frac{1}{2}t(t-1)$ multiplications, and at most $t(d-1)$ additions over over F_{q^m} for the following reason. From Proposition A.2, for any $i \in [0, r-1]$, it costs 1 inversion, $(d_{i+1}+1)(d_i - d_{i+1}+1)$ multiplications and $d_{i+1}(d_i - d_{i+1}+1)$ additions over F_{q^m} to compute $G_{i+1}(x), F_{i+2}(x)$ from $F_i(x), F_{i+1}(x)$. By doing this from $i = 0$ to $r-1$, it costs $r(\leq t)$ inversion over F_{q^m} , $(d-2)(d-1) - \frac{1}{2}t(t-5)$ multiplications over F_{q^m}, and $(d-2)(d-1) - \frac{1}{2}t(t-1)$ additions over F_{q^m} to compute $F_0(x), \ldots, F_{r+1}(x)$ and $G_1(x), \ldots, G_r(x)$. We derive an upper bound of the number of multiplications as follows.

$$\sum_{i=0}^{r-1}(d_{i+1}+1)(d_i - d_{i+1}+1) \tag{8}$$

$$= r + \sum_{i=0}^{r-1} d_{i+1}(d_i - d_{i+1}) + \sum_{i=0}^{r-1} d_i \tag{9}$$

$$\leq r + d_1 \sum_{i=0}^{r-1}(d_i - d_{i+1}) + \sum_{i=0}^{r-1}(d-1-i) \tag{10}$$

$$\leq r + (d-2)(d-1-d_r) + rd - \frac{1}{2}r(r+1) \tag{11}$$

$$= (d-2)(d-1) - \frac{1}{2}r(r-5) + (d-2)(r-d_r) \tag{12}$$

$$\leq (d-2)(d-1) - \frac{1}{2}r(r-5) + (d-2)(r-\frac{d-1}{2}) \tag{13}$$

$$\leq (d-2)(d-1) - \frac{1}{2}t(t-5) + (d-2)(t-\frac{d-1}{2}) \tag{14}$$

$$\leq (d-2)(d-1) - \frac{1}{2}t(t-5). \tag{15}$$

Similary, we derive that of additions as follows.

$$\sum_{i=0}^{r-1} d_{i+1}(d_i - d_{i+1}+1) \tag{16}$$

$$= -(d_0 - d_r + r) + \sum_{i=0}^{r-1}(d_{i+1}+1)(d_i - d_{i+1}+1) \tag{17}$$

$$\leq -2r + (d-2)(d-1) - \frac{1}{2}t(t-5). \tag{18}$$

The computation in Step 2-2 costs at most $\frac{(d-1)(d+2t-5)}{4}$ multiplications, at most $\frac{(d-3)(d+2t-5)}{4}$ additions over F_{q^m} for the following reason. From Proposition A.1, for any $i \in [r]$, it costs

$$(1 + \deg_q G_i(x))(1 + \deg_q A_{i-1}(x)) \tag{19}$$

$$= (d_{i-1} - d_i + 1)(d - d_{i-1}) \tag{20}$$

multiplications and

$$\deg_q G_i(x) \deg_q A_{i-1}(x) + (1 + \deg_q A_{i-2}(x)) \tag{21}$$

$$= (d_{i-1} - d_i)(d - 1 - d_{i-1}) + (d - d_{i-2}) \tag{22}$$

additions over F_{q^m} to compute $A_i(x)$ from $A_{i-2}(x), A_{i-1}(x), G_i(x)$. When $i = 1$, no computation are needed since $A_i(x) = G_1(x)$. Thus some computations are needed only when $i = 2, \ldots, r$. It costs at most $2(d-t-1)(d-t-2)$ multiplications over F_{q^m} and at most $2t(d-t-2)$ additions over F_{q^m} to compute $A_2(x), \ldots, A_r(x)$. We derive an upper bound of the number of multiplications when $(t \geq) r \geq 2$ as follows.

$$\sum_{i=2}^{r}(d_{i-1} - d_i + 1)(d - d_{i-1}) \tag{23}$$

$$\leq \sum_{i=2}^{r}(d_{i-1} - d_i + 1)(d - d_{r-1}) \tag{24}$$

$$= (d_1 - d_r + r - 1)(d - d_{r-1}) \tag{25}$$

$$\leq (d - 2 - \frac{d-1}{2} + r - 1)(d - d_r - 1) \tag{26}$$

$$\leq \frac{d+2t-5}{2}\frac{d-1}{2} \tag{27}$$

$$\leq (2d - 2t - 4)\frac{d-1}{2} \tag{28}$$

$$\leq 2(d-t-2)(d-t-1). \tag{29}$$

The value of Eq.(29) is also an upper bound of the number of operations (=0) when $r = 1$. Similary, we derive that of additions when $(t \geq) r \geq 2$ as follows. Since $d \geq 5$,

$$\sum_{i=2}^{r}(d_{i-1} - d_i)(d - 1 - d_{i-1}) + (d - d_{i-2}) \tag{30}$$

$$\leq \sum_{i=2}^{r}(d_{i-1} - d_i)(d - 1 - d_{i-1}) + (d - 1 - d_{i-1}) \tag{31}$$

$$\leq (d - 1 - d_{r-1}) \sum_{i=2}^{r}(d_{i-1} - d_i + 1) \tag{32}$$

$$\leq (d-2-d_r)(d_1-d_r+r-1) \tag{33}$$
$$\leq (d-2-\frac{d-1}{2})(d-2-\frac{d-1}{2}+t-1) \tag{34}$$
$$\leq \frac{d-3}{2}\frac{d+2t-5}{2} \tag{35}$$
$$\leq \frac{d-3}{2}(2d-2t-4) \tag{36}$$
$$\leq 2t(d-t-2) \tag{37}$$

The value of Eq.(37) is also an upper bound of the number of operations (=0) when $r=1$. Thus, the upper bound of the number of operations in Step 2 is derived.

B. Step 3

The computation in Step 3 is constructed from the following Steps 3-1 and 3-2. The input is a linearized polynomials $\Lambda(x)$ The output is r ($\in [t]$) and r roots $E_1,\ldots,E_r \in \mathbb{F}_{q^m}$ of $\Lambda(x)$ which are linear independent over $\mathbb{F}_q$. We define r as the dimension of the linear space, which is the set of all roots of $\Lambda(x)$.

3-1 Compute $\Lambda(v_1),\ldots,\Lambda(v_m)$. It costs at most $(t+1)m$ multiplications over $\mathbb{F}_{q^m}$ and at most tm additions over $\mathbb{F}_{q^m}$.

3-2 We define $\Lambda \in \mathbb{F}_q^{m\times m}$ as

$$\boldsymbol{f}^m\left((\Lambda(v_1),\ldots,\Lambda(v_m))^\top\right) \in \mathbb{F}_{q^m}^m. \tag{38}$$

It holds that $\boldsymbol{U}\Lambda=\boldsymbol{0}$. Compute the full-rank matrix $\boldsymbol{U}\in\mathbb{F}_q^{r\times m}$ by Gaussian elimination over $\mathbb{F}_q$. For any $i\in[r]$, $E_i \coloneqq u_{i1}v_1+\cdots+u_{im}v_m$.

An upper bound of the number of four arithmetic operations over $\mathbb{F}_q$ is $m^2+m-1+\frac{2}{3}(m(m-1)(2m-1)-t(t-1)(2t-1))-(t-2)(m-t)-(t-1)(m^2-m-t^2+t)$.

Then $r=m-\mathrm{rank}\Lambda$ holds.

We derive an upper bound of the number of four arithemetic operations over $\mathbb{F}_q$ as follows. Step 3 is an algorithm to output r ($\in[t]$) roots $E_1,\ldots,E_r\in\mathbb{F}_{q^m}$ linearly independent on $\mathbb{F}_q$ from the input (a linearized polynomial $\Lambda(x)\in\mathbb{F}_{q^m}[x]$).

Proposition A.4: Let $r\in[t]$ be given. For any $i\in[r]$, for some $u_{i1},\ldots,u_{im}\in\mathbb{F}_q$, it holds that $E_i=u_{i1}v_1+\cdots+u_{im}v_m$. $E_1,\ldots,E_r\in\mathbb{F}_{q^m}$ are roots of a linearized polynomial $\Lambda(x)\in\mathbb{F}_{q^m}[x]$ and linearly independent on $\mathbb{F}_q$ if and only if $\boldsymbol{U}\Lambda=\boldsymbol{0}$ and $\mathrm{rank}(\boldsymbol{U})=r$.

Proof : $E_1,\ldots,E_r\in\mathbb{F}_{q^m}$ are linearly independent on $\mathbb{F}_q$ if and only if the rank of $r\times m$ matrix $\boldsymbol{U}$,whose (i,j) entry is u_{ij}, over $\mathbb{F}_q$ is $\min\{m,r\}=r$. This is because

$$(\forall \boldsymbol{a}=(a_1,\ldots,a_r)\in\mathbb{F}_q^r, \sum_{i\in[t]} a_iE_i=0 \Rightarrow \boldsymbol{a}=\boldsymbol{0}) \tag{39}$$
$$\Leftrightarrow (\forall \boldsymbol{a}\in\mathbb{F}_q^r, \sum_{i\in[t]} a_i \sum_{j\in[m]} u_{ij}v_j=0 \Rightarrow \boldsymbol{a}=\boldsymbol{0}) \tag{40}$$
$$\Leftrightarrow (\forall \boldsymbol{a}\in\mathbb{F}_q^r, \sum_{j\in[m]}\left(\sum_{i\in[t]} a_iu_{ij}\right)v_j=0 \Rightarrow \boldsymbol{a}=\boldsymbol{0}) \tag{41}$$
$$\Leftrightarrow (\forall \boldsymbol{a}\in\mathbb{F}_q^r, \forall j\in[m], \sum_{i\in[t]} a_iu_{ij}=0 \Rightarrow \boldsymbol{a}=\boldsymbol{0}) \tag{42}$$
$$\Leftrightarrow (\forall \boldsymbol{a}\in\mathbb{F}_q^r, \forall j\in[m], \sum_{i\in[t]} a_iu_{ij}=0 \Rightarrow \boldsymbol{a}=\boldsymbol{0}) \tag{43}$$
$$\Leftrightarrow (\forall \boldsymbol{a}\in\mathbb{F}_q^r, \boldsymbol{U}\boldsymbol{a}=\boldsymbol{0} \Rightarrow \boldsymbol{a}=\boldsymbol{0}) \tag{44}$$
$$\Leftrightarrow \mathrm{rank}(\boldsymbol{U})=\min\{m,r\}=r. \tag{45}$$

$E_1,\ldots,E_r$ are roots of a linearized polynomial $\Lambda(x)$ if and only if $\boldsymbol{U}\Lambda=\boldsymbol{0}$.

$$\forall i\in[r], \Lambda(E_i)=0 \tag{46}$$
$$\Leftrightarrow \forall i\in[r], \Lambda(u_{i1}v_1+\cdots+u_{im}v_m)=0 \tag{47}$$
$$\Leftrightarrow \forall i\in[r], u_{i1}\Lambda(v_1)+\cdots+u_{im}\Lambda(v_m)=0 \tag{48}$$
$$\Leftrightarrow \begin{pmatrix} u_{11} & \ldots & u_{1m}\\ \vdots & \ddots & \vdots \\ u_{r1} & \ldots & u_{rm}\end{pmatrix}\begin{pmatrix}\Lambda(v_1)\\ \vdots \\ \Lambda(v_m)\end{pmatrix}=\boldsymbol{0}(\in\mathbb{F}_{q^m}^r) \tag{49}$$
$$\Leftrightarrow \boldsymbol{U}\Lambda\boldsymbol{v}=\boldsymbol{0}(\in\mathbb{F}_{q^m}^r) \tag{50}$$
$$\Leftrightarrow \boldsymbol{U}\Lambda=\boldsymbol{0}(\in\mathbb{F}_q^{r\times m}). \tag{51}$$

Thus $E_1,\ldots,E_r$ are roots of the linearized polynomial $\Lambda(x)$ and are linearly independent over $\mathbb{F}_q$ if and only if $\boldsymbol{U}\Lambda=\boldsymbol{0}$ and $\mathrm{rank}(\boldsymbol{U})=r$. □

Thus if $\Lambda(x)$ is input, then compute $(\Lambda(v_1),\ldots,\Lambda(v_m))^\top\in\mathbb{F}_{q^m}^m$, compute $\boldsymbol{U}\in\mathbb{F}_q^{r\times m}$ which satisfies the above conditions by Gaussian elimination over $\mathbb{F}_q$, and compute $E_i=u_{i1}v_1+\cdots+u_{im}v_m$ for all $i\in[r]$.

The matrix $\boldsymbol{U}$ can be computed by *Algorithm 1* under the following assumption : λ_{ij} denotes (i,j)-th entry of Λ. For simplicity, the reduced column echelon form of Λ by *Algorithm 1* is assumed to be as follows[4].

$$\Lambda'=\begin{pmatrix}\boldsymbol{I}_{(m-r)\times(m-r)} & \boldsymbol{0}_{(m-r)\times r}\\ \Lambda'_{[m-r+1,m],[m-r]} & \boldsymbol{0}_{r\times r}\end{pmatrix}, \tag{52}$$

where $\boldsymbol{I}_{(m-r)\times(m-r)}$ is the $(m-r)\times(m-r)$ identity matrix. $\Lambda'_{[m-r+1,m],[m-r]}\in\mathbb{F}_q^{r\times(m-r)}$ is some matrix. Then we define $\boldsymbol{U}$ as $\left(\Lambda'_{[m-r+1,m],[m-r]},\boldsymbol{I}_{r\times r}\right)$.

Algorithm 1 Step 3

Require: Λ
Ensure: $\boldsymbol{U}$
1: **for** $k\in[m-r]$ **do**
2: Compute λ_{kk}^{-1}.
3: **for** $j\in[k+1,m]$ **do**
4: Compute $\lambda_{kj}\lambda_{kk}^{-1}$.
5: **for** $i\in[k+1,m]$ **do**
6: Compute $\lambda_{ij}-\left(\lambda_{kj}\lambda_{kk}^{-1}\right)\lambda_{ik}$.
7: **end for**
8: **end for**
9: **end for**
10: $\boldsymbol{U}\coloneqq\left(\Lambda_{[m-r+1,m],[m-r]},\boldsymbol{I}_{r\times r}\right)$

[4]If you need it, you can change $\boldsymbol{U}$ to the matrix in Eq.(52) by the permutation of the indices of row numbers, and perform the subsequent operations to compute the matrix corresponding to $\boldsymbol{U}$. Then you can define $\boldsymbol{U}$ as this matrix.

First, we enumerate the number of operations in Step 3-1. For $i' \in [m]$, it costs $t+1$ multiplications and t additions over $\mathbb{F}_{q^m}$ to compute $\Lambda(v_{i'})$ from the input $\Lambda(x) = \sum_{j\in[0,t]} \lambda_j x^{q^j} \in \mathbb{F}_{q^m}[x]$.

$$\Lambda(v_{i'}) = \sum_{j\in[0,t]} \lambda_j v_{i'}^{q^j} = \sum_{j\in[0,t]} \lambda_j v_1^{q^{i'+j-1}} = \sum_{j\in[0,t]} \lambda_j v_{i'+j}. \tag{53}$$

Since the value of the normal basis $v_{i'}, \ldots, v_{i'+t}$ are already computed, it costs $t+1$ multiplications and t additions over $\mathbb{F}_{q^m}$ to compute $\sum_{j\in[0,t]} \lambda_j v_{i'+j}$.

Next, we enumerate the number of operations in Step 3-2. The number of four arithmetic operations over $\mathbb{F}_q$ from the 1-th row to 9-th row in *Algorithm 3* is as follows.

$$\sum_{k=1}^{m-r} \left(1 + \sum_{j=k+1}^{m} \left(1 + \sum_{i=k+1}^{m} 2\right)\right) \tag{54}$$

$$= \frac{1}{3}(m(m-1)(2m-1) - r(r-1)(2r-1)) + (m-r) + \frac{1}{2}(m(m-1) - r(r-1)). \tag{55}$$

Therefore we derive an upper bound of the number of all four arithemetic operations over $\mathbb{F}_q$ as follows.

$$\frac{1}{3}(m(m-1)(2m-1) - r(r-1)(2r-1)) + (m-r) + \frac{1}{2}(m(m-1) - r(r-1)) \tag{56}$$

$$\leq \frac{2}{3}(m(m-1)(2m-1) - r(r-1)(2r-1)) + (2-r)(m-r) + (1-r)(m(m-1) - r(r-1)) \tag{57}$$

$$\leq \frac{2}{3}(m(m-1)(2m-1) - t(t-1)(2t-1)) + (2-t)(m-t) + (1-t)(m(m-1) - t(t-1)). \tag{58}$$