A Generic Transformation for MDS Codes with Optimal Repair Access and Low Sub-Packetization

Hanxu Hou, Patrick P. C. Lee and Yunghsiang S. Han

Abstract

Maximum distance separable (MDS) codes are of particular interest in distributed storage systems. Minimum storage regenerating (MSR) codes are a class of MDS codes that can repair any one failed node by accessing the minimum number of symbols from d surviving nodes. Existing results show that the sub-packetization α of an (n,k,d) MSR code with high code rate (i.e., $k/n > 0.5$) is at least $(d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$. In this paper, we propose a generic transformation for any (n,k) MDS code to obtain another (n,k) MDS code such that (i) the sub-packetization of the obtained MDS code is $d-k+1$, (ii) the repair access of any chosen $(d-k+1)\eta$ nodes of the obtained MDS code is optimal, where η is an integer with $\eta \geq 1$. By applying the proposed transformation to an (n,k) MDS code for $\lceil \frac{n}{(d-k+1)\eta} \rceil$ times, we can obtain an MSR code that the sub-packetization is $(d-k+1)^{\lceil \frac{n}{(d-k+1)\eta} \rceil}$ and the repair access is optimal for all nodes. We show that the sub-packetization of the proposed MSR codes is strictly less than the existing known lower bound when $\eta = \lfloor \frac{r-1}{d-k} \rfloor > 1$. In our MSR codes, a restriction on the chosen d nodes in repairing a failed node is that some of the chosen d nodes are specific. Moreover, we show that our transformation can also be employed to binary MDS array codes, using EVENODD codes as a motivating example, to obtain the transformed EVENODD codes that have optimal repair access for all nodes and have lower sub-packetization than the existing binary MDS array codes with optimal repair access for all nodes. With the proposed generic transformation, we can choose some well-designed MDS codes or binary MDS array codes with low computational complexity to obtain the

Hanxu Hou and Yunghsiang S. Han are with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology (E-mail: houhanxu@163.com, yunghsiangh@gmail.com). Patrick P. C. Lee is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (E-mail: pclee@cse.cuhk.edu.hk). This work was partially supported by the National Natural Science Foundation of China (No. 61701115, 61671007), Start Fund of Dongguan University of Technology (No. GB200902-19, KCYXM2017025), and Research Grants Council of Hong Kong (GRF 14216316 and CRF C7036-15G).

corresponding MSR codes that have low computational complexity, optimal repair access for all nodes and relatively small sub-packetization, which is attractive in the real distributed storage systems.

Index Terms

Minimum distance separable codes, minimum storage regenerating codes, optimal repair access, sub-packetization.

I. Introduction

Erasure codes are now widely adopted in distributed storage systems that can provide significantly higher fault tolerance and lower storage redundancy in comparing to traditional replication. Maximum distance separable (MDS) codes are a class of the erasure codes that can offer reliability at lowest possible extra storage.

Upon failure of a storage node, we want to repair the failed node by downloading the symbols from some surviving nodes as less as possible. *Regenerating codes* (RGCs) [1] are such erasure codes with the objective of minimizing the *repair bandwidth* that is defined as the total number of symbols downloaded in repairing a failed node. Minimum storage regenerating (MSR) codes are the sub-class of RGCs that correspond to the minimum storage point of RGCs. In addition, MSR codes can also be viewed as a special class of MDS codes that have optimal repair bandwidth for all nodes

A. Related Works

In an (n, k) MSR code, a data file of $k\alpha$ *symbols* over the finite field $\mathbb{F}_q$ is encoded into $n\alpha$ symbols, which are distributed into n storage nodes, where each node stores α symbols, such that the data file can be retrieved from any k out of n nodes. The number of symbols α stored in each node is also called *sub-packetization level*. It is shown in [1] that the number β of symbols downloaded from each of the chosen d helper nodes in repairing the α symbols of a failed node in an MSR code is lower bounded by

$$\beta = \frac{\alpha}{d-k+1}.$$

Therefore, the minimum repair bandwidth of an (n, k) MDS code is bounded by

$$\gamma = d\beta = \frac{d\alpha}{d-k+1}. \tag{1}$$

We say that the MDS code has *optimal repair bandwidth* if the lower bound in (1) is achieved for the repair bandwidth of any failure node, and has *optimal repair access* if the amount of symbols accessed in repairing the failure node is (1).

Many constructions of MSR codes [2]–[9] have been proposed in the literature. Product-matrix MSR codes are presented in [2] with parameters satisfying $2k-2 \leq d \leq n-1$, and are improved in [3] with lower computational complexity. Another class of MSR codes are constructed in [4] based on interference alignment. However, the above two constructions are only suitable for low code rates (i.e, $k/n \leq 0.5$). MSR codes with high code rates (i.e, $k/n > 0.5$) are more important in practical applications. Some existing constructions of MSR codes with high code rates can be found in [5]–[9]. It is shown in [10] that a tight lower bound of the sub-packetization of high code rate MSR codes with optimal repair access is $(d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$. More general, for an (n,k) MDS code with optimal repair access for w nodes ($w < n$), the minimum sub-packetization level is $\alpha = (d-k+1)^{\lceil \frac{w}{d-k+1} \rceil}$.

There are other important concerns in distributed storage systems, such as computational complexity. Binary MDS array codes are a special class of MDS codes that have low computational complexity since the encoding and decoding procedures only involve XOR operations. Typical examples of binary MDS array codes are EVENODD [11], [12], X-code [13] and RDP [14], [15]. Some efficient decoding methods of binary MDS array codes are given in [16]–[20]. There have been many studies [16], [21]–[24] on the optimal repair bandwidth of binary MDS array codes.

In this paper, we propose a generic transformation, which can be applied to any MDS code to obtain a new class of optimal repair access MSR codes for any parameters $k < d < n$ such that the sub-packetization level is less than the lower bound $(d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$ when $\lfloor \frac{n-k-1}{d-k} \rfloor > 1$. We also show how to apply the transformation for binary MDS array codes by using EVENODD codes as an example to obtain the transformed codes with optimal repair access, relatively small sub-packetization level and low computational complexity.

B. Contributions

The main contributions of this paper are as follows.

1) We present a generic transformation for any (n,k) MDS code to obtain another (n,k) MDS code that have optimal repair access for any chosen $(d-k+1)\eta$ nodes and the sub-packetization level of the obtained code is $d-k+1$, where $\eta = \lfloor \frac{r-1}{d-k} \rfloor \geq 1$.

2) By applying the proposed transformation for an (n, k) MDS code for $\lceil \frac{n}{(d-k+1)\eta} \rceil$ times, we can obtain an (n, k) MSR code that has optimal repair access for all nodes and the sub-packetization level is $(d-k+1)^{\lceil \frac{n}{(d-k+1)\eta} \rceil}$. When $\lfloor \frac{n-k-1}{d-k} \rfloor > 1$. We show that the proposed MSR codes have less sub-packetization level than the lower bound given in [10] and also less than that of the existing high code rates MSR codes.
3) By applying the proposed transformation for an (n, k) MDS code for $\lceil \frac{w}{(d-k+1)\eta} \rceil$ times, we can obtain an (n, k) MDS code that has optimal repair access for w nodes and the sub-packetization level is $(d-k+1)^{\lceil \frac{w}{(d-k+1)\eta} \rceil}$, which is less than the lower bound in [10] when $\lfloor \frac{n-k-1}{d-k} \rfloor > 1$.
4) We present the transformation in the binary array version for binary MDS array codes. We use EVENODD codes as a motivating example to support optimal repair access for the chosen $(d-k+1)\eta$ columns. By recursively applying the transformation for a binary MDS array code, the obtained transformed MDS array code has optimal repair access for all columns and have lower sub-packetization than that of the existing binary MDS array codes with optimal repair access.

There exist some similar transformations [6], [16], [24] for MDS codes to enable optimal repair bandwidth of some nodes. The main differences between our transformation and the transformations in [6], [16], [24] are summarized as follows. First, Li *et al.* [6] propose a transformation for MDS codes to enable optimal repair bandwidth for any of the chosen r nodes. Our transformation can enable efficient repair bandwidth for any of the chosen $(d-k+1)\eta$ nodes. The transformation in [6] can be viewed as a special case of our transformation with $\eta = 1$ and $d = n-1$. Second, the transformations in [16], [24] can be viewed as the variants of the transformation in [6] that are designed for binary MDS array codes. With a slightly modification, our transformation is also applicable to binary MDS array codes and the transformation in [16], [24] can be viewed as a special case of our transformation in the binary version.

The rest of the paper is organized as follows. Section II presents a motivating example that illustrates the main ideas of the transformation. Section III presents the transformation that can be applied to any MDS code to enable optimal repair access for any chosen $(d-k+1)\eta$ nodes. Section IV shows the construction of MDS codes with optimal repair access for all nodes by recursively applying the proposed transformation for any MDS code. We show that the proposed transformation can also be applied for EVENODD codes to enable optimal repair access for all

nodes in Section V. We show in Section VI that the obtained MDS codes with optimal repair access have lower sub-packetization than that of the existing MDS codes with optimal repair access. Section VII concludes the paper.

II. A Motivating Example

In this section, we present an example of the transformed (n, k) MDS code with $n = 8$, $k = 5$ and $d = 6$ that has optimal repair access for each node and the sub-packetization is 4. Note that the example is obtained by recursively applying the transformation given in Section III twice.

In the example, the data file contains 20 data symbols, which are denoted as $c_1^\ell, c_2^\ell, c_3^\ell, c_4^\ell, c_5^\ell$ over finite field $\mathbb{F}_q$, where $\ell = 1, 2, 3, 4$. We need to first obtain 32 symbols and then store them in eight nodes with each node storing four symbols such that any $k = 5$ out of $n = 8$ nodes can retrieve 20 data symbols. In the example, we have eight nodes with each node storing four symbols. Given the 20 data symbols $c_1^\ell, c_2^\ell, c_3^\ell, c_4^\ell, c_5^\ell$, we first compute 12 coded symbols $c_6^\ell, c_7^\ell, c_8^\ell$ with $\ell = 1, 2, 3, 4$ by

$$\begin{bmatrix} c_6^\ell & c_7^\ell & c_8^\ell \end{bmatrix} = \begin{bmatrix} c_1^\ell & c_2^\ell & c_3^\ell & c_4^\ell & c_5^\ell \end{bmatrix} \begin{bmatrix} 1 & p_1 & p_1^2 \\ 1 & p_2 & p_2^2 \\ 1 & p_3 & p_3^2 \\ 1 & p_4 & p_4^2 \\ 1 & p_5 & p_5^2 \end{bmatrix},$$

where p_1, p_2, p_3, p_4, p_5 are distinct and non-zero elements in $\mathbb{F}_q$. Thus, we have 32 symbols

$$\begin{bmatrix} c_1^1 & c_2^1 & c_3^1 & c_4^1 & c_5^1 & c_6^1 & c_7^1 & c_8^1 \\ c_1^2 & c_2^2 & c_3^2 & c_4^2 & c_5^2 & c_6^2 & c_7^2 & c_8^2 \\ c_1^3 & c_2^3 & c_3^3 & c_4^3 & c_5^3 & c_6^3 & c_7^3 & c_8^3 \\ c_1^4 & c_2^4 & c_3^4 & c_4^4 & c_5^4 & c_6^4 & c_7^4 & c_8^4 \end{bmatrix}.$$

Table I shows the storage by applying the transformation given in Section III for the first four nodes, where e_1, e_2 are field elements in $\mathbb{F}_q$ except zero and one. Note that the first two symbols stored in each node are the same as the last two symbols stored in the same node in Table I, except that the superscripts are different. In fact, the code in Table I can be viewed as two instances of the code with $n = 8$, $k = 5$ and $d = 6$ by applying the transformation given in Section III for the first four nodes. Table II shows the code by recursively applying the

TABLE I: The storage with $n = 8$, $k = 5$ and $d = 6$ by applying the transformation for the first four nodes, where e_1, e_2 are field elements in $\mathbb{F}_q$ except zero and one.

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8
c_1^1	$c_2^1 + c_1^2$	c_3^1	$c_4^1 + c_3^2$	c_5^1	c_6^1	c_7^1	c_8^1
$c_1^2 + e_1 c_2^1$	c_2^2	$c_3^2 + e_2 c_4^1$	c_4^2	c_5^2	c_6^2	c_7^2	c_8^2
c_1^3	$c_2^3 + c_1^4$	c_3^3	$c_4^3 + c_3^4$	c_5^3	c_6^3	c_7^3	c_8^3
$c_1^4 + e_1 c_2^3$	c_2^4	$c_3^4 + e_2 c_4^3$	c_4^4	c_5^4	c_6^4	c_7^4	c_8^4

TABLE II: The code with $n = 8$, $k = 5$ and $d = 6$ by recursively applying the transformation twice, where e_1, e_2, e_3, e_4 are field elements in $\mathbb{F}_q$ except zero and one.

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8
c_1^1	$c_2^1 + c_1^2$	c_3^1	$c_4^1 + c_3^2$	c_5^1	$c_6^1 + c_5^3$	c_7^1	$c_8^1 + c_7^3$
$c_1^2 + e_1 c_2^1$	c_2^2	$c_3^2 + e_2 c_4^1$	c_4^2	c_5^2	$c_6^2 + c_5^4$	c_7^2	$c_8^2 + c_7^4$
c_1^3	$c_2^3 + c_1^4$	c_3^3	$c_4^3 + c_3^4$	$c_5^3 + e_3 c_6^1$	c_6^3	$c_7^3 + e_4 c_8^1$	c_8^3
$c_1^4 + e_1 c_2^3$	c_2^4	$c_3^4 + e_2 c_4^3$	c_4^4	$c_5^4 + e_3 c_6^2$	c_6^4	$c_7^4 + e_4 c_8^2$	c_8^4

transformation given in Section III twice, where e_1, e_2, e_3, e_4 are field elements in $\mathbb{F}_q$ except zero and one. The code in Table I has optimal repair access for each of the first four nodes, while the code in Table II has optimal repair access for each of all eight nodes. In the following, we show the detailed repair method for the code in Table II, the repair method of each of the first four node in Table I is similar to that in Table II.

We demonstrate that we can repair the four symbols in each node of the code in Table II by accessing two symbols from each of the chosen $d = 6$ nodes. Suppose that node 1 fails, we can recover the four symbols stored in node 1 by downloading the first symbol and the third symbol from nodes $2, 3, 5, 6, 7, 8$, i.e., by downloading the following bits.

$$c_2^1 + c_1^2, c_3^1, c_5^1, c_6^1 + c_5^3, c_7^1, c_8^1 + c_7^3,$$

$$c_2^3 + c_1^4, c_3^3, c_5^3 + e_3 c_6^1, c_6^3, c_7^3 + e_4 c_8^1, c_8^3.$$

Fig. 1 shows the detailed repair procedure of node 1. Specifically, we can first compute c_6^1 and c_5^3 from $c_6^1 + c_5^3$ and $c_5^3 + e_3 c_6^1$, and compute c_8^1 and c_7^3 from $c_8^1 + c_7^3$ and $c_7^3 + e_4 c_8^1$, as $e_3 \neq 1$ and

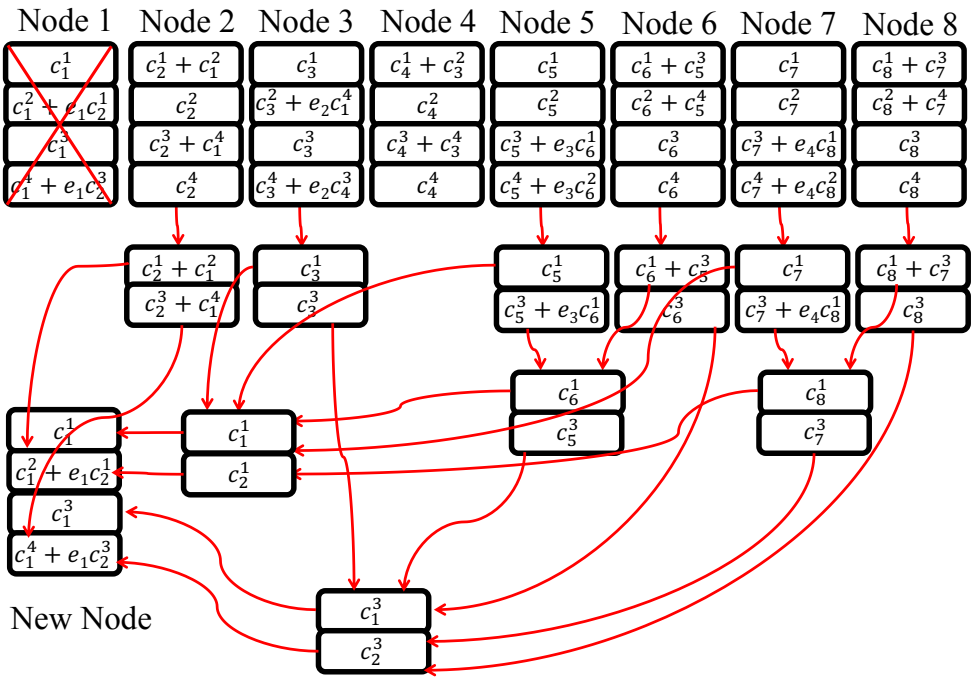

Fig. 1: The repair procedure of node 1 of the example with $k = 5$, $r = 3$ and $d = 6$.

$e_4 \neq 1$. Then, we can obtain c_1^1, c_2^1, c_4^1 and c_1^3, c_2^3, c_4^3 from $c_3^1, c_5^1, c_6^1, c_7^1, c_8^1$ and $c_3^3, c_5^3, c_6^3, c_7^3, c_8^3$ by

$$\begin{bmatrix} c_1^1 & c_2^1 & c_4^1 \end{bmatrix} = \begin{bmatrix} c_6^1 - c_3^1 - c_5^1 & c_7^1 - p_3 c_3^1 - p_5 c_5^1 & c_8^1 - p_3^2 c_3^1 - p_5^2 c_5^1 \end{bmatrix} \cdot \begin{bmatrix} 1 & p_1 & p_1^2 \\ 1 & p_2 & p_2^2 \\ 1 & p_4 & p_4^2 \end{bmatrix}^{-1},$$

and

$$\begin{bmatrix} c_1^3 & c_2^3 & c_4^3 \end{bmatrix} = \begin{bmatrix} c_6^3 - c_3^3 - c_5^3 & c_7^3 - p_3 c_3^3 - p_5 c_5^3 & c_8^3 - p_3^2 c_3^3 - p_5^2 c_5^3 \end{bmatrix} \cdot \begin{bmatrix} 1 & p_1 & p_1^2 \\ 1 & p_2 & p_2^2 \\ 1 & p_4 & p_4^2 \end{bmatrix}^{-1},$$

respectively. Finally, we can recover $c_1^2 + e_1 c_2^1$ and $c_1^4 + e_1 c_2^3$ by $c_1^2 + e_1 c_2^1 = (c_2^1 + c_1^2) + (e_1 - 1)c_2^1$ and $c_1^4 + e_1 c_2^3 = (c_2^3 + c_1^4) + (e_1 - 1)c_2^3$, respectively. We can also repair node 3 by downloading the following symbols

$$c_1^1, c_4^1 + c_3^2, c_5^1, c_6^1 + c_5^3, c_7^1, c_8^1 + c_7^3,$$

$$c_1^3, c_4^3 + c_3^4, c_5^3 + e_3 c_6^1, c_6^3, c_7^3 + e_4 c_8^1, c_8^3,$$

from nodes $1, 4, 5, 6, 7, 8$. Node 2 and node 4 can be repaired by downloading the symbols

$$c_1^2 + e_1 c_2^1, c_4^2, c_5^2, c_6^2 + c_5^4, c_7^2, c_8^2 + c_7^4,$$
$$c_1^4 + e_1 c_2^3, c_4^4, c_5^4 + e_3 c_6^2, c_6^4, c_7^4 + e_4 c_8^2, c_8^4,$$

from nodes $1, 4, 5, 6, 7, 8$ and

$$c_2^2, c_3^2 + e_2 c_1^4, c_5^2, c_6^2 + c_5^4, c_7^2, c_8^2 + c_7^4,$$
$$c_2^4, c_3^4 + e_2 c_4^3, c_5^4 + e_3 c_6^2, c_6^4, c_7^4 + e_4 c_8^2, c_8^4,$$

from nodes $2, 3, 5, 6, 7, 8$, respectively.

Similarly, we can repair node 5 and node 7 by downloading the symbols

$$c_1^1, c_2^1 + c_1^2, c_3^1, c_4^1 + c_3^2, c_6^1 + c_5^3, c_7^1,$$
$$c_1^2 + e_1 c_2^1, c_2^2, c_3^2 + e_2 c_1^4, c_4^2, c_6^2 + c_5^4, c_7^2,$$

from nodes $1, 2, 3, 4, 6, 7$ and

$$c_1^1, c_2^1 + c_1^2, c_3^1, c_4^1 + c_3^2, c_5^1, c_8^1 + c_7^3,$$
$$c_1^2 + e_1 c_2^1, c_2^2, c_3^2 + e_2 c_1^4, c_4^2, c_5^2, c_8^2 + c_7^4,$$

from nodes $1, 2, 3, 4, 5, 8$, respectively, and repair node 6 and node 8 by downloading the symbols

$$c_1^3, c_2^3 + c_1^4, c_3^3, c_4^3 + c_3^4, c_5^3 + e_3 c_6^1, c_8^3,$$
$$c_1^4 + e_1 c_2^3, c_2^4, c_3^4 + e_2 c_4^3, c_4^4, c_5^4 + e_3 c_6^2, c_8^4,$$

from nodes $1, 2, 3, 4, 5, 8$ and

$$c_1^3, c_2^3 + c_1^4, c_3^3, c_4^3 + c_3^4, c_6^3, c_7^3 + e_4 c_8^1,$$
$$c_1^4 + e_1 c_2^3, c_2^4, c_3^4 + e_2 c_4^3, c_4^4, c_6^4, c_7^4 + e_4 c_8^2,$$

from nodes $1, 2, 3, 4, 6, 7$, respectively.

If the field size q is large enough, we can always find the field elements $p_1, p_2, p_3, p_4, p_5, e_1, e_2, e_3, e_4$ in $\mathbb{F}_q$ such that any five out of the eight nodes can retrieve all 20 data symbols by Theorem 4 given in Section III-C.

III. A GENERIC TRANSFORMATION

In this section, we present a generic transformation for MDS codes that can convert any MDS code into another MDS code that has optimal repair access for each node and possess low sub-packetization.

An (n, k) MDS code encodes k *data symbols* $s_1, s_2, \ldots, s_k$ over finite field $\mathbb{F}_q$ into n *coded symbols* $c_1, c_2, \ldots, c_n$ by

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} = \begin{bmatrix} s_1 & s_2 & \cdots & s_k \end{bmatrix} \cdot \mathbf{G}_{k\times n},$$

where $\mathbf{G}_{k\times n}$ is a $k \times n$ matrix. Any $k \times k$ sub-matrix of $\mathbf{G}_{k\times n}$ must be non-singular, in order to maintain the MDS property. Typically, we can choose the matrix $\mathbf{G}_{k\times n}$ to be a $k\times n$ Vandermonde matrix or Cauchy matrix. If we want to generate the systematic version of the code, i.e., $c_i = s_i$ for $i = 1, 2, \ldots, k$, then the matrix $\mathbf{G}_{k\times n}$ is composed of a $k \times k$ identity matrix $\mathbf{I}_{k\times k}$ and a $k \times (r = n - k)$ encoding matrix $\mathbf{P}_{k\times r}$, i.e.,

$$\mathbf{G}_{k\times n} = \begin{bmatrix} \mathbf{I}_{k\times k} & \mathbf{P}_{k\times r} \end{bmatrix}, \tag{2}$$

where

$$\mathbf{P}_{k\times r} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,r} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k,1} & p_{k,2} & \cdots & p_{k,r} \end{bmatrix}.$$

To ensure the MDS property, all the square sub-matrices of $\mathbf{P}_{k\times r}$ should be invertible.

A. The Transformation

Next we present a transformation on an (n, k) MDS code to generate an (n, k) transformed MDS code with $\alpha = (d - k + 1)$. In the transformed codes, we have $(d - k + 1) \cdot k$ data symbols $s_1^\ell, s_2^\ell, \ldots, s_k^\ell$ with $\ell = 1, 2, \ldots, d - k + 1$, where $k + 1 \le d \le k + r - 1$. We can compute $(d - k + 1) \cdot n$ *coded symbols* $c_1^\ell, c_2^\ell, \ldots, c_n^\ell$ by

$$\begin{bmatrix} c_1^\ell & c_2^\ell & \cdots & c_n^\ell \end{bmatrix} = \begin{bmatrix} s_1^\ell & s_2^\ell & \cdots & s_k^\ell \end{bmatrix} \cdot \mathbf{G}_{k\times n},$$

where $\mathbf{G}_{k\times n}$ is given in (2) and $\ell = 1, 2, \ldots, d - k + 1$. Let $t = d - k + 1$ and denote η as

$$\eta = \left\lfloor \frac{r-1}{d-k} \right\rfloor.$$

TABLE III: The transformed codes with $n = 11$, $k = 6$, $r = 5$, $d = 8$ and $\eta = 2$.

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	Node 10	Node 11
c_1^1	$c_2^1 + c_1^2$	$c_3^1 + c_1^3$	c_4^1	$c_5^1 + c_4^2$	$c_6^1 + c_4^3$	c_7^1	c_8^1	c_9^1	c_{10}^1	c_{11}^1
$c_1^2 + e_1 c_2^1$	c_2^2	$c_3^2 + c_2^3$	$c_4^2 + e_2 c_5^1$	c_5^2	$c_6^2 + c_5^3$	c_7^2	c_8^2	c_9^2	c_{10}^2	c_{11}^2
$c_1^3 + e_1 c_3^1$	$c_2^3 + e_1 c_3^2$	c_3^3	$c_4^3 + e_2 c_6^1$	$c_5^3 + e_2 c_6^2$	c_6^3	c_7^3	c_8^3	c_9^3	c_{10}^3	c_{11}^3

For $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, \eta$, node $(j-1)t + i$ stores the following t symbols

$$\begin{aligned}
&c^1_{(j-1)t+i} + c^i_{(j-1)t+1},\\
&c^2_{(j-1)t+i} + c^i_{(j-1)t+2}, \ldots,\\
&c^{i-1}_{(j-1)t+i} + c^i_{(j-1)t+i-1},\\
&c^i_{(j-1)t+i},\\
&c^{i+1}_{(j-1)t+i} + e_j c^i_{(j-1)t+i+1},\\
&c^{i+2}_{(j-1)t+i} + e_j c^i_{(j-1)t+i+2}, \ldots,\\
&c^t_{(j-1)t+i} + e_j c^i_{(j-1)t+t},
\end{aligned} \tag{3}$$

where e_j is an element from the finite field except zero and one. For $h = t \cdot \eta + 1, t \cdot \eta + 2, \ldots, n$, node h stores t symbols

$$c_h^1, c_h^2, \ldots, c_h^t.$$

The obtained codes are called *transformed codes*, which are denoted as $\mathcal{C}_1(n, k, \eta, t)$. Note that the transformation in [6] can be viewed as a special case of our transformation with $d = n - 1$ and $\eta = 1$. Table III shows an example of the transformed codes with $n = 11$, $k = 6$, $r = 5$, $d = 8$ and $\eta = 2$.

Lemma 1. *Given integers ℓ, i and j, we can compute*

1) *c_ℓ^i and c_ℓ^j from $c_\ell^i + c_\ell^j$ and $c_\ell^i + e c_\ell^j$, if $e \neq 1$;*
2) *$c_\ell^i + c_\ell^j$ from $c_\ell^i + e c_\ell^j$ and c_ℓ^i, if $e \neq 0$;*
3) *$c_\ell^i + e c_\ell^j$ from $c_\ell^i + c_\ell^j$ and c_ℓ^i, if $e \neq 0$.*

Proof. Consider the first claim, we can obtain c_ℓ^j by

$$c_\ell^j = \{(c_\ell^i + c_\ell^j) - (c_\ell^i + e c_\ell^j)\}(1 - e)^{-1},$$

and further compute c_ℓ^i by $c_\ell^j + (c_\ell^i + c_\ell^j)$. The other two claims can be proved similarly. □

B. Optimal Repair Access

We show in the next theorem that the first $t \cdot \eta$ nodes of the transformed codes have optimal repair access.

Theorem 2. *We can recover each of the first $t \cdot \eta$ nodes of the transformed codes by accessing d symbols from d nodes.*

Proof. For $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, \eta$, we show that we can recover t symbols stored in node $(j-1)t+i$ by accessing k symbols $c^i_{h_1}, c^i_{h_2}, \ldots, c^i_{h_k}$ with

$$\{h_1, \ldots, h_k\} \subset \begin{cases} \{t+i, \ldots, (\eta-1)t+i, t\eta+1, t\eta+2, \ldots, n\} \text{ for } j = 1 \\ \{i, \ldots, (j-2)t+i, jt+i, \ldots, (\eta-1)t+i, t\eta+1, \ldots, n\} \text{ for } \eta-1 \geq j \geq 2 \\ \{i, t+i, \ldots, (\eta-2)t+i, t\eta+1, t\eta+2, \ldots, n\} \text{ for } j = \eta \end{cases} \tag{4}$$

and $d-k$ symbols

$$\begin{aligned} &c^i_{(j-1)t+1} + e_j c^1_{(j-1)t+i}, \ldots, c^i_{(j-1)t+i-1} + e_j c^{i-1}_{(j-1)t+i}, \\ &c^i_{(j-1)t+i+1} + c^{i+1}_{(j-1)t+i}, \ldots, c^i_{(j-1)t+t} + c^t_{(j-1)t+i}. \end{aligned} \tag{5}$$

Note that

$$\begin{aligned} \eta - 1 + (n - t\eta) =& n - (d-k) \left\lfloor \frac{n-k-1}{d-k} \right\rfloor - 1 \\ \geq& k, \end{aligned}$$

we can thus choose k different values in (4).

Recall that the t symbols stored in node $(j-1)t+i$ are given in (3). By accessing $c^i_{h_1}, c^i_{h_2}, \ldots, c^i_{h_k}$, we can compute $c^i_{(j-1)t+1}, c^i_{(j-1)t+2}, \ldots, c^i_{(j-1)t+t}$ according to the MDS property. With the computed $c^i_{(j-1)t+1}, c^i_{(j-1)t+2}, \ldots, c^i_{(j-1)t+t}$ and the accessed $d-k$ symbols in (5), we can compute all t symbols stored in node $(j-1)t+i$ by Lemma 1. Therefore, we can recover node $(j-1)t+i$ by downloading d symbols from d helper nodes and the repair access of node $(j-1)t+i$ is optimal according to (1). □

Consider the example in Table III. We can repair the three symbols c^1_1, $c^2_1 + e_1 c^1_2$ and $c^3_1 + e_1 c^1_3$ in node 1 by downloading the following eight symbols

$$c^1_4, c^1_7, c^1_8, c^1_9, c^1_{10}, c^1_{11}, c^1_2 + c^2_1, c^1_3 + c^3_1.$$

Specifically, we can first compute c_1^1, c_2^1 and c_3^1 from the first six symbols of the above downloaded symbols, and then compute $c_1^2 + e_1 c_2^1$ and $c_1^3 + e_1 c_3^1$ by

$$c_1^2 + e_1 c_2^1 = (c_2^1 + c_1^2) + (e_1 - 1)c_2^1,$$
$$c_1^3 + e_1 c_3^1 = (c_3^1 + c_1^3) + (e_1 - 1)c_3^1.$$

We can repair node 2 and node 3 by downloading

$$c_5^2, c_7^2, c_8^2, c_9^2, c_{10}^2, c_{11}^2, c_1^2 + e_1 c_2^1, c_3^2 + c_2^3,$$

and

$$c_6^3, c_7^3, c_8^3, c_9^3, c_{10}^3, c_{11}^3, c_1^3 + e_1 c_3^1, c_2^3 + e_1 c_3^2,$$

respectively. Similarly, we can repair node 4, node 5 and node 6 by downloading

$$c_1^1, c_7^1, c_8^1, c_9^1, c_{10}^1, c_{11}^1, c_5^1 + c_4^2, c_6^1 + c_4^3,$$

$$c_2^2, c_7^2, c_8^2, c_9^2, c_{10}^2, c_{11}^2, c_4^2 + e_2 c_5^1, c_6^2 + c_5^3,$$

and

$$c_3^3, c_7^3, c_8^3, c_9^3, c_{10}^3, c_{11}^3, c_4^3 + e_2 c_6^1, c_5^3 + e_2 c_6^2,$$

respectively.

C. The MDS Property

If we put tk data symbols in a vector $\mathbf{s}$ of length tk, i.e.,

$$\mathbf{s} = \begin{bmatrix} s_1^1 & \cdots & s_k^1 & s_1^2 & \cdots & s_k^2 & \cdots & s_1^t & \cdots & s_k^t \end{bmatrix},$$

then all tn symbols stored in n nodes are computed as the multiplication of $\mathbf{s}$ and the $tk \times tn$ generator matrix $\mathbf{G}_{tk \times tn}$. We can write the matrix $\mathbf{G}_{tk \times tn}$ as

$$\mathbf{G}_{tk \times tn} = \begin{bmatrix} \mathbf{G}_{tk \times t}^1 & \mathbf{G}_{tk \times t}^2 & \cdots & \mathbf{G}_{tk \times t}^n \end{bmatrix},$$

where $\mathbf{G}_{tk \times t}^i$ is the generator matrix of node i with $i = 1, 2, \ldots, n$. In the example in Table III, the generator matrix is

$$\mathbf{G}_{18 \times 33} = \begin{bmatrix} \mathbf{G}_{18 \times 3}^1 & \mathbf{G}_{18 \times 3}^2 & \cdots & \mathbf{G}_{18 \times 3}^{11} \end{bmatrix},$$

where

$$\begin{bmatrix} \mathbf{G}^1_{18\times 3} & \mathbf{G}^2_{18\times 3} & \mathbf{G}^3_{18\times 3} & \mathbf{G}^4_{18\times 3} & \mathbf{G}^5_{18\times 3} & \mathbf{G}^6_{18\times 3} \end{bmatrix}$$

$$= \left[\begin{array}{ccc|ccc|ccc|ccc|ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e_1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_2 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

In the following, we show that we can always find $e_1, e_2, \ldots, e_\eta$ such that the determinant of the $tk \times tk$ matrix composed of any k out of the n generator matrices

$$\mathbf{G}^1_{tk\times t}, \mathbf{G}^2_{tk\times t}, \cdots, \mathbf{G}^n_{tk\times t} \tag{6}$$

is non-zero, when the field size is large enough. We first review the Schwartz-Zippel Lemma, and then present the condition of MDS property.

Lemma 3. *(Schwartz-Zippel [25]) Let $Q(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a non-zero multivariate polynomial of total degree d. Let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from a subset $\mathbb{S}$ of $\mathbb{F}_q$. Then*

$$Pr[Q(r_1, \ldots, r_n) = 0] \leq \frac{d}{|\mathbb{S}|}. \tag{7}$$

The MDS property condition is given in the following theorem.

Theorem 4. *If the field size q is larger than*

$$\eta^{\frac{(t-1)t}{2}}\left(\binom{n}{k}-\sum_{\ell=0}^{\eta}\binom{n-\eta t}{k-\ell t}\cdot\binom{\eta}{\ell}\right), \tag{8}$$

then there exist η variables $e_1, e_2, \ldots, e_\eta$ over $\mathbb{F}_q$ such that the determinant of the $tk \times tk$ matrix composed of the generator matrices of any k nodes is non-zero.

Proof. We view each entry of the matrix $\mathbf{G}_{k\times n}$ as a constant and $e_1, e_2, \ldots, e_\eta$ as variables. We need to evaluate $\binom{n}{k}$ determinants that correspond to the determinants of the matrices composed of any k out of n matrices in (6) to be non-zero element in $\mathbb{F}_q$. We divide the first ηt nodes into η groups, group j contains t nodes that are from nodes $(j-1)t+1$ to node jt with $j = 1, 2, \ldots, \eta$. We first prove the following simple lemma.

Lemma 5. *For $j = 1, 2, \ldots, \eta$, we can compute t^2 coded symbols*

$$c^{\ell}_{(j-1)t+1}, c^{\ell}_{(j-1)t+2}, \cdots, c^{\ell}_{jt},$$

for $\ell = 1, 2, \ldots, t$, from t^2 symbols stored in t nodes of group j.

Proof. Let us consider the following t nodes of group j:

$$\begin{bmatrix} \text{Node } (j-1)t+1 & \text{Node } (j-1)t+2 & \cdots & \text{Node } jt \\ c^1_{(j-1)t+1} & c^1_{(j-1)t+2}+c^2_{(j-1)t+1} & \cdots & c^1_{jt}+c^t_{(j-1)t+1} \\ c^2_{(j-1)t+1}+e_jc^1_{(j-1)t+2} & c^2_{(j-1)t+2} & \cdots & c^2_{jt}+c^t_{(j-1)t+2} \\ c^3_{(j-1)t+1}+e_jc^1_{(j-1)t+3} & c^3_{(j-1)t+2}+e_jc^2_{(j-1)t+3} & \cdots & c^3_{jt}+c^t_{(j-1)t+3} \\ \vdots & \vdots & \ddots & \vdots \\ c^t_{(j-1)t+1}+e_jc^1_{jt} & c^t_{(j-1)t+2}+e_jc^2_{jt} & \cdots & c^t_{jt} \end{bmatrix}.$$

For $i = 1, 2, \ldots, t-1$ and $\ell = i+1, i+2, \ldots, t$, the symbol in row ℓ stored in node $(j-1)t+i$ is $c^{\ell}_{(j-1)t+i}+e_jc^i_{(j-1)t+\ell}$ and the symbol in row i stored in node $(j-1)t+\ell$ is $c^i_{(j-1)t+\ell}+c^{\ell}_{(j-1)t+i}$. By Lemma 1, we can compute two coded symbols $c^i_{(j-1)t+\ell}$ and $c^{\ell}_{(j-1)t+i}$ from $c^{\ell}_{(j-1)t+i}+e_jc^i_{(j-1)t+\ell}$ and $(j-1)t+\ell$ is $c^i_{(j-1)t+\ell}+c^{\ell}_{(j-1)t+i}$. Recall that the symbol in row ℓ stored in node $(j-1)t+\ell$ is the coded symbol $c^{\ell}_{(j-1)t+\ell}$. Therefore, we can compute the t^2 coded symbols from t^2 symbols stored in t nodes of group j. □

TABLE IV: The values in (8) with some specific parameters.

Parameters (n, k, η, t)	(8,5,2,2)	(9,6,2,2)	(14,10,2,2)
Values in (8)	92	128	1400

If the k nodes are composed of $k - \ell t$ nodes from the last $n - \eta t$ nodes and ℓt nodes from ℓ out of η groups, then we can obtain tk coded symbols by Lemma 5 and further compute tk data symbols, where $\ell \leq \eta$. Therefore, we only need to evaluate

$$\binom{n}{k} - \sum_{\ell=0}^{\eta} \binom{n-\eta t}{k-\ell t} \cdot \binom{\eta}{\ell}$$

determinants to be non-zero in $\mathbb{F}_q$ or not. Note that there are η variables $e_1, e_2, \ldots, e_\eta$ and each of the η variables appears in $(t-1)t/2$ entries of the generator matrix $\mathbf{G}_{tk \times tn}$. Therefore, each determinant is a polynomial with at most degree $\eta(t-1)t/2$. The multiplication of all the determinants can be interpreted as a polynomial with total degree

$$\eta \frac{(t-1)t}{2} \left(\binom{n}{k} - \sum_{\ell=0}^{\eta} \binom{n-\eta t}{k-\ell t} \cdot \binom{\eta}{\ell} \right).$$

Therefore, the MDS property condition is satisfied if the field size is larger than (8) according to the Schwartz-Zippel Lemma. □

Table IV shows the underlying field size such that there exist at least one assignment of $e_1, e_2, \ldots, e_\eta$ with the transformed codes satisfying the MDS property.

According to Theorem 2, the transformed codes have optimal repair access for each of the first ηt nodes. Similarly, we can also apply the transformation for nodes from $\eta t + 1$ to $2\eta t$ such that the repair access of each of nodes from $\eta t + 1$ to $2\eta t$ is optimal.

D. Systematic Codes

Note that the transformed code $\mathcal{C}_1(n, k, \eta, t)$ given in Section III-A is non-systematic code. It is important to obtain the systematic transformed code for practical consideration. We can obtain the systematic code by replacing $c^{\ell}_{(j-1)t+i} + c^{i}_{(j-1)t+\ell}$ with $\ell < i$ by $\bar{c}^{\ell}_{(j-1)t+i}$ and replacing

$c^{\ell}_{(j-1)t+i} + e_j c^{i}_{(j-1)t+\ell}$ with $\ell > i$ by $\bar{c}^{\ell}_{(j-1)t+i}$, i.e.,

$$\begin{aligned}
\bar{c}^{1}_{(j-1)t+i} &= c^{1}_{(j-1)t+i} + c^{i}_{(j-1)t+1},\\
\bar{c}^{2}_{(j-1)t+i} &= c^{2}_{(j-1)t+i} + c^{i}_{(j-1)t+2}, \ldots,\\
\bar{c}^{i-1}_{(j-1)t+i} &= c^{i-1}_{(j-1)t+i} + c^{i}_{(j-1)t+i-1},\\
\bar{c}^{i+1}_{(j-1)t+i} &= c^{i+1}_{(j-1)t+i} + e_j c^{i}_{(j-1)t+i+1},\\
\bar{c}^{i+2}_{(j-1)t+i} &= c^{i+2}_{(j-1)t+i} + e_j c^{i}_{(j-1)t+i+2}, \ldots,\\
\bar{c}^{t}_{(j-1)t+i} &= c^{t}_{(j-1)t+i} + e_j c^{i}_{(j-1)t+t},
\end{aligned} \tag{9}$$

where $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, \eta$. For $\ell < i$, we have

$$\begin{aligned}
\bar{c}^{\ell}_{(j-1)t+i} =& c^{\ell}_{(j-1)t+i} + c^{i}_{(j-1)t+\ell},\\
\bar{c}^{i}_{(j-1)t+\ell} =& c^{i}_{(j-1)t+\ell} + e_j c^{\ell}_{(j-1)t+i},
\end{aligned}$$

and further obtain

$$\begin{aligned}
c^{\ell}_{(j-1)t+i} =& \frac{\bar{c}^{i}_{(j-1)t+\ell} - \bar{c}^{\ell}_{(j-1)t+i}}{e_j - 1},\\
c^{i}_{(j-1)t+\ell} =& \frac{e_j \bar{c}^{\ell}_{(j-1)t+i} - \bar{c}^{i}_{(j-1)t+\ell}}{e_j - 1}.
\end{aligned}$$

In the above equation, $\frac{1}{e_j - 1}$ is the inverse of $e_j - 1$ over the finite field. Recall that $c^{\ell}_i = s^{\ell}_i$ for $\ell = 1, 2, \ldots, t$ and $h = 1, 2, \ldots, k$, and

$$c^{\ell}_h = p_{1,h-k} c^{\ell}_1 + p_{2,h-k} c^{\ell}_2 + \ldots + p_{k,h-k} c^{\ell}_k,$$

for $h = k+1, k+2, \ldots, n$ and $\ell = 1, 2, \ldots, t$. We can thus obtain that

$$\begin{aligned}
c^{\ell}_h =& \Big(\sum_{i=1}^{\ell-1} p_{i,h-k} \frac{e_1 \bar{c}^{i}_{\ell} - \bar{c}^{\ell}_{i}}{e_1 - 1} \Big) + p_{\ell,h-k} c^{\ell}_{\ell} + \Big(\sum_{i=\ell+1}^{t} p_{i,h-k} \frac{\bar{c}^{i}_{\ell} - \bar{c}^{\ell}_{i}}{e_1 - 1} \Big) + \cdots +\\
& \Big(\sum_{i=1}^{\ell-1} p_{(\eta-1)t+i,h-k} \frac{e_\eta \bar{c}^{i}_{(\eta-1)t+\ell} - \bar{c}^{\ell}_{(\eta-1)t+i}}{e_\eta - 1} \Big) + p_{(\eta-1)t+\ell,h-k} c^{\ell}_{(\eta-1)t+\ell} +\\
& \Big(\sum_{i=\ell+1}^{t} p_{(\eta-1)t+i,h-k} \frac{\bar{c}^{i}_{(\eta-1)t+\ell} - \bar{c}^{\ell}_{(\eta-1)t+i}}{e_\eta - 1} \Big) + p_{\eta t+1,h-k} c^{\ell}_{\eta t+1} + \ldots + p_{k,h-k} c^{\ell}_k.
\end{aligned}$$

Table V shows the systematic transformed code with $n = 11$, $k = 6$, $r = 5$, $d = 8$ and $\eta = 2$.

The systematic transformed codes also satisfy Theorem 2 and Theorem 4, as the two transformed codes are equivalent. According to Theorem 2, each of the first six nodes in Table V has optimal

TABLE V: The systematic transformed codes with $n = 11$, $k = 6$, $r = 5$, $d = 8$ and $\eta = 2$.

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7
c_1^1	c_2^1	c_3^1	c_4^1	c_5^1	c_6^1	$p_{1,1}c_1^1 + p_{2,1}\frac{c_1^2-c_2^1}{e_1-1} + p_{3,1}\frac{c_1^3-c_3^1}{e_1-1} + p_{4,1}c_4^1 + p_{5,1}\frac{c_4^2-c_5^1}{e_2-1} + p_{6,1}\frac{c_4^3-c_6^1}{e_2-1}$
c_1^2	c_2^2	c_3^2	c_4^2	c_5^2	c_6^2	$p_{1,1}\frac{e_1c_2^1-c_1^2}{e_1-1} + p_{2,1}c_2^2 + p_{3,1}\frac{c_2^3-c_3^2}{e_1-1} + p_{4,1}\frac{e_2c_5^1-c_4^2}{e_2-1} + p_{5,1}c_5^2 + p_{6,1}\frac{c_5^3-c_6^2}{e_2-1}$
c_1^3	c_2^3	c_3^3	c_4^3	c_5^3	c_6^3	$p_{1,1}\frac{e_1c_3^1-c_1^3}{e_1-1} + p_{2,1}\frac{e_1c_3^2-c_2^3}{e_1-1} + p_{3,1}c_3^3 + p_{4,1}\frac{e_2c_6^1-c_4^3}{e_2-1} + p_{5,1}\frac{e_2c_6^2-c_5^3}{e_2-1} + p_{6,1}c_6^3$

Node 8	Node 9
$p_{1,2}c_1^1 + \sum_{i=2}^3 p_{i,2}\frac{c_1^i-c_i^1}{e_1-1} + p_{4,2}c_4^1 + \sum_{i=2}^3 p_{3+i,2}\frac{c_4^i-c_{3+i}^1}{e_2-1}$	$p_{1,3}c_1^1 + \sum_{i=2}^3 p_{i,3}\frac{c_1^i-c_i^1}{e_1-1} + p_{4,3}c_4^1 + \sum_{i=2}^3 p_{3+i,3}\frac{c_4^i-c_{3+i}^1}{e_2-1}$
$p_{1,2}\frac{e_1c_2^1-c_1^2}{e_1-1} + p_{2,2}c_2^2 + p_{4,2}\frac{e_2c_6^1-c_4^3}{e_2-1} + \sum_{i=1}^2 p_{3i,2}\frac{c_{3i-1}^3-c_{3i}^2}{e_i-1} + p_{5,2}c_5^2$	$p_{1,3}\frac{e_1c_2^1-c_1^2}{e_1-1} + p_{2,3}c_2^2 + p_{4,3}\frac{e_2c_6^1-c_4^3}{e_2-1} + \sum_{i=1}^2 p_{3i,3}\frac{c_{3i-1}^3-c_{3i}^2}{e_i-1} + p_{5,3}c_5^2$
$\sum_{i=1}^2 p_{i,2}\frac{e_1c_3^i-c_i^3}{e_1-1} + p_{3,2}c_3^3 + \sum_{i=1}^2 p_{3+i,2}\frac{e_2c_6^i-c_{3+i}^3}{e_2-1} + p_{6,2}c_6^3$	$\sum_{i=1}^2 p_{i,3}\frac{e_1c_3^i-c_i^3}{e_1-1} + p_{3,3}c_3^3 + \sum_{i=1}^2 p_{3+i,3}\frac{e_2c_6^i-c_{3+i}^3}{e_2-1} + p_{6,3}c_6^3$
Node 10	**Node 11**
$p_{1,4}c_1^1 + \sum_{i=2}^3 p_{i,4}\frac{c_1^i-c_i^1}{e_1-1} + p_{4,4}c_4^1 + \sum_{i=2}^3 p_{3+i,4}\frac{c_4^i-c_{3+i}^1}{e_2-1}$	$p_{1,5}c_1^1 + \sum_{i=2}^3 p_{i,5}\frac{c_1^i-c_i^1}{e_1-1} + p_{4,5}c_4^1 + \sum_{i=2}^3 p_{3+i,5}\frac{c_4^i-c_{3+i}^1}{e_2-1}$
$p_{1,4}\frac{e_1c_2^1-c_1^2}{e_1-1} + p_{2,4}c_2^2 + p_{4,4}\frac{e_2c_6^1-c_4^3}{e_2-1} + \sum_{i=1}^2 p_{3i,4}\frac{c_{3i-1}^3-c_{3i}^2}{e_i-1} + p_{5,4}c_5^2$	$p_{1,5}\frac{e_1c_2^1-c_1^2}{e_1-1} + p_{2,5}c_2^2 + p_{4,5}\frac{e_2c_6^1-c_4^3}{e_2-1} + \sum_{i=1}^2 p_{3i,5}\frac{c_{3i-1}^3-c_{3i}^2}{e_i-1} + p_{5,5}c_5^2$
$\sum_{i=1}^2 p_{i,4}\frac{e_1c_3^i-c_i^3}{e_1-1} + p_{3,4}c_3^3 + \sum_{i=1}^2 p_{3+i,4}\frac{e_2c_6^i-c_{3+i}^3}{e_2-1} + p_{6,4}c_6^3$	$\sum_{i=1}^2 p_{i,5}\frac{e_1c_3^i-c_i^3}{e_1-1} + p_{3,5}c_3^3 + \sum_{i=1}^2 p_{3+i,5}\frac{e_2c_6^i-c_{3+i}^3}{e_2-1} + p_{6,5}c_6^3$

repair access. For example, we can recover the three symbols on node 1 by accessing the following eight symbols

$$c_2^1, c_3^1, c_4^1, c_7^1, c_8^1, c_9^1, c_{10}^1, c_{11}^1.$$

Specifically, we can first subtract c_4^1 from c_7^1, c_8^1, c_9^1, c_{10}^1 and c_{11}^1, respectively, to obtain

$$\begin{bmatrix} p_{1,1}c_1^1 + p_{2,1}\frac{c_1^2-c_2^1}{e_1-1} + p_{3,1}\frac{c_1^3-c_3^1}{e_1-1} + p_{5,1}\frac{c_4^2-c_5^1}{e_2-1} + p_{6,1}\frac{c_4^3-c_6^1}{e_2-1} \\ p_{1,2}c_1^1 + p_{2,2}\frac{c_1^2-c_2^1}{e_1-1} + p_{3,2}\frac{c_1^3-c_3^1}{e_1-1} + p_{5,2}\frac{c_4^2-c_5^1}{e_2-1} + p_{6,2}\frac{c_4^3-c_6^1}{e_2-1} \\ p_{1,3}c_1^1 + p_{2,3}\frac{c_1^2-c_2^1}{e_1-1} + p_{3,3}\frac{c_1^3-c_3^1}{e_1-1} + p_{5,3}\frac{c_4^2-c_5^1}{e_2-1} + p_{6,3}\frac{c_4^3-c_6^1}{e_2-1} \\ p_{1,4}c_1^1 + p_{2,4}\frac{c_1^2-c_2^1}{e_1-1} + p_{3,4}\frac{c_1^3-c_3^1}{e_1-1} + p_{5,4}\frac{c_4^2-c_5^1}{e_2-1} + p_{6,4}\frac{c_4^3-c_6^1}{e_2-1} \\ p_{1,5}c_1^1 + p_{2,5}\frac{c_1^2-c_2^1}{e_1-1} + p_{3,5}\frac{c_1^3-c_3^1}{e_1-1} + p_{5,5}\frac{c_4^2-c_5^1}{e_2-1} + p_{6,5}\frac{c_4^3-c_6^1}{e_2-1} \end{bmatrix}$$

$$= \begin{bmatrix} c_1^1 & \frac{c_1^2-c_2^1}{e_1-1} & \frac{c_1^3-c_3^1}{e_1-1} & \frac{c_4^2-c_5^1}{e_2-1} & \frac{c_4^3-c_6^1}{e_2-1} \end{bmatrix} \cdot \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & p_{1,5} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} & p_{2,5} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} & p_{3,5} \\ p_{5,1} & p_{5,2} & p_{5,3} & p_{5,4} & p_{5,5} \\ p_{6,1} & p_{6,2} & p_{6,3} & p_{6,4} & p_{6,5} \end{bmatrix}.$$

As the square matrix in the above equation is invertible, we can compute the five symbols

$$c_1^1, \frac{c_1^2 - c_2^1}{e_1 - 1}, \frac{c_1^3 - c_3^1}{e_1 - 1}, \frac{c_4^2 - c_5^1}{e_2 - 1}, \frac{c_4^3 - c_6^1}{e_2 - 1},$$

from the above five symbols, and further obtain

$$c_1^1, c_1^2 - c_2^1, c_1^3 - c_3^1, c_4^2 - c_5^1, c_4^3 - c_6^1,$$

as $e_1 - 1$ is non-zero in the finite field. We have recovered c_1^1. Together with c_2^1 and c_3^1, we can recover c_1^2 and c_1^3 by

$$c_1^2 = (c_1^2 - c_2^1) + c_2^1,$$
$$c_1^3 = (c_1^3 - c_3^1) + c_3^1.$$

Similarly, we can recover node 2, node 3, node 4, node 5 and node 6 by accessing

$$c_1^2, c_3^2, c_5^2, c_7^2, c_8^2, c_9^2, c_{10}^2, c_{11}^2,$$

$$c_1^3, c_2^3, c_6^3, c_7^3, c_8^3, c_9^3, c_{10}^3, c_{11}^3,$$

$$c_1^1, c_5^1, c_6^1, c_7^1, c_8^1, c_9^1, c_{10}^1, c_{11}^1,$$

$$c_1^2, c_4^2, c_6^2, c_7^2, c_8^2, c_9^2, c_{10}^2, c_{11}^2,$$

and

$$c_3^3, c_4^3, c_5^3, c_7^3, c_8^3, c_9^3, c_{10}^3, c_{11}^3.$$

respectively.

IV. Multi-Layer Transformed MDS Codes with Optimal Repair Access

In this section, we present the construction of multi-layer transformed MDS codes by recursively applying the transformation given in Section III.

A. Construction

We divide n nodes into $\lceil \frac{n}{\eta t} \rceil$ sets, each set contains ηt nodes. For $i = 1, 2, \ldots, \lceil \frac{n}{\eta t} \rceil - 1$, set i contains nodes between $(i-1)\eta t + 1$ and $i\eta t$. Set $\lceil \frac{n}{\eta t} \rceil$ contains the last ηt nodes. We further divide each set into η groups, each group contains t nodes.

If we apply the transformation in Section III for the first set of an (n, k) MDS code, we can obtain a transformed code $\mathcal{C}_1(n, k, \eta, t)$ with each node having t symbols such that, according to Theorem 4, $\mathcal{C}_1(n, k, \eta, t)$ is an MDS code and has optimal repair access for the first ηt nodes according to Theorem 2. If we apply the transformation in Section III for the second set of

the code $\mathcal{C}_1(n,k,\eta,t)$, we can obtain the code $\mathcal{C}_2(n,k,\eta,t)$ with each node having $\alpha = t^2$ symbols. Specifically, we can obtain $\mathcal{C}_2(n,k,\eta,t)$ as follows. We first generate t instances of the code $\mathcal{C}_1(n,k,\eta,t)$ and view the t symbols stored in each node of $\mathcal{C}_1(n,k,\eta,t)$ as a vector. For $\ell = 1,2,\ldots,t$ and $h = 1,2,\ldots,n$, the vector stored in node h of instance ℓ of $\mathcal{C}_1(n,k,\eta,t)$ is denoted as $\mathbf{v}_h^\ell$. For $i = 1,2,\ldots,t$ and $j = 1,2,\ldots,\eta$, node $t\eta + (j-1)t + i$ of $\mathcal{C}_2(n,k,\eta,t)$ stores the following t vectors (t^2 symbols)

$$\begin{aligned}
&\mathbf{v}^1_{t\eta+(j-1)t+i} + \mathbf{v}^i_{t\eta+(j-1)t+1},\\
&\mathbf{v}^2_{t\eta+(j-1)t+i} + \mathbf{v}^i_{t\eta+(j-1)t+2},\ldots,\\
&\mathbf{v}^{i-1}_{t\eta+(j-1)t+i} + \mathbf{v}^i_{t\eta+(j-1)t+i-1},\\
&\mathbf{v}^i_{t\eta+(j-1)t+i},\\
&\mathbf{v}^{i+1}_{t\eta+(j-1)t+i} + e_{\eta+j}\mathbf{v}^i_{t\eta+(j-1)t+i+1},\\
&\mathbf{v}^{i+2}_{t\eta+(j-1)t+i} + e_{\eta+j}\mathbf{v}^i_{t\eta+(j-1)t+i+2},\ldots,\\
&\mathbf{v}^{t}_{t\eta+(j-1)t+i} + e_{\eta+j}\mathbf{v}^i_{t\eta+(j-1)t+t},
\end{aligned} \tag{10}$$

where $e_{\eta+j}$ is an element from the finite field except zero and one. Note that the multiplication of $e_{\eta+j}$ and a vector

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \ldots & v_t \end{bmatrix}$$

is defined as

$$e_{\eta+j}\mathbf{v} = \begin{bmatrix} e_{\eta+j}v_1 & e_{\eta+j}v_2 & \ldots & e_{\eta+j}v_t, \end{bmatrix}$$

and the addition of two vectors

$$\mathbf{v}^1 = \begin{bmatrix} v_1^1 & v_2^1 & \ldots & v_t^1 \end{bmatrix}$$

and

$$\mathbf{v}^2 = \begin{bmatrix} v_1^2 & v_2^2 & \ldots & v_t^2 \end{bmatrix}$$

is defined as

$$\mathbf{v}^1 + \mathbf{v}^2 = \begin{bmatrix} v_1^1 + v_1^2 & v_2^1 + v_2^2 & \ldots & v_t^1 + v_t^2 \end{bmatrix}.$$

For $h \in \{1,2,\ldots,n\} \setminus \{t\cdot\eta+1, t\cdot\eta+2,\ldots,2t\cdot\eta\}$, node h stores t vectors (t^2 symbols)

$$\mathbf{v}_h^1, \mathbf{v}_h^2, \ldots, \mathbf{v}_h^t.$$

According to Theorem 4, $\mathcal{C}_2(n,k,\eta,t)$ is an MDS code and, according to Theorem 2, has optimal repair access for the second ηt nodes.

In the example in Section II, we have $n = 8$, $k = 5$, $d = 6$ and $\eta = t = 2$. Table I shows two instances of $\mathcal{C}_1(n = 8, k = 5, \eta = 2, t = 2)$. We thus have

$$\begin{aligned}
\mathbf{v}_1^{2(\ell-1)+1} &= \left[c_1^{2(\ell-1)+1} \quad c_1^{2(\ell-1)+2} + e_1 c_2^{2(\ell-1)+1}\right], \\
\mathbf{v}^{2(\ell-1)+1} l_2 &= \left[c_2^{2(\ell-1)+1} + c_1^{2(\ell-1)+2} \quad c_2^{2(\ell-1)+2}\right], \\
\mathbf{v}_3^{2(\ell-1)+1} &= \left[c_3^{2(\ell-1)+1} \quad c_3^{2(\ell-1)+2} + e_2 c_4^{2(\ell-1)+1}\right], \\
\mathbf{v}_4^{2(\ell-1)+1} &= \left[c_4^{2(\ell-1)+1} + c_3^{2(\ell-1)+2} \quad c_4^{2(\ell-1)+2}\right], \\
\mathbf{v}_5^{2(\ell-1)+1} &= \left[c_5^{2(\ell-1)+1} \quad c_5^{2(\ell-1)+2}\right], \\
\mathbf{v}_6^{2(\ell-1)+1} &= \left[c_6^{2(\ell-1)+1} \quad c_6^{2(\ell-1)+2}\right], \\
\mathbf{v}_7^{2(\ell-1)+1} &= \left[c_7^{2(\ell-1)+1} \quad c_7^{2(\ell-1)+2}\right], \\
\mathbf{v}_8^{2(\ell-1)+1} &= \left[c_8^{2(\ell-1)+1} \quad c_8^{2(\ell-1)+2}\right],
\end{aligned}$$

where $\ell = 1, 2$. According to (10), the storage of the last four nodes is

$$\left[\begin{array}{c|cc}
\text{Node } 5 & \mathbf{v}_5^1 = [c_5^1, c_5^2] & \mathbf{v}_5^2 + e_3\mathbf{v}_6^1 = [c_5^3 + e_3 c_6^1, c_5^4 + e_3 c_6^2] \\ \hline
\text{Node } 6 & \mathbf{v}_6^1 + \mathbf{v}_5^2 = [c_6^1 + c_5^3, c_6^2 + c_5^4] & \mathbf{v}_6^2 = [c_6^3, c_6^4] \\ \hline
\text{Node } 7 & \mathbf{v}_7^1 = [c_7^1, c_7^2] & \mathbf{v}_7^2 + e_4\mathbf{v}_8^1 = [c_7^3 + e_4 c_8^1, c_7^4 + e_4 c_8^2] \\ \hline
\text{Node } 8 & \mathbf{v}_8^1 + \mathbf{v}_7^2 = [c_8^1 + c_7^3, c_8^2 + c_7^4] & \mathbf{v}_8^2 = [c_8^3, c_8^4]
\end{array}\right],$$

which is the same as the storage of the last four nodes in Table II.

Table VI shows the t symbols stored in each of the first ηt nodes of $\mathcal{C}_1(n, k, \eta, t)$. For $i = \eta t + 1, \ldots, n$, the t symbols stored in node i of $\mathcal{C}_1(n, k, \eta, t)$ are $c_i^1, c_i^2, \ldots, c_i^t$. Table VII shows the storage of the first $2\eta t$ nodes of $\mathcal{C}_2(n, k, \eta, t)$. In the next theorem, we show that the repair access of each of the first ηt nodes is also optimal.

Theorem 6. *The repair access of node i of $\mathcal{C}_2(n, k, \eta, t)$ for $i = 1, 2, \ldots, \eta t$ is optimal.*

Proof. For $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, \eta$, we can repair t^2 symbols in node $(j-1)t + i$ by downloading ηt^2 symbols from nodes $\eta t + 1, \eta t + 2, \ldots, 2\eta t$ in rows $i, i + t, \ldots, i + (t-1)t$, and $(k - \eta t)t$ symbols nodes $h_1, \ldots, h_{k-\eta t}$ in rows $i, i + t, \ldots, i + (t-1)t$ with indices $h_1, \ldots, h_k$ in

$$\{h_1, \ldots, h_{k-\eta t}\} \subset \{i, \ldots, (j-2)t + i, jt + i, \ldots, (\eta - 1)t + i, 2t\eta + 1, \ldots, n\} \tag{11}$$

TABLE VI: The storage of the first ηt nodes of $\mathcal{C}_1(n,k,\eta,t)$.

Node 1	Node 2	$\cdots$	Node t	$\cdots$	Node $(\eta-1)t+1$	Node $(\eta-1)t+2$	$\cdots$	Node ηt
c_1^1	$c_2^1+c_1^2$	$\cdots$	$c_t^1+c_1^t$	$\cdots$	$c_{(\eta-1)t+1}^1$	$c_{(\eta-1)t+2}^1+c_{(\eta-1)t+1}^2$	$\cdots$	$c_{\eta t}^1+c_{(\eta-1)t+1}^t$
$c_1^2+e_1c_2^1$	c_2^2	$\cdots$	$c_t^2+c_2^t$	$\cdots$	$c_{(\eta-1)t+1}^2+e_\eta c_{(\eta-1)t+2}^1$	$c_{(\eta-1)t+2}^2$	$\cdots$	$c_{\eta t}^2+c_{(\eta-1)t+2}^t$
$c_1^3+e_1c_3^1$	$c_2^3+e_1c_3^2$	$\cdots$	$c_t^3+c_3^t$	$\cdots$	$c_{(\eta-1)t+1}^3+e_\eta c_{(\eta-1)t+3}^1$	$c_{(\eta-1)t+2}^3+e_\eta c_{(\eta-1)t+3}^2$	$\cdots$	$c_{\eta t}^3+c_{(\eta-1)t+3}^t$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$c_1^t+e_1c_t^1$	$c_2^t+e_1c_t^2$	$\cdots$	c_t^t	$\cdots$	$c_{(\eta-1)t+1}^t+e_\eta c_{\eta t}^1$	$c_{(\eta-1)t+2}^t+e_\eta c_{\eta t}^2$	$\cdots$	$c_{\eta t}^t$

TABLE VII: The storage of the first $2\eta t$ nodes of $\mathcal{C}_2(n,k,\eta,t)$. In the table, $i=0,1,\ldots,\eta-1$.

Node 1	$\cdots$	Node t	$\cdots$	Node $(\eta-1)t+1$	$\cdots$	Node ηt
c_1^1	$\cdots$	$c_t^1+c_1^t$	$\cdots$	$c_{(\eta-1)t+1}^1$	$\cdots$	$c_{\eta t}^1+c_{(\eta-1)t+1}^t$
$c_1^2+e_1c_2^1$	$\cdots$	$c_t^2+c_2^t$	$\cdots$	$c_{(\eta-1)t+1}^2+e_\eta c_{(\eta-1)t+2}^1$	$\cdots$	$c_{\eta t}^2+c_{(\eta-1)t+2}^t$
$\vdots$	$\ddots$	$\vdots$	$\cdots$	$\vdots$	$\ddots$	$\vdots$
$c_1^t+e_1c_t^1$	$\cdots$	c_t^t	$\cdots$	$c_{(\eta-1)t+1}^t+e_\eta c_{\eta t}^1$	$\cdots$	$c_{\eta t}^t$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$c_1^{(t-1)t+1}$	$\cdots$	$c_t^{(t-1)t+1}+c_1^{(t-1)t+t}$	$\cdots$	$c_{(\eta-1)t+1}^{(t-1)t+1}$	$\cdots$	$c_{\eta t}^{(t-1)t+1}+c_{(\eta-1)t+1}^{t^2}$
$c_1^{(t-1)t+2}+e_1c_2^{(t-1)t+1}$	$\cdots$	$c_t^{(t-1)t+2}+c_2^{(t-1)t+t}$	$\cdots$	$c_{(\eta-1)t+1}^{(t-1)t+2}+e_\eta c_{(\eta-1)t+2}^{(t-1)t+1}$	$\cdots$	$c_{\eta t}^{(t-1)t+2}+c_{(\eta-1)t+2}^{t^2}$
$\vdots$	$\ddots$	$\vdots$	$\cdots$	$\vdots$	$\ddots$	$\vdots$
$c_1^{(t-1)t+t}+e_1c_t^{(t-1)t+1}$	$\cdots$	$c_t^{(t-1)t+t}$	$\cdots$	$c_{(\eta-1)t+1}^{(t-1)t+t}+e_\eta c_{\eta t}^{(t-1)t+1}$	$\cdots$	$c_{\eta t}^{t^2}$

Node $\eta t+it+1$	Node $\eta t+it+2$	$\cdots$	Node $\eta t+it+t$
$c_{\eta t+it+1}^1$	$c_{\eta t+it+2}^1+c_{\eta t+it+1}^{t+1}$	$\cdots$	$c_{\eta t+it+t}^1+c_{\eta t+it+1}^{(t-1)t+1}$
$c_{\eta t+it+1}^2$	$c_{\eta t+it+2}^2+c_{\eta t+it+1}^{t+2}$	$\cdots$	$c_{\eta t+it+t}^2+c_{\eta t+it+1}^{(t-1)t+2}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$c_{\eta t+it+1}^t$	$c_{\eta t+it+2}^t+c_{\eta t+it+1}^{2t}$	$\cdots$	$c_{\eta t+it+t}^t+c_{\eta t+it+1}^{t^2}$
$c_{\eta t+it+1}^{t+1}+e_{\eta+i+1}c_{\eta t+it+2}^1$	$c_{\eta t+it+2}^{t+1}$	$\cdots$	$c_{\eta t+it+t}^{t+1}+c_{\eta t+it+2}^{(t-1)t+1}$
$c_{\eta t+it+1}^{t+2}+e_{\eta+i+1}c_{\eta t+it+2}^2$	$c_{\eta t+it+2}^{t+2}$	$\cdots$	$c_{\eta t+it+t}^{t+2}+c_{\eta t+it+2}^{(t-1)t+2}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$c_{\eta t+it+1}^{2t}+e_{\eta+i+1}c_{\eta t+it+2}^t$	$c_{\eta t+it+2}^{2t}$	$\cdots$	$c_{\eta t+it+t}^{2t}+c_{\eta t+it+2}^{t^2}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$c_{\eta t+it+1}^{(t-1)t+1}+e_{\eta+i+1}c_{\eta t+it+t}^1$	$c_{\eta t+it+2}^{(t-1)t+1}+e_{\eta+i+1}c_{\eta t+it+t}^{t+1}$	$\cdots$	$c_{\eta t+it+t}^{(t-1)t+1}$
$c_{\eta t+it+1}^{(t-1)t+2}+e_{\eta+i+1}c_{\eta t+it+t}^2$	$c_{\eta t+it+2}^{(t-1)t+2}+e_{\eta+i+1}c_{\eta t+it+t}^{t+2}$	$\cdots$	$c_{\eta t+it+t}^{(t-1)t+2}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$
$c_{\eta t+it+1}^{t^2}+e_{\eta+i+1}c_{\eta t+it+t}^t$	$c_{\eta t+it+2}^{t^2}+e_{\eta+i+1}c_{\eta t+it+t}^{2t}$	$\cdots$	$c_{\eta t+it+t}^{t^2}$

and $(d-k)t$ symbols

$$\begin{aligned}
&c^i_{(j-1)t+1}+e_jc^1_{(j-1)t+i},\ldots,c^i_{(j-1)t+i-1}+e_jc^{i-1}_{(j-1)t+i},\\
&c^i_{(j-1)t+i+1}+c^{i+1}_{(j-1)t+i},\ldots,c^i_{(j-1)t+t}+c^t_{(j-1)t+i},\\
&c^{t+i}_{(j-1)t+1}+e_jc^{t+1}_{(j-1)t+i},\ldots,c^{t+i}_{(j-1)t+i-1}+e_jc^{t+i-1}_{(j-1)t+i},\\
&c^{t+i}_{(j-1)t+i+1}+c^{t+i+1}_{(j-1)t+i},\ldots,c^{t+i}_{(j-1)t+t}+c^{2t}_{(j-1)t+i},\\
&\vdots\\
&c^{(t-1)t+i}_{(j-1)t+1}+e_jc^{(t-1)t+1}_{(j-1)t+i},\ldots,c^{(t-1)t+i}_{(j-1)t+i-1}+e_jc^{(t-1)t+i-1}_{(j-1)t+i},\\
&c^{(t-1)t+i}_{(j-1)t+i+1}+c^{(t-1)t+i+1}_{(j-1)t+i},\ldots,c^{(t-1)t+i}_{(j-1)t+t}+c^{(t^2}_{(j-1)t+i}.
\end{aligned}\tag{12}$$

Note that

$$\begin{aligned}
\eta-1+(n-2t\eta)=&n-(2t-1)\lfloor\frac{n-k-1}{d-k}\rfloor-1\\
\geq&k-\eta t,
\end{aligned}$$

we can thus choose $k-\eta t$ different values in (11).

First, we claim that we can obtain the following kt symbols

$$\begin{aligned}
&c^i_{\eta t+1},c^i_{\eta t+2},\ldots,c^i_{2\eta t},c^i_{h_1},c^i_{h_2},\ldots,c^i_{h_{k-\eta t}},\\
&c^{i+t}_{\eta t+1},c^{i+t}_{\eta t+2},\ldots,c^{i+t}_{2\eta t},c^{i+t}_{h_1},c^{i+t}_{h_2},\ldots,c^{i+t}_{h_{k-\eta t}},\\
&\vdots\\
&c^{i+(t-1)t}_{\eta t+1},c^{i+(t-1)t}_{\eta t+2},\ldots,c^{i+(t-1)t}_{2\eta t},c^{i+(t-1)t}_{h_1},c^{i+(t-1)t}_{h_2},\ldots,c^{i+(t-1)t}_{h_{k-\eta t}},
\end{aligned}\tag{13}$$

from the downloaded kt symbols from nodes $\eta t+1,\eta t+2,\ldots,2\eta t$ and $h_1,\ldots,h_{k-\eta t}$. As $c^i_{h_j},c^{i+t}_{h_j},\ldots,c^{i+(t-1)t}_{h_j}$ are directly downloaded from node h_j for $j=1,2,\ldots,k-\eta t$, we only need to show that we can obtain

$$\begin{aligned}
&c^i_{\eta t+1},c^i_{\eta t+2},\ldots,c^i_{2\eta t},\\
&c^{i+t}_{\eta t+1},c^{i+t}_{\eta t+2},\ldots,c^{i+t}_{2\eta t},\\
&\vdots\\
&c^{i+(t-1)t}_{\eta t+1},c^{i+(t-1)t}_{\eta t+2},\ldots,c^{i+(t-1)t}_{2\eta t},
\end{aligned}\tag{14}$$

from the downloaded ηt^2 symbols from nodes $\eta t+1$ to $2\eta t$. Recall that the ηt^2 symbols

downloaded from nodes $\eta t+1$ to $2\eta t$ are

$$\begin{aligned}
&c^i_{\eta t+\ell t+1}, c^i_{\eta t+\ell t+2}+c^{t+i}_{\eta t+\ell t+1}, \ldots, c^i_{\eta t+\ell t+t}+c^{(t-1)t+i}_{\eta t+\ell t+1},\\
&c^{t+i}_{\eta t+\ell t+1}+e_{\eta+\ell+1}c^i_{\eta t+\ell t+2}, c^{t+i}_{\eta t+\ell t+2}, \ldots, c^{t+i}_{\eta t+\ell t+t}+c^{(t-1)t+i}_{\eta t+\ell t+2},\\
&\vdots,\\
&c^{(t-1)t+i}_{\eta t+\ell t+1}+e_{\eta+\ell+1}c^i_{\eta t+\ell t+t}, c^{(t-1)t+i}_{\eta t+\ell t+2}+e_{\eta+\ell+1}c^{t+i}_{\eta t+\ell t+t}, \ldots, c^{(t-1)t+i}_{\eta t+\ell t+t},
\end{aligned}$$

where $\ell=0,1,\ldots,\eta-1$. We can compute $c^i_{\eta t+\ell t+2}$ and $c^{t+i}_{\eta t+\ell t+1}$ from $c^i_{\eta t+\ell t+2}+c^{t+i}_{\eta t+\ell t+1}$ and $c^{t+i}_{\eta t+\ell t+1}+e_{\eta+\ell+1}c^i_{\eta t+\ell t+2}$ by Lemma 1. Similarly, we can compute all the symbols in (14) from the above symbols by Lemma 1. Therefore, we can obtain kt symbols in (13) and further compute $c_1^{i+\ell t}, c_2^{i+\ell t}, \ldots, c_t^{i+\ell t}$ with $\ell=0,1,\ldots,t-1$ according to the MDS property. With the computed $c_1^{i+\ell t}, c_2^{i+\ell t}, \ldots, c_t^{i+\ell t}$ with $\ell=0,1,\ldots,t-1$ and the accessed $(d-k)t$ symbols in (12), we can compute all t symbols stored in node $(j-1)t+i$ by Lemma 1. Therefore, we can recover t^2 symbols in node $(j-1)t+i$ by downloading td symbols from d helper nodes and the repair access is optimal according to . □

According to Theorem 2 and Theorem 6, the codes $\mathcal{C}_2(n,k,\eta,t)$ have optimal repair access for the first $2\eta t$ nodes. By recursively applying the transformation for set $i+1$ of the codes $\mathcal{C}_i(n,k,\eta,t)$ for $i=1,2,\ldots,\lceil\frac{n}{\eta t}\rceil-1$, we can obtain the codes $\mathcal{C}_{\lceil\frac{n}{\eta t}\rceil}(n,k,\eta,t)$ that satisfy the MDS property according to Theorem 4 and have optimal repair access for all n nodes according to Theorem 2 and Theorem 6.

B. Repair Method

For $i=1,2,\ldots,t$, $j=1,2,\ldots,\eta$ and $\ell=1,2,\ldots,\lceil\frac{n}{\eta t}\rceil$, we can repair $t^{\lceil\frac{n}{\eta t}\rceil}$ symbols in node $(\ell-1)\eta t+(j-1)t+i$ by downloading the symbols in row f for

$$f \bmod (t^\ell) \in \{(i-1)(t^{\ell-1})+1, (i-1)(t^{\ell-1})+2, \ldots, i(t^{\ell-1})-1\}$$

from $\eta-1$ nodes $(\ell-1)\eta t+i, \ldots, (\ell-1)\eta t+(j-2)t+i, (\ell-1)\eta t+jt+i, \ldots, (\ell-1)\eta t+(\eta-1)t+i$, $t-1$ nodes $(j-1)t+1, \ldots, (j-1)t+i-1, (j-1)t+i+1, \ldots, (j-1)t+t$ and $d-\eta-t+2$ nodes $h_1, h_2, \ldots, h_{d-\eta-t+2}$. For $i=1,2,\ldots,d-\eta-t+2$, if h_i belongs to a group in set μ with $\mu>\ell$, then all t nodes of the group should be chosen in the helper nodes $h_1, h_2, \ldots, h_{d-\eta-t+2}$. With the same argument of the proof of Theorem 6, we can show that node $(\ell-1)\eta t+(j-1)t+i$

can be repaired by the above repair method with repair access being $d \cdot t^{\lceil \frac{n}{nt} \rceil - 1}$, which is optimal according to (1).

The example given in Section II is the code $\mathcal{C}_2(n, k, \eta, t)$ with $n = 8$, $k = 5$, $d = 6$ and $\eta = 2$, which is shown in Table II. Suppose that node 1 fails, i.e., $i = 1$, $j = 1$ and $\ell = 1$. According to the above repair method, we can repair node 1 by downloading the symbols in row f for $f \bmod 2 \in \{1\}$ from nodes $2, 3, 5, 6, 7, 8$, i.e.,

$$c_2^1 + c_1^2, c_3^1, c_5^1, c_6^1 + c_5^3, c_7^1, c_8^1 + c_7^3,$$
$$c_2^3 + c_1^4, c_3^3, c_5^3 + e_3 c_6^1, c_6^3, c_7^3 + e_4 c_8^1, c_8^3.$$

The detailed repair procedure of node 1 is illustrated in Fig. 1 When $i = 1$, $j = 2$ and $\ell = 1$, we can repair node 3 by downloading the symbols in rows 1 and 3 from nodes $1, 4, 5, 6, 7, 8$. When $i = 2$, $j = 1$ and $\ell = 1$, we can repair node 2 by downloading the symbols in rows 2 and 4 from nodes $1, 4, 5, 6, 7, 8$. When $i = 2$, $j = 2$ and $\ell = 1$, we can repair node 2 by downloading the symbols in rows 2 and 4 from nodes $2, 3, 5, 6, 7, 8$. According to the above repair method, we can repair node 5 and node 7 by downloading the symbols in rows $1, 2$ from nodes $1, 2, 3, 4, 6, 7$ and $1, 2, 3, 4, 5, 8$, respectively, and repair node 6 and node 8 by downloading the symbols in rows $3, 4$ from nodes $1, 2, 3, 4, 5, 8$ and $1, 2, 3, 4, 6, 7$, respectively.

V. Transformation for Binary MDS Array Codes

In this section, we present the method of applying the proposed transformation for binary MDS array codes, using EVENODD codes as a motivating example, to obtain the transformed codes that have optimal repair access.

A. EVENODD Codes

An EVENODD code can be presented by a $(p-1) \times (k+r)$ array $[a_{i,j}]$ for $i = 0, 1, \ldots, p-2$ and $j = 0, 1, \ldots, k+r-1$. The first k columns are information columns that store information bits and the last r columns are parity columns that store parity bits. For $j = 0, 1, \ldots, k+r-1$, we represent the $p-1$ bits $a_{0,j}, a_{1,j}, \ldots, a_{p-2,j}$ in column j by the polynomial

$$a_j(x) = a_{0,j} + a_{1,j}x + \ldots + a_{p-2,j}x^{p-2}.$$

The first k polynomials $a_0(x), \ldots, a_{k-1}(x)$ are information polynomials, and the last r polynomials $a_k(x), \ldots, a_{k+r-1}(x)$ are parity polynomials that are computed as

$$\begin{bmatrix} a_k(x) & \cdots & a_{k+r-1}(x) \end{bmatrix} = \begin{bmatrix} a_0(x) & \cdots & a_{k-1}(x) \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & x & \cdots & x^{r-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x^{k-1} & \cdots & x^{(r-1)(k-1)} \end{bmatrix}$$

over the ring $\mathbb{F}_2[x]/(1 + x + \cdots + x^{p-1})$. Given parameters k and r, we need to choose the parameter p such that the MDS property that is any k out of $k + r$ columns can retrieve all information bits stored in the k information columns should be satisfied. The MDS property condition of EVENODD codes is given in [26], [27].

B. Transformation for EVENODD Codes

We will present how to apply the proposed transformation in Section III for EVENODD codes such that the transformed EVENODD codes have optimal repair access for any chosen $(d - k + 1)\eta$ columns.

The transformed EVENODD code is an array code of size $(p-1)(d-k+1) \times (k+r)$. Given the $(p-1)(d-k+1) \times k$ information array

$$\left[\begin{array}{cccc} a^0_{0,0} & a^0_{0,1} & \cdots & a^0_{0,k-1} \\ a^0_{1,0} & a^0_{1,1} & \cdots & a^0_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ a^0_{p-2,0} & a^0_{p-2,1} & \cdots & a^0_{p-2,k-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline a^{d-k}_{0,0} & a^{d-k}_{0,1} & \cdots & a^{d-k}_{0,k-1} \\ a^{d-k}_{1,0} & a^{d-k}_{1,1} & \cdots & a^{d-k}_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ a^{d-k}_{p-2,0} & a^{d-k}_{p-2,1} & \cdots & a^{d-k}_{p-2,k-1} \end{array}\right],$$

we first represent each $p - 1$ information bits $a^\ell_{0,j}, a^\ell_{1,j}, \ldots, a^\ell_{p-2,j}$ by the *information polynomial*

$$a^\ell_j(x) = a^\ell_{0,j} + a^\ell_{1,j}x + \ldots + a^\ell_{p-2,j}x^{p-2},$$

and then compute $(d-k+1)n$ *coded polynomials* by

$$\begin{bmatrix} a_k^\ell(x) & \cdots & a_{k+r-1}^\ell(x) \end{bmatrix} = \begin{bmatrix} a_0^\ell(x) & \cdots & a_{k-1}^\ell(x) \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & x & \cdots & x^{r-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x^{k-1} & \cdots & x^{(r-1)(k-1)} \end{bmatrix}$$

over the ring $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$. Recall that $t=d-k+1$ and $\eta=\lfloor\frac{r-1}{d-k}\rfloor$. For $i=1,2,\ldots,t$ and $j=1,2,\ldots,\eta$, column $(j-1)t+i$ stores the following t polynomials

$$\begin{aligned} &a^1_{(j-1)t+i}(x)+a^i_{(j-1)t+1}(x),\\ &a^2_{(j-1)t+i}(x)+a^i_{(j-1)t+2}(x),\ldots,\\ &a^{i-1}_{(j-1)t+i}(x)+a^i_{(j-1)t+i-1}(x),\\ &a^i_{(j-1)t+i}(x),\\ &a^{i+1}_{(j-1)t+i}(x)+e_j(x)a^i_{(j-1)t+i+1}(x),\\ &a^{i+2}_{(j-1)t+i}(x)+e_j(x)a^i_{(j-1)t+i+2}(x),\ldots,\\ &a^t_{(j-1)t+i}(x)+e_j(x)a^i_{(j-1)t+t}(x), \end{aligned} \tag{15}$$

where $e_j(x)$ is a non-zero polynomial in $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$ such that both $e_j(x)$ and $e_j(x)+1$ are invertible in $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$. For $h=t\cdot\eta+1,t\cdot\eta+2,\ldots,n$, column h stores t polynomials

$$a_h^1(x),a_h^2(x),\ldots,a_h^t(x).$$

The obtained above codes are called *transformed EVENODD codes*. The transformation in [16] can be viewed as a special case of our transformation with $\eta=1$. Note that the transformed codes in Section III and the transformed EVENODD codes are essentially the same codes, the difference is that the transformed codes in Section III are operated over the finite field $\mathbb{F}_q$ and the transformed EVENODD codes are operated over the ring $\mathbb{F}_2[x]/(1+x+\cdots+x^{p-1})$.

With the same proof of Theorem 4, we can show that the transformed EVENODD codes satisfy the MDS property if EVENODD codes satisfy the MDS property and p is large enough. We can also show that the repair access of each of the first ηt columns is optimal, as like the transformed codes in Section III. By recursively applying the transformation for EVENODD codes for $\lceil\frac{n}{\eta t}\rceil$ times, we can obtain the transformed $\mathsf{EVENODD}_{\lceil\frac{n}{\eta t}\rceil}(n,k,\eta,t)$ codes that have optimal repair access for all columns and satisfy the MDS property when p is large enough.

TABLE VIII: Comparison of existing MDS codes with optimal repair access.

MDS codes	d	Sub-packetization α	No. of nodes with optimal repair
The first codes in [8]	$n-1$	r^n	n
The second codes in [8]	$n-1$	r^{n-1}	n
Codes in [9]	any d	$(d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$	n
Codes in [32]	any d	$(d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$	n
Codes in [6]	$n-1$	$r^{\lceil \frac{n}{r} \rceil}$	n
[10, Corollary 4]	$n-1$	$\geq r^{\lceil \frac{n}{r} \rceil}$	n
[10, Corollary 6]	any d	$\geq (d-k+1)^{\lceil \frac{n-1}{d-k+1} \rceil}$	n
[10, Corollary 5]	any d	$\geq (d-k+1)^{\lceil \frac{w}{d-k+1} \rceil}$	$k-1 < w \leq n$
Our $\mathcal{C}_{\lceil \frac{n}{\eta t} \rceil}(n,k,\eta,t)$	any d	$(d-k+1)^{\lceil \frac{n}{\eta(d-k+1)} \rceil}$	n
Our $\mathcal{C}_{\lceil \frac{n}{2\eta} \rceil}(n,k,\eta,t=2)$	$d=k+1$	$2^{\lceil \frac{n}{2\eta} \rceil}$	n
Our $\mathcal{C}_{\lceil \frac{w}{\eta t} \rceil}(n,k,\eta,t)$	any d	$(d-k+1)^{\lceil \frac{w}{\eta(d-k+1)} \rceil}$	w
Our $\mathcal{C}_{\lceil \frac{w}{2\eta} \rceil}(n,k,\eta,t=2)$	$d=k+1$	$2^{\lceil \frac{w}{2\eta} \rceil}$	$k-1 < w \leq n$

C. Transformation for Other Binary MDS Array Codes

The transformation can also be employed in other binary MDS array codes, such as the codes in [19], [21]–[23], [28], [29], to enable optimal repair access for all columns.

Specifically, the transformation for RDP and codes in [19], [28], [29] is similar to the transformation for EVENODD codes in Section V-B. By applying the transformation for RDP for $\lceil \frac{n}{\eta t} \rceil$ times, we can show that the obtained $\mathsf{RDP}_{\lceil \frac{n}{\eta t} \rceil}(n,k,\eta,t)$ codes have optimal repair access for all columns, as in $\mathsf{EVENODD}_{\lceil \frac{n}{\eta t} \rceil}(n,k,\eta,t)$ codes.

Recall that the codes in [21]–[23], [30], [31] have optimal repair access or efficient optimal repair access for information column, we only need to apply the transformation for them for $\lceil \frac{n}{\eta t} \rceil$ times to obtain the transformed codes with lower sub-packetization that have optimal repair access for all parity columns and optimal repair access or efficient repair access for the information column. Note that to preserve the efficient repair property of the information column, we need to carefully design the transformation for different codes in [21]–[23], [30], [31], and that will be one of our future work.

TABLE IX: Sub-packetization of our codes and the codes in [9], [32] for some parameters.

Parameters (n,k,d)	$(14,10,11)$	$(12,8,9)$	$(18,14,15)$	$(18,13,15)$	$(24,19,21)$	$(80,71,72)$
α in [9], [32]	128	64	512	729	6561	$2^{40}=1099511627776$
α of our codes	8	4	8	27	81	1024

VI. Comparison

A comparison of existing MDS codes with optimal repair access and the proposed codes $\mathcal{C}_{\lceil \frac{n}{\eta t}\rceil}(n,k,\eta,t)$ is given in Table VIII. The results in Table VIII show that our codes $\mathcal{C}_{\lceil \frac{n}{\eta t}\rceil}(n,k,\eta,t)$ have two advantages: (i) when $\lfloor \frac{r-1}{d-k} \rfloor > 1$, the obtained codes $\mathcal{C}_{\lceil \frac{n}{\eta t}\rceil}(n,k,\eta,t)$ with optimal repair access for all nodes have lower sub-packetization level compared to both the existing MDS codes in [9], [32] with optimal repair access for all nodes and the lower bound of sub-packetization level of MDS codes with optimal repair access for all nodes in [10, Corollary 6]; (ii) when $\lfloor \frac{r-1}{d-k} \rfloor > 1$, the codes $\mathcal{C}_{\lceil \frac{w}{\eta t}\rceil}(n,k,\eta,t)$ with optimal repair access for w nodes have less sub-packetization level than the tight lower bound on the sub-packetization level of MDS codes with optimal repair access for w nodes in [10, Corollary 5]. Table IX shows the sub-packetization of our codes and the codes in [9], [32] for some parameters.

Compared with the existing binary MDS array codes with optimal repair access for columns, the proposed $\mathsf{EVENODD}_{\lceil \frac{n}{\eta t}\rceil}(n,k,\eta,t)$ codes have less sub-packetization. Recall that the sub-packetization of both the transformed EVENODD codes in [16] and the transformed codes in is $(d-k+1)^{\lceil \frac{n}{d-k+1}\rceil}$, which is strictly larger than the sub-packetization of $\mathsf{EVENODD}_{\lceil \frac{n}{\eta t}\rceil}(n,k,\eta,t)$, if $\lceil \frac{r-1}{d-k} \rceil \geq 2$.

VII. Conclusions and Future Work

In this paper, we propose a generic transformation for MDS codes that can enable optimal repair access for any chosen $(d-k+1)\eta$ nodes. By applying the proposed transformation for any MDS codes for $\lceil \frac{n}{\eta t} \rceil$ times, we can obtain the transformed codes that have optimal repair access for all nodes. With a slightly modification, we also present the transformation for binary MDS array codes, using EVENODD codes as a motivating example. We show that the $\mathsf{EVENODD}_{\lceil \frac{n}{\eta t}\rceil}(n,k,\eta,t)$ codes obtained by applying the transformation for EVENODD codes

for $\lceil \frac{n}{\eta t} \rceil$ times have optimal repair access for all columns. Moreover, the proposed transformed codes have less sub-packetization than that of the existing MDS codes with optimal repair access for all nodes. How to design a specific transformation for binary MDS codes in with efficient repair access, to obtain the transformed codes with lower sub-packetization that have optimal repair access for all parity columns and efficient repair access for all information columns is one of our future work. When more than one columns fail, how to retrieve all the information bits from any k of the surviving nodes with lower computational complexity is the decoding problem. How to design MDS codes with lower sub-packetization, efficient repair access for all nodes and lower decoding complexity is another future work.

REFERENCES

[1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[2] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction," *IEEE Trans. Information Theory*, vol. 57, no. 8, pp. 5227–5239, August 2011.

[3] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC Codes: Low-Complexity Regenerating Codes for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 62, no. 6, pp. 3053–3069, 2016.

[4] C. Suh and K. Ramchandran, "Exact-Repair MDS Code Construction Using Interference Alignment," *IEEE Trans. Information Theory*, vol. 57, no. 3, pp. 1425–1442, 2011.

[5] I. Tamo, Z. Wang, and J. Bruck, "Zigzag Codes: MDS Array Codes with Optimal Rebuilding," *IEEE Trans. Information Theory*, vol. 59, no. 3, pp. 1597–1616, May 2013.

[6] J. Li, X. Tang, and C. Tian, "A Generic Transformation to Enable Optimal Repair in MDS Codes for Distributed Storage Systems," *IEEE Trans. Information Theory*, vol. 64, no. 9, pp. 6257–6267, 2018.

[7] S. Goparaju, A. Fazeli, and A. Vardy, "Minimum Storage Regenerating Codes for All Parameters," *IEEE Trans. Information Theory*, vol. 63, no. 10, pp. 6318–6328, 2017.

[8] M. Ye and A. Barg, "Explicit Constructions of High-Rate MDS Array Codes with Optimal Repair Bandwidth," *IEEE Trans. Information Theory*, vol. 63, no. 4, pp. 2001–2014, 2017.

[9] ——, "Explicit Constructions of Optimal-Access MDS Codes with Nearly Optimal Sub-Packetization," *IEEE Trans. Information Theory*, p. 63076317, 2017.

[10] S. Balaji and P. Kumar, "A Tight Lower Bound on the Sub-Packetization Level of Optimal-Access MSR and MDS Codes," *arXiv:1710.05876v1*, 2017.

[11] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures," *IEEE Trans. on Computers*, vol. 44, no. 2, pp. 192–202, 1995.

[12] M. Blaum, J. Brady, J. Bruck, J. Menon, and A. Vardy, "The EVENODD Code And Its Generalization," *High Performance Mass Storage and Parallel I/O*, pp. 187–208, 2001.

[13] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding," *IEEE Trans. Information Theory*, vol. 45, no. 1, pp. 272–276, 1999.

[14] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-Diagonal Parity for Double Disk Failure Correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, 2004, pp. 1–14.

[15] M. Blaum, "A Family of MDS Array Codes with Minimal Number of Encoding Operations," in *IEEE Int. Symp. on Inf. Theory*, 2006, pp. 2784–2788.

[16] H. Hou and P. P. C. Lee, "Binary MDS Brray Codes with Optimal Repair," *arXiv preprint: arXiv:1809.04380*, 2018.

[17] Z. Huang, H. Jiang, and K. Zhou, "An Improved Decoding Algorithm for Generalized RDP Codes," *IEEE Communications Letters*, vol. 20, no. 4, pp. 632–635, 2016.

[18] H. Hou and P. P. C. Lee, "A New Construction of EVENODD Codes with Lower Computational Complexity," *IEEE Communications Letters*, vol. 22, no. 6, pp. 1120–1123, 2018.

[19] H. Hou and Y. S. Han, "A New Construction and an Efficient Decoding Method for Rabin-Like Codes," *IEEE Trans. Communications*, vol. 66, no. 2, pp. 521–533, 2018.

[20] H. Hou, Y. S. Han, K. W. Shum, and H. Li, "A Unified Form of EVENODD and RDP Codes and Their Efficient Decoding," *IEEE Trans. Communications*, pp. 1–1, 2018.

[21] H. Hou, P. P. C. Lee, Y. S. Han, and Y. Hu, "Triple-Fault-Tolerant Binary MDS Array Codes with Asymptotically Optimal Repair," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, June 2017.

[22] H. Hou and Y. S. Han, "A Class of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *accepted in SCIENCE CHINA Information Sciences http://engine.scichina.com/doi/10.1007/s11432-018-9485-7*, vol. 61, no. 10, pp. 1–12, 2018.

[23] H. Hou, Y. S. Han, P. P. C. Lee, Y. Hu, and H. Li, "A New Design of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *arXiv preprint https://arxiv.org/pdf/1802.07891.pdf*, 2018.

[24] J. Li and X. Tang, "A Note on the Transformation to Enable Optimal Repair in MDS Codes for Distributed Storage Systems," *arXiv preprint: arXiv:1901.06067v1*, 2019.

[25] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[26] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," *IEEE Trans. Information Theory*, vol. 42, no. 2, pp. 529–542, 1996.

[27] H. Hou, K. W. Shum, and H. Li, "On the MDS Condition of Blaum-Bruck-Vardy Codes With Large Number Parity Columns," *IEEE Communications Letters*, vol. 20, no. 4, pp. 644–647, 2016.

[28] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-Based Erasure-Resilient Coding Scheme," *Int. Comput. Sci. Inst., Berkeley, CA, USA, Tech. Rep. TR-95-048*, 1995.

[29] G.-L. Feng, R. H. Deng, F. Bao, and J.-C. Shen, "New Efficient MDS Array Codes for RAID. Part II. Rabin-Like Codes for Tolerating Multiple (≥ 4) Disk Failures," *IEEE Trans. on Computers*, vol. 54, no. 12, pp. 1473–1483, 2005.

[30] E. E. Gad, R. Mateescu, F. Blagojevic, C. Guyot, and Z. Bandic, "Repair-Optimal MDS Array Codes over GF(2)," in *Proc. IEEE Int. Symp. Inf. Theory*, 2013, pp. 887–891.

[31] L. Pamies-Juarez, F. Blagojevic, R. Mateescu, C. Guyot, E. E. Gad, and Z. Bandic, "Opening the Chrysalis: On the Real Repair Performance of MSR Codes," in *Proc. of USENIX FAST*, 2016, pp. 81–94.

[32] M. Vajha, V. Ramkumar, B. Puranik, G. Kini, E. Lobo, B. Sasidharan, P. V. Kumar, A. Barg, M. Ye, S. Narayanamurthy, S. Hussain, and S. Nandi, "Clay Codes: Moulding MDS Codes to Yield an MSR Code," in *16th USENIX Conference on*

File and Storage Technologies (FAST 18). Oakland, CA: USENIX Association, 2018, pp. 139–154. [Online]. Available: https://www.usenix.org/conference/fast18/presentation/vajha