Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5615041, 10 pages
https://doi.org/10.1155/2022/5615041

Research Article

Multiple Dimensional Encoding/Modulation Shift-and-Addition Design for Distributed Systems

Mingjun Dai,[1] Chanting Zhang,[1] Zhaoyan Hong,[2] Paweł Wawrzyński,[3] Tomasz Trzciński,[3] and Xiaohui Lin[1]

[1]*College of Electronic and Information Engineering, Shenzhen University, 518000 Shenzhen, China*
[2]*College of Mathematics and Statistics, Shenzhen University, 518000 Shenzhen, China*
[3]*Institute of Computer Science, Warsaw University of Technology, 00-665 Warsaw, Poland*

Correspondence should be addressed to Zhaoyan Hong; 1546623526@qq.com

Received 20 July 2022; Accepted 27 August 2022; Published 15 September 2022

Academic Editor: Jun Li

In distributed computing/storage/machine learning system, the method of encoding and decoding combing shift-and-addition (SA) and zigzag decoding (ZD) is proposed to solve the problem of high computational complexity. However, in each encoded packet, one element takes part in the encoding only once, so the obtained overhead is extremely high. In this work, based on the idea of multidimensional encoding/modulation, we propose to employ one element of the encoding process multiple times when constructing one encoded packet based on the Cauchy matrix, thereby leveraging the favourable properties of the code based on Cauchy matrix. The overhead is reduced from square to logarithmic in certain parameters. Compared with the overhead of the existing square computational complexity, it is greatly reduced.

1. Introduction

In the era of big data [1], the amount of data is growing at a doubling rate annually. The way of data processing has been shifted from the centralized data processing to the distributed data processing. However, in distributed applications, not all devices are reliable. Some devices may fail to work, or the performance of the devices is not consistent. In any task of data processing, there will be some unreliable devices whose computing speed is slower than the average speed, which are called stragglers [2, 3]. For example, in a data center of Facebook, more than 100 nodes may fail per day [4, 5]. The completion time of data processing is constrained by the slowest working node. Therefore, how to deal with stragglers becomes a challenge for data processing. To solve this problem, network coding techniques have been developed, and the code with combination property (CP) is proposed: k original packets are encoded into n packets, where $n > k$, and any k out of these n packets are able to recover the original data. The code with CP has been widely used in distributed systems, including distributed storage (DS) [6–10], distributed computing (DC) [11–16], and distributed machine learning [17].

In distributed systems, linear code is adopted in most of the coding technologies, but linear code involves a lot of multiplication and division operations, which greatly increase the complexity of coding and decoding. For the sake of low computation complexity, a kind of CP-ZD code [18–20] with CP is proposed, which combines shift-and-addition (SA) and zigzag decoding (ZD) [21]. However, for the case where one element takes part in the encoding only once when constructing an encoded packet, the overhead is as high as square of the parameters' number (n or k).

Multidimensional encoding/modulation promises high data rate [18, 19], which has been used as promising technique in communication [20] and distributed systems [22, 23]. As a result, to further reduce the overhead, based on the idea of multidimensional encoding/modulation, we propose the idea of one element taking part in the encoding process multiple times when constructing one encoded packet. Using the properties of the code based on Cauchy matrix in finite field, we design a framework of one element taking part in the encoding process multiple times for constructing an encoded packet. The overhead of this coding framework reduces from square to logarithmic with respect to the parameter. Specifically, the idea that one element takes part in the encoding only once in each encoded packet can be interpreted as each source packet being treated as an element and occurring at most once in a coded packet. Similarly, the idea that one element takes part in the encoding process multiple times when constructing one encoded packet is that each source packet occurs multiple times in an encoded packet, which is added to its own multiple distinct shifts.

2. Preliminary

2.1. Definition of Cauchy Matrix. Given $x_1, x_2, \cdots x_n$, y_1, $y_2, \cdots y_n$, let $c_{(i,j)} = 1/(x_i + y_j)(1 \le i, j \le n)$, then the matrix $C = (c_{i,j})$ is called the Cauchy matrix [24], and its determinant is as follows:

$$\det(C) = \frac{\prod_{1 \le i < j \le n} (x_j - x_i)\left(y_j - y_i\right)}{\prod_{i=1}^{n} \prod_{j=1}^{n} \left(x_i + y_j\right)}. \tag{1}$$

Similarly, a Cauchy matrix over a finite field is defined as follows: let X and Y be two sets of elements in a finite field. Among them, $X = \{x_1, x_2, \cdots, x_p\}$ and $Y = \{y_1, y_2, \cdots, y_q\}$. If for $\forall i \in \{1, 2, \cdots, p\}$, $\forall j \in \{1, 2, \cdots, q\}$, the following is satisfied:

(1) $x_i + y_j \ne 0$

(2) $\forall i, j \in \{1, 2, \cdots, p\}$, $i \ne j$, $x_i \ne x_j$

(3) $\forall i, j \in \{1, 2, \cdots, q\}, i \ne j, y_i \ne y_j$

Then, the following matrix is called a Cauchy matrix over a finite field:

$$G = \begin{bmatrix} \frac{1}{x_1 + y_1} & \frac{1}{x_1 + y_2} & \cdots & \frac{1}{x_1 + y_q} \\ \frac{1}{x_2 + y_1} & \frac{1}{x_2 + y_2} & \cdots & \frac{1}{x_2 + y_q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_p + y_1} & \frac{1}{x_p + y_2} & \cdots & \frac{1}{x_p + y_q} \end{bmatrix}. \tag{2}$$

It is straightforward to obtain the following theorem from the construction rules of the Cauchy matrix:

Theorem 1. *When G is a Cauchy matrix, any square submatrix G_l of G is nonsingular, where l indicates the number of rows and columns of the submatrix G_l $(1 \le l \le \min(p, q))$; then,*

$$\det(G_l) = \frac{\prod_{1 \le i < j \le l} (x_j - x_i)\left(y_j - y_i\right)}{\prod_{i=1}^{l} \prod_{j=1}^{l} \left(x_i + y_j\right)} \ne 0. \tag{3}$$

In other words, every submatrix of the Cauchy matrix is invertible.

2.2. The Arithmetic Operation in Finite Fields. Finite field is a field with a finite number of elements, for example, $GF(2^\omega)$ represents a finite field containing 2^ω elements. Before describing the arithmetic operation in finite fields, we briefly introduce the concept of the primitive polynomial.

The primitive polynomial is essentially a polynomial that cannot be factored. When a finite field determines its primitive polynomials, the arithmetic operations in that field are also determined. In general, the primitive polynomial of a field can be obtained by looking up the table, and the primitive polynomial of a field is not unique. Take the finite field $GF(2^3)$ as an example, there is more than one primitive polynomial over $GF(2^3)$, and the most common primitive polynomial is $q(z) = z^3 + z + 1$. Table 1 shows some of the primitive polynomials [25] present in $GF(2^\omega)$.

The addition and subtraction operation [22] of finite fields are the XOR operation in polynomial calculation. The rule for adding and subtracting is to XOR coefficients of the same order in two polynomials, and there is no difference between the two operations, such as $(z^2 + z) + (z + 1) = z^2 + 1$, $(z^2 + z) - (z + 1) = z^2 + 1$. At present, the multiplication and division operations [23] of finite fields usually count on the look-up tables. Each field has positive and negative tables, which are denoted as gf log and gfi log, respectively, on the $GF(2^\omega)$ field. Taking $GF(2^3)$ as an example, its table gf log and gfi log are generated as shown in Table 2 [16]:

If the multiplication and division operations are performed on the $GF(2^3)$ field, as shown in Table 2, the multiplication operation is as follows:

$$\begin{aligned} 2 \times 3 &= gfi \log [gf \log [2] + gf \log [3]] \\ &= gfi \log [1 + 3] = gfi \log [4] = 6, \end{aligned} \tag{4}$$

and the division operation is as follows:

$$\begin{aligned} 1 \div 5 &= gfi \log [gf \log [1] - gf \log [5]] \\ &= gfi \log [0 - 6] = gfi \log [1] = 2. \end{aligned} \tag{5}$$

2.3. Transformation from Field $GF(2^\omega)$ to Field $GF(2)[z]/q(z)$. Field $GF(2^\omega)$ is constructed by finding a primitive polynomial of ω degrees on $GF(2)$ and then enumerating elements (in polynomial form) by using the generating element z. The addition in this field is performed using polynomial addition, and multiplication is performed using polynomial multiplication and modulo the result with respect to $q(z)$, such field $GF(2^\omega)$ can be written as $GF(2^\omega) = GF(2)[z]/q(z)$ [25], which can also be said that field $GF(2^\omega)$ and field $GF(2^\omega) = GF(2)[z]/q(z)$ are isomorphic [26].

Table 1: Table of common primitive polynomials.

Finite field	Primitive polynomial
$GF(2^2)$	z^2+z+1
$GF(2^3)$	z^3+z+1
$GF(2^4)$	z^4+z+1
$GF(2^8)$	$z^8+z^4+z^3+z^2+1$
$GF(2^{16})$	$z^{16}+z^{12}+z^3+z+1$

Table 2: gf log and gfi log for $GF(2^3)$.

i	gf log $[i]$	gfi log $[i]$
0	-	1
1	0	2
2	1	4
3	3	3
4	2	6
5	6	7
6	4	5
7	5	-

The conversion rule for field $GF(2^\omega)$ to field $GF(2)[z]/q(z)$ [25] is the conversion of numerical form to polynomial form. Taking $GF(2^\omega)$ as an example, the implementation steps are as follows:

Step 1. Initialize the set as $\{0,1,z\}$.

Step 2. Multiply the last element of the set by z, such as z, and modulo the result with respect to $q(z)$ if the resulting degree is greater than or equal to ω.

Step 3. Continue Step 2 until there are 2^ω elements in the set, at which point the last element is multiplied by z and modulo $q(z)$, resulting in a value of 1.

To better understand the above steps, let us elaborate on a simple example:

Example 1. Suppose $\omega=2$; then, the original polynomial is $q(z)=z^2+z+1$; to construct $GF(2^2)=GF(4)$, we initialize the set as $\{0,1,z\}$, so the next element is z^2; since the degree of the element is 2, modulo it with respect to $q(z)=z^2+z+1$, which resulting in $z+1$. Therefore, four elements are generated: $\{0,1,z,z+1\}$, and the corresponding numerical forms are $\{0,1,2,3\}$, which are shown in Table 3. If we continue, we can get the following:

$$(z+1)z \bmod q(z) = (z^2+z) \bmod (z^2+z+1) = 1. \quad (6)$$

According to Step 3, we can end the enumeration.

2.4. Mathematical Model. We want to construct (n,k) code that possesses the (n,k) CP. This section adopts the method in reference [27]. We represent each packet as a polynomial of $GF(2^\omega)$, where a number is denoted by several bits within this packet. Source packet s_i is represented with the polynomial form, as shown in Formula (7), $i\in K \triangleq \{1,2,\cdots,k\}$,

Table 3: Transformation of $GF(4)$ from field $GF(2^\omega)$ to field $GF(2^\omega)=GF(2)[z]/q(z)$.

Generating element	Polynomial form	Numerical form
0	0	0
z^0	1	1
z^1	z	2
z^2	$z+1$	3

$$s_i(z) \triangleq s_{i,1}+s_{i,2}z+s_{i,3}z^2+\cdots+s_{i,L}z^{L-1}, \quad (7)$$

where L indicates the length of the source packet and z indicates the right shift by one bit.

For $i\in K$, the i-th encoded packet can be expressed as $c_i(z)=s_i(z)$. Let $m\triangleq n-k$ denote the number of parity packets. For $i\in M\triangleq\{1,2,\cdots,m\}$, the polynomial form of the i-th parity packet is as follows:

$$c_{k+i}(z) \triangleq \alpha_{i,1}(z)s_1(z)+\alpha_{i,2}(z)s_2(z)+\cdots+\alpha_{i,k}(z)s_k(z), \quad (8)$$

where $\alpha_{i,j}(z)\triangleq z^{T_{ij}}$, $i\in M, j\in K$.

Combined with the systematic packets and parity packets, the final coding expression is shown in the following formula:

$$c(z)=A(z)s(z), \quad (9)$$

where $c(z)\triangleq(c_1(z),c_2(z),\cdots,c_n(z))$, $s(z)\triangleq(s_1(z),s_2(z),\cdots,s_k(z))$, and

$$A(z)\triangleq\begin{bmatrix} I_K(z) \\ T(z)\end{bmatrix}, \quad (10)$$

which is a matrix with dimension of $n\times k$, $I_k(z)$ is a $k\times k$ identity matrix, and $T(z)$ is a $m\times k$ shift matrix,

$$T(z)\triangleq\begin{bmatrix} z^{T_{11}}\, z^{T_{12}}\, z^{T_{13}} \dots z^{T_{1k}} \\ z^{T_{21}}\, z^{T_{22}}\, z^{T_{23}} \dots z^{T_{2k}} \\ z^{T_{31}}\, z^{T_{32}}\, z^{T_{33}} \dots z^{T_{3k}} \\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ z^{T_{m1}}\, z^{T_{m2}}\, z^{T_{m3}} \dots z^{T_{mk}} \end{bmatrix}. \quad (11)$$

The exponent of the element in (z) is indicated by T, whose actual meaning is the shifted bits of packets.

3. Encoding Design

The encoding framework that one element takes part in the encoding process multiple times when constructing one encoded packet based on Cauchy matrix is mainly constructed in three steps, and the detailed rules are as follows:

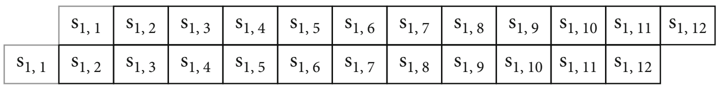
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,4}$ $s_{1,5}$ $s_{1,6}$ $s_{1,7}$ $s_{1,8}$ $s_{1,9}$ $s_{1,10}$ $s_{1,11}$ $s_{1,12}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,4}$ $s_{1,5}$ $s_{1,6}$ $s_{1,7}$ $s_{1,8}$ $s_{1,9}$ $s_{1,10}$ $s_{1,11}$ $s_{1,12}$

FIGURE 1: The form of elements taking part in encoding process multiple times

Step 1. Determine the size of the finite field and the dimension of the Cauchy matrix according to the relation of coding parameters (n, k).

According to the relation of (n, k), when $k < n \leq 2^{\omega}$ and ω is a positive integer, we can determine the size of the finite field to be $GF(2^{\omega})$, where $\omega = \lceil \log_2 n \rceil$ and the sign $\lceil \, \rceil$ indicates an integer ceiling function. In finite field $GF(2^{\omega})$, the dimension of Cauchy matrix over $GF(2^{\omega})$ is determined as $m \times k$ according to the size of m and k. The specific construction process of the corresponding Cauchy matrix (i.e., the coding matrix) is to determine the element set of X and Y first. The elements of X could be any m elements in $GF(2^{\omega})$, $X = \{x_1, x_2, \cdots, x_m\}$, which correspond to m parity packets. The elements of Y are any k elements in the $2^{\omega} - m$ elements in $GF(2^{\omega})$ finite field, $Y = \{y_1, y_2, \cdots, y_k\}$, which correspond to k parity packets. According to the construction rules of Cauchy matrix in the finite field, the following relationships should be met between the element sets of X and Y:

If for $\forall i \in \{1, 2, \cdots, m\}$, $\forall j \in \{1, 2, \cdots, k\}$, the following is satisfied:

(1) $x_i + y_j \neq 0$

(2) $\forall i, j \in \{1, 2, \cdots, m\}, i \neq j, x_i \neq x_j$

(3) $\forall i, j \in \{1, 2, \cdots, k\}, i \neq j, y_i \neq y_j$

Then, the elements in X and Y are different, and a coding matrix G with dimension $m \times k$ can be constructed as follows:

$$G = \begin{bmatrix} \frac{1}{x_1 + y_1} & \frac{1}{x_1 + y_2} & \cdots & \frac{1}{x_1 + y_k} \\ \frac{1}{x_2 + y_1} & \frac{1}{x_2 + y_2} & \cdots & \frac{1}{x_2 + y_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_m + y_1} & \frac{1}{x_m + y_2} & \cdots & \frac{1}{x_m + y_k} \end{bmatrix}. \tag{12}$$

Step 2. Convert the numeric form of elements in the coding matrix to polynomial form.

Through the arithmetic operation in finite fields and the transformation from field $GF(2^{\omega})$ to field $GF(2)[z]/q(z)$ in Section 2.3, the matrix G is transformed into the polynomial form. Each element of G can be uniquely represented by a polynomial, that is, $G \Rightarrow G(z)$.

In particular, the exponent of z of each polynomial represents the size of the bit-shifting. For example, $z^2 + z$ represents that a systematic package is shifted to the right by 2 and 1 bits, respectively, and then added over. For example, the shift value of the source packet s_1 of length $L = 12$ is $z^2 + z$, as shown in Figure 1.

Step 3. Determine the shift matrix combined with the systematic code.

Combined with the systematic code, a coding matrix

$$A(z) \triangleq \begin{bmatrix} I_k(z) \\ G(z) \end{bmatrix} \tag{13}$$

with dimension $n \times k$ is obtained through vertical connection.

In order to better understand the design rules that one element takes part in the encoding process multiple times, we will walk through the coding steps with a concrete example.

Example 2. If $(n, k) = (8, 5)$, then the system has k systematic packages and m parity packages, where $k = 5$ and $m = n - k = 3$.

Step 1. We have $k < n \leq 2^{\omega} \Rightarrow 5 < 8 \leq 2^3$, $\omega = \lceil \log_2 8 \rceil = 3$, so we can determine that the size of the finite field is $GF(2^3)$. For the finite field $GF(2^3)$, the commonly used polynomial $q(z) = z^3 + z + 1$ is chosen as the primitive polynomial. In this case, X and Y take 3 and 5 elements, respectively, and the elements of X and Y are different. Taking $X = \{5, 6, 7\}$ and $Y = \{0, 1, 2, 3, 4\}$ as an example, the coding matrix is as follows:

$$G = \begin{bmatrix} \frac{1}{5+0} & \frac{1}{5+1} & \frac{1}{5+2} & \frac{1}{5+3} & \frac{1}{5+4} \\ \frac{1}{6+0} & \frac{1}{6+1} & \frac{1}{6+2} & \frac{1}{6+3} & \frac{1}{6+4} \\ \frac{1}{7+0} & \frac{1}{7+1} & \frac{1}{7+2} & \frac{1}{7+3} & \frac{1}{7+4} \end{bmatrix}. \tag{14}$$

Step 2. Through the arithmetic operations in the finite field, the addition operation is converted to the XOR operation, for example, $5 \oplus 1 \Rightarrow 101 \oplus 1 = 100 = 4$. Multiplication and division can be calculated by looking up tables. For example, Table 2 lists the positive and negative tables of $GF(2^3)$. Based on the above, we can convert the matrix G into the following form:

$$G = \begin{bmatrix} 2 & 7 & 4 & 3 & 1 \\ 3 & 4 & 7 & 2 & 5 \\ 4 & 3 & 2 & 7 & 6 \end{bmatrix}. \tag{15}$$

Through the transformation from field $GF(2^\omega)$ to field $GF(2)[z]/q(z)$ in Section 2.3, the elements of coding matrix G are expressed by polynomials one by one to obtain the coding matrix $G(z)$:

$$G(z)=\begin{bmatrix} z & z^2+z+1 & z^2 & z+1 & 1 \\ z+1 & z^2 & z^2+z+1 & z & z^2+1 \\ z^2 & z+1 & z & z^2+z+1 & z^2+z \end{bmatrix}. \tag{16}$$

Step 3. By vertically concatenating 5 systematic codes, we can obtain the coding matrix $A(z)$:

$$A(z)=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ z & z^2+z+1 & z^2 & z+1 & 1 \\ z+1 & z^2 & z^2+z+1 & z & z^2+1 \\ z^2 & z+1 & z & z^2+z+1 & z^2+z \end{bmatrix}. \tag{17}$$

4. Properties of the Code Based on Cauchy Matrix

4.1. CP Property. Before we prove the CP property of the code, we will introduce the properties and lemmas mentioned in the proof.

Isomorphism property: the mathematical idea of isomorphism is to establish a one-to-one mapping of two sets that have the same properties associated with operations. For example, assuming that the sets B and $\bar{B}$ of algebraic operations are isomorphic, if one set B has a property that is only relevant to the algebraic operations of this set, then the other set $\bar{B}$ has exactly similar properties [28].

Lemma 2. *Any square submatrix of $G(z)$ is invertible.*

Proof. According to Section 3, matrix G is a Cauchy matrix, and the transformation from G to its polynomial form $G(z)$ is equivalent to the transformation from field $GF(2^\omega)$ to field $GF(2)[z]/q(z)$, as shown in Section 2.3. In addition, since field $GF(2^\omega)$ and field $GF(2)[z]/q(z)$ are isomorphic [17] and the isomorphism property indicates that reversibility in a field will remain reversibility in the isomorphism field, the reversibility of G in field $GF(2^\omega)$ will be mapped to that of $G(z)$ in field $GF(2)[z]/q(z)$. Theorem 1 says that any submatrix of G is invertible, so any submatrix of $G(z)$ is also invertible. □

CP property: Any k out of n encoded packets are able to recover the information of the original k packets.

Proof. First, we use a mathematical model to solve the above coding and decoding problems. The Cauchy matrix G and the coding matrix A are constructed from the previous coding design in Section 3, where

$$A \triangleq \begin{bmatrix} I_k \\ G \end{bmatrix}, \tag{18}$$

and I_k represents the matrix

$$I_k=\begin{bmatrix} 0 & -\infty & \cdots & -\infty \\ -\infty & 0 & \cdots & -\infty \\ \vdots & \vdots & \ddots & \vdots \\ -\infty & -\infty & \cdots & 0 \end{bmatrix} \tag{19}$$

with the dimension of $k \times k$. For example,

$$I_3=\begin{bmatrix} 0 & -\infty & -\infty \\ -\infty & 0 & -\infty \\ -\infty & -\infty & 0 \end{bmatrix}, \tag{20}$$

where element 0 means that the source packet involved in encoding shifts 0 bit, which means nonshift, and element $-\infty$ means that the source packet does not participate in encoding.

According to the mathematical model of Section 2.4, in the finite field $GF(2^\omega)$, the matrix above is represented by polynomial form of z, where z is the radix (assuming that the modulus of z is greater than 1), and every element of matrix A is raised to a proper power and then mod the original polynomial of the finite field. This process is actually a transformation from field $GF(2^\omega)$ to field $GF(2)[z]/q(z)$. For example, the identity matrix I_k to $I_k(z)$ applies the above transformation to

$$\begin{aligned} I_k=\begin{bmatrix} 0 & -\infty & \cdots & -\infty \\ -\infty & 0 & \cdots & -\infty \\ \vdots & \vdots & \ddots & \vdots \\ -\infty & -\infty & \cdots & 0 \end{bmatrix} &\Rightarrow I_k(z) \\ =\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}&, \end{aligned} \tag{21}$$

where element 1 indicates that the source packet participating in the encoding is not shifted, and element 0 indicates that the source packet is not participating in the encoding. The transformation process of matrix G to $G(z)$ is shown in Section 2.3. At this point, the polynomial form of the coding matrix A is

$$A(z) \triangleq \begin{bmatrix} I_k(z) \\ G(z) \end{bmatrix}, \tag{22}$$

which is a matrix with a size of $n \times k$.

Based on the polynomial form of z above and the mathematical model in Section 2.4, the polynomial $c(z)$ represents the encoded packet and the polynomial $s(z)$ represents the systematic packet. The coding relationship can be expressed as $c(z) = A(z)s(z)$. Take any k packets from n coded packets, that is, extract k lines at the same position from encoded packet $c(z)$ and form $c_k(z)$, corresponding to take k lines at the same position from $A(z)$ and form $k \times k$ matrix $A_k(z)$.The encoding relationship can be expressed by the expression $c_k(z) = A_k(z)s(z)$.

Based on the above model, satisfying CP property is equivalent to the invertibility of $A_k(z)$. We claim that $A_k(z)$ is invertible due to the following reasons:

In

$$A(z) \triangleq \begin{bmatrix} I_k(z) \\ G(z) \end{bmatrix}, \tag{23}$$

take any k rows from $A(z)$ and form $k \times k$ matrix $A_k(z)$. In this case, $A_k(z)$ to be decoded can be formed in two ways: First, $A_k(z)$ does not contain any rows of $I_k(z)$; from Lemma 2 above, we know that every square submatrix of $G(z)$ is invertible, so $A_k(z)$ is invertible. Second, $A_k(z)$ is composed of α rows in $I_k(z)$ and β rows in $G(z)$, where $I_\alpha(z)$ represents α rows in $I_k(z)$ and $G_\beta(z)$ represents β rows in $G(z)$, and $\alpha + \beta = k$. Since the matrix $I_\alpha(z)$ is a known systematic packet, substituting it into $G_\beta(z)$ is equivalent to deleting α columns from it. The deleted $G_\beta(z)$ is equivalent to extracting the $\beta \times \beta$ submatrix from $G(z)$, where $k - \alpha = \beta$, as shown in the following Example 3. Similarly, Lemma 2 shows that any square submatrix of $G(z)$ is invertible, so $A_k(z)$ is also invertible.

To sum up, $A(z)$ is invertible, so this coding framework can meet the CP property. □

Example 3. Following Example 2, if $(n, k) = (8, 5)$, the system has 5 systematic packets, m parity packets, where $m = n - k = 3$. The matrix $G(z)$ has been constructed as

$$G(z) = \begin{bmatrix} z & z^2+z+1 & z^2 & z+1 & 1 \\ z+1 & z^2 & z^2+z+1 & z & z^2+1 \\ z^2 & z+1 & z & z^2+z+1 & z^2+z \end{bmatrix}. \tag{24}$$

Table 4: Success/failure for different (n, k) parameters.

(n, k)	Success	Failure	Total
$(5, 3)$	8	2	10
$(8, 5)$	46	9	56
$(12, 8)$	405	90	495
$(15, 10)$	2408	595	3003

Combining the identity matrix $I_5(z)$,

$$A(z) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ z & z^2+z+1 & z^2 & z+1 & 1 \\ z+1 & z^2 & z^2+z+1 & z & z^2+1 \\ z^2 & z+1 & z & z^2+z+1 & z^2+z \end{bmatrix}. \tag{25}$$

(z) is composed of some parts of $I_5(z)$ and $G(z)$, let

$$A_5(z) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ z & z^2+z+1 & z^2 & z+1 & 1 \\ z+1 & z^2 & z^2+z+1 & z & z^2+1 \end{bmatrix}, \tag{26}$$

and then, it can be seen from the matrix $A_5(z)$ that the known systematic packets are s_1, s_2, s_3, so $A_5(z)$ can be simplified as

$$A_5(z)' = \begin{bmatrix} z+1 & 1 \\ z & z^2+1 \end{bmatrix}, \tag{27}$$

which is equivalent to the 2×2 submatrix of $G(z)$. According to Lemma 2, any submatrix of $G(z)$ is invertible, so $A_5(z)'$ is invertible.

4.2. Zigzag Decodability (ZD) Property.

4.2. Zigzag Decodability (ZD) Property. The ZD property of one element which takes part in the encoding process multiple times when constructing one encoded packet can be proved by experiment that it cannot be fully zigzag decodable. Starting from the experiment, we set several groups of parameters to obtain the probability of zigzag decodability of one element which takes part in the encoding process

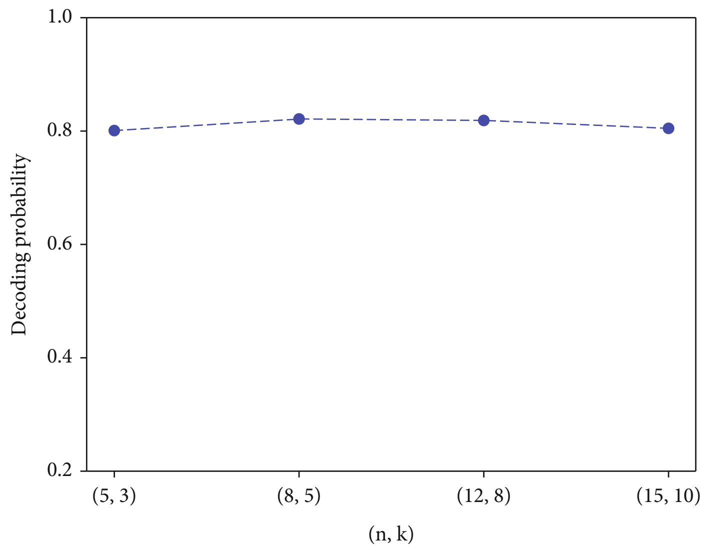

FIGURE 2: Decoding probability for different (n, k) parameters.

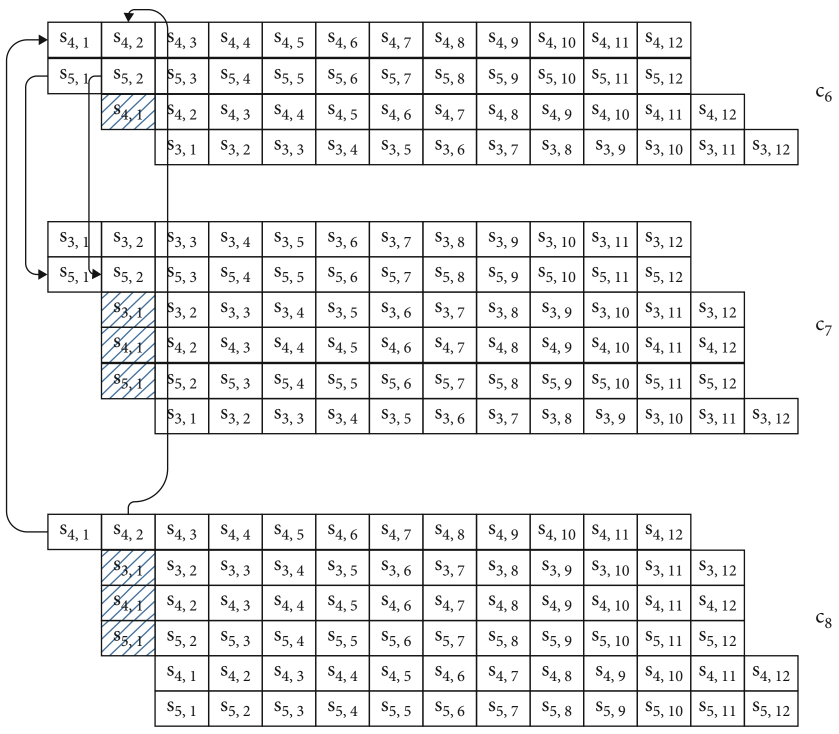

FIGURE 3: An example of able to perform ZD.

multiple times. Through experimental simulation, we set several sets of parameters (n, k) as $(5, 3)$, $(8, 5)$, $(12, 8)$, and $(15, 10)$, respectively, and the conditions of zigzag decodable and not are shown in Table 4.

It can be seen from Figure 2 that the encoding framework that one element takes part in the encoding process multiple times when constructing one encoded packet based on the Cauchy matrix has about 80% probability of ZD decoding, which means that there is about 20% probability that zigzag decoding will not be possible. The following two examples illustrate the case that the encoding framework is able or unable to perform zigzag decoding.

Example 4. Following Example 2 where $(n, k) = (8, 5)$, we assume that there are encoded packets c_1, c_2, c_6, c_7, c_8, where $c_1 = s_1$ and $c_2 = s_2$. Based on the above, the remaining problem is to use the encoded packets c_6, c_7, c_8 to decode the three source packets s_3, s_4, s_5. The corresponding coding matrix is as follows:

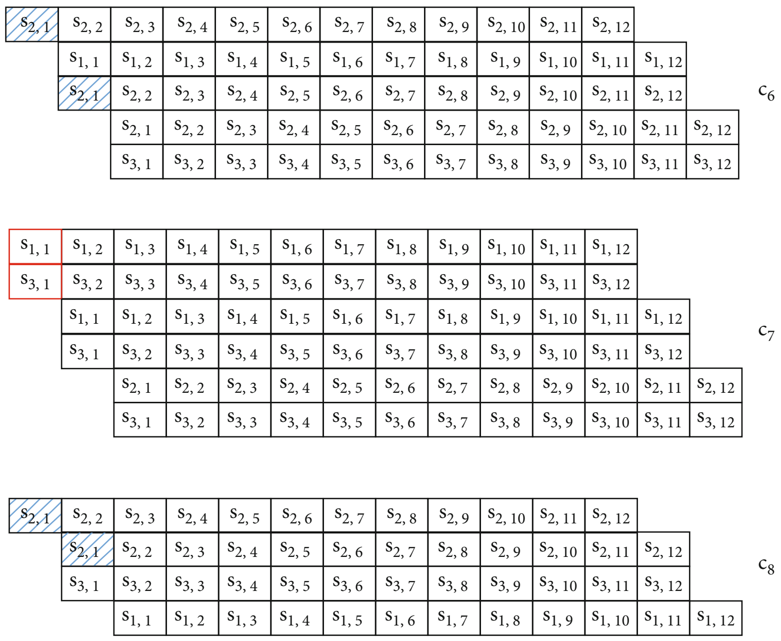
FIGURE 4: An example of failure in ZD.

$$G = \begin{bmatrix} z^2 & z+1 & 1 \\ z^2+z+1 & z & z^2+1 \\ z & z^2+z+1 & z^2+z \end{bmatrix}. \quad (28)$$

Assuming that the length of the encoded packets is $L = 12$, we can obtain the decoding diagram of Figure 3 through shift-and-addition encoding by row. In this condition, the coding framework can be zigzagged: $s_{4,1}$ can be obtained directly from the first exposed bit of c_8. Substitute $s_{4,1}$ into the first bit of c_6 to obtain $s_{5,1}$ through shift-and-addition, and then, substitute $s_{5,1}$ into c_7 to obtain $s_{3,1}$. In this manner, the first bit of each source packet has been obtained. The decoding of the second bit of each source packet is similar to the decoding of the first bit. Substitute $s_{3,1}, s_{4,1}, s_{5,1}$ into c_6, c_7, c_8, respectively, so that $s_{4,2}$ can be obtained. Substitute $s_{4,2}$ into the second bit of c_6, and $s_{5,2}$ can be obtained by XOR operation. Finally, substitute $s_{5,2}$ into c_7, and then, $s_{3,2}$ can be obtained.

Example 5. Following Example 2 where $(n, k) = (8, 5)$, we assume that there are encoded packets c_4, c_5, c_6, c_7, c_8, where $c_4 = s_4$ and $c_5 = s_5$.Therefore, the remaining problem is to use the encoded packets c_6, c_7, c_8 to decode the three source packets s_1, s_2, s_3 to form the encoding matrix,

$$G = \begin{bmatrix} z & z^2+z+1 & z^2 \\ z+1 & z^2 & z^2+z+1 \\ z^2 & z+1 & z \end{bmatrix}. \quad (29)$$

Assuming the length of the encoded packets is $L = 12$, through shift-and-addition encoding by row, we cannot perform zigzag decoding in this case. From the first bit of each coded packet, only $s_{2,1}$ is exposed bit. When it is substituted into the second bit of c_6, c_8, respectively, no new exposed bit can be obtained. The whole decoding process is locked, so the zigzag decoding cannot proceed, as shown in Figure 4.

5. Performance Analysis

In the case of one element takes part in the encoding only once in shift-and-addition encoding, the overhead of several existing CP-ZD codes is the square of k or n. As can be seen from Section 3, the overhead of the encoding framework that one element taking part in the encoding process multiple times when constructing one encoded packet is related to the size of the finite field; if the size of the finite field is $GF(2^{\omega})$, then $\omega = \lceil \log_2 n \rceil$. Therefore, the maximum overhead of this encoding framework is determined by the number n of encoded packets, and the overhead is $\mathrm{OH}_{\text{cauchy}} = \lceil \log_2 n \rceil - 1$. Therefore, the overhead OH has a logarithmic relationship with the number n of encoded packets, which has a huge advantage over the existing zigzag codes. However, due to the existence of multiple encodings, a source packet may be encoded once or more, which is more complex than the case of single encodings, leading to the possibility of decoding failure during zigzag decoding, as shown in Example 5. As can be seen from Figure 2, the ZD decoding rate of the encoding framework that one element takes part in the encoding process multiple times when constructing one encoded packet based on Cauchy matrix in shift-and-addition is about 80%.

In general, compared with the encoding framework of one element taking part in the encoding only once in shift-and-addition, the encoding framework of elements' multi-participation based on the Cauchy matrix has a good constraint on the overhead and can reduce the overhead from the existing square level to the logarithmic level. However, it has some losses in ZD properties and cannot guarantee ZD decoding.

6. Conclusions

Aiming at the problem of high overhead in CP-ZD codes, in this paper, we design a coding framework based on the idea of elements taking part in encoding multiple times in constructing an encoded packet based on Cauchy matrix and shift-and-addition. It is proved here that the framework has CP properties, but not completely with ZD properties. Experimental results show that the ZD decoding rate of this code is about 80%. However, the overhead is $\mathrm{OH}_{\mathrm{cauchy}} = \lceil \log_2 n \rceil - 1$, which is reduced from the existing square level to the logarithmic level. The coding framework confirms the advantage of the element participating in the encoding for multiple times and lays a foundation for future research.

7. Future Works

The new coding framework proposed in this paper satisfies the CP property, but not the ZD property. At present, there is no mature encoding framework with one element taking part in encoding process multiple times. Aiming at the idea of elements taking part in encoding multiple times, it is obviously of prospective academic and application value to design a feasible ZD decoding framework. What is more, it is also necessary to study the closed form expression of CP-ZD code, which can be used to describe the necessary and sufficient conditions of CP-ZD code that elements taking part in encoding process multiple times.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Key Research and Development Program under Grant 2019YFB1803305, the Natural Science Foundation of China (62071304), the Natural Science Foundation of Guangdong Province (2020A1515010381), the Basic Research Foundation of Shenzhen City (20200826152915001), the Guangdong Basic and Applied Basic Research Foundation (2022A1515011219), the Natural Science Foundation of Shenzhen City (JCYJ20190808120415286), and the Natural Science Foundation of Shenzhen University (00002501).

References

[1] M. Chen, S. Mao, and Y. Liu, "Big data: a survey," *Mobile networks & Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[2] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[3] P. Soto and J. Li, "Straggler-free coding for concurrent matrix multiplications," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 3017–3021, Los Angeles, CA, USA, June 2020.

[4] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: a study on the Facebook warehouse cluster," in *5th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 13)*, San Jose, CA, June 2013.

[5] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos et al., "XORing elephants," *Proceedings of the VLDB Endowment*, vol. 6, no. 5, pp. 325–336, 2013.

[6] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *IEEE Journal on Selected Areas in Communications*, vol. 99, no. 3, pp. 476–489, 2011.

[7] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: network coding for cloud storage," in *Proceedings - IEEE INFOCOM*, pp. 2801–2805, Orlando, FL, USA, March 2011.

[8] V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh, "Asymptotic interference alignment for optimal repair of MDS codes in distributed storage," *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 2974–2987, 2013.

[9] P. K. Akulakrishna, J. Lakshmi, and S. Nandy, "Efficient storage of big-data for real-time GPS applications," in *IEEE Fourth International Conference on Big Data & Cloud Computing*, pp. 1–8, Sydney, NSW, Australia, December 2014.

[10] Y. Aikebaier, T. E. Yang, and M. Takizawa, "Energy-efficient computation models for distributed systems," in *International Conference on Network-Based Information Systems*, pp. 423–431, Indianapolis, IN, USA, August 2009.

[11] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 34–40, 2017.

[12] A. Severinson, A. G. I Amat, and E. Rosnes, "Block-diagonal coding for distributed computing with straggling servers," in *2017 IEEE Information Theory Workshop (ITW)*, pp. 464–468, Kaohsiung, Taiwan, November 2017.

[13] A. B. Das, L. Tang, and A. Ramamoorthy, "C3LES: codes for coded computation that leverage stragglers," in *2018 IEEE Information Theory Workshop (ITW)*, pp. 1–5, Guangzhou, China, November 2018.

[14] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with D-dimensional product codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1993–1997, Vail, CO, USA, June 2018.

[15] M. Dai, Z. Zheng, S. Zhang, H. Wang, and X. Lin, "SAZD: a low computational load coded distributed computing framework for IoT systems," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3640–3649, 2020.

[16] L. Chen, R. Zhao, K. He, Z. Zhao, and L. Fan, "Intelligent ubiquitous computing for future UAV-enabled MEC network systems," *Cluster Computing*, vol. 25, no. 4, pp. 1–11, 2022.

[17] M. Dai, A. Xu, Q. Huang, Z. Zhang, and X. Lin, "Vertical federated DNN training," *Physical Communication*, vol. 49, p. 101465, 2021.

[18] J. Li, S. Dang, Y. Yan, Y. Peng, S. al-Rubaye, and A. Tsourdos, "Generalized quadrature spatial modulation and its application to vehicular networks with NOMA," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4030–4039, 2021.

[19] X. Pei, Y. Chen, M. Wen, H. Yu, E. Panayirci, and H. V. Poor, "Next-generation multiple access based on NOMA with power level modulation," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 4, pp. 1072–1083, 2022.

[20] M. Wen, Q. Li, K. J. Kim et al., "Private 5G networks: concepts, architectures, and research landscape," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 1, pp. 7–25, 2022.

[21] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 159–170, 2008.

[22] L. Zhang, W. Zhou, J. Xia et al., "DQN-based mobile edge computing for smart internet of vehicle," *EURASIP Journal on Advances in Signal Processing*, vol. 2022, 16 pages, 2022.

[23] J. Lu, L. Chen, J. Xia et al., "Analytical offloading design for mobile edge computing-based smart internet of vehicle," *EURASIP Journal on Advances in Signal Processing*, vol. 2022, 19 pages, 2022.

[24] H. Hou and Y. S. Han, "Cauchy MDS array codes with efficient decoding method," 2016, http://arxiv.org/abs/1611.09968.

[25] J. S. Plank, "A tutorial on Reed–Solomon coding for fault-tolerance in raid-like systems," *Software: Practice and Experience*, vol. 27, no. 9, pp. 995–1012, 1997.

[26] J. Bloemer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, *An XOR-based erasure-resilient coding scheme*, Technical Report at ICSI, 1999.

[27] M. Dai, C. W. Sung, H. Wang, X. Gong, and Z. Lu, "A new zigzag-decodable code with efficient repair in wireless distributed storage," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1218–1230, 2017.

[28] N. Jacobson, *Basic Algebra I*, Freeman & Co, 1985.