Scalar MSCR Codes via the Product Matrix Construction

Yaqian Zhang and Zhifang Zhang

Abstract—An (n, k, d) cooperative regenerating code provides the optimal-bandwidth repair for any t (t > 1) node failures in a cooperative way. In particular, an MSCR (minimum storage cooperative regenerating) code retains the same storage overhead as an (n, k) MDS code. Suppose each node stores α symbols which indicates the sub-packetization level of the code. A scalar MSCR code attains the minimum sub-packetization, i.e., \( \alpha = d - k + 1 \). By now, all existing constructions of scalar MSCR codes restrict to very special parameters, e.g., \( d = k \) or \( k = 2 \), etc. In a recent work, Ye and Barg construct MSCR codes for all \( n, k, d, t \), but however, their construction needs \( \alpha \approx \exp(n^\gamma) \) which is almost infeasible in practice. In this paper, we give an explicit construction of scalar MSCR codes for all \( d \geq \max(2k - 2 - t, k) \), which covers all possible parameters except the case of \( k \leq d \leq 2k - 2 - t \) when \( k < 2k - 1 - t \). Moreover, as a complementary result, for \( k < d < 2k - 2 - t \) we prove the nonexistence of linear scalar MSCR codes that have invariant repair spaces. Our construction and most of the previous scalar MSCR codes all have invariant repair spaces and this property is appealing in practice because of convenient repair. In this sense, this work presents an almost full description of usual scalar MSCR codes.

Index Terms—Regenerating code, cooperative repair, product matrix.

I. INTRODUCTION

A CENTRAL issue in large-scale distributed storage systems (DSS) is the efficient repair of node failures. Suppose a data file is stored across \( n \) nodes such that a data collector can retrieve the original file by reading the contents of any \( k \) nodes. When some node fails, a self-sustaining storage system tends to regenerate the failed node by downloading data from surviving nodes. An important metric for the node repair efficiency is the repair bandwidth, namely, the total amount of data downloaded during the repair process. In a celebrated work [1], Dimakis et al. proposed regenerating codes which achieve the tradeoff (i.e. the cut-set bound) between the repair bandwidth and the storage overhead. In particular, regenerating codes with the minimum storage and with the minimum bandwidth (i.e., MSR codes and MBR codes) are extensively studied in a series of works [2]–[7].

Regenerating codes typically deal with single node failures, however, the scenarios of multiple node failures are quite common in DSS. For example, in Total Recall [8] a repair process is triggered only after the total number of failed nodes has reached a predefined threshold. There are two typical models for repairing multiple node failures. One is the centralized repair where a special node called data center is assumed to complete all repairs. The other is the cooperative repair where all new nodes are generated in a distributed and cooperative way. It was proved in [9] that an MDS code achieving the optimal bandwidth in the cooperative repair model also achieves the optimal bandwidth in the centralized repair model. In this sense, the cooperative repair is a stronger model than the centralized repair. Moreover, due to the distributed pattern, the cooperative repair fits DSS better than the centralized repair. The centralized repair for regenerating codes is studied in [6], [10], and [11], while this paper is dedicated to the cooperative repair.

The idea of cooperative repair was proposed by Hu et al. [12]. Specifically, suppose \( t \) new nodes (i.e. newcomers) are to be generated as replacements of \( t \) failed nodes respectively. The regenerating process is carried out in two phases. Firstly, each newcomer connects to \( d \) (\( d \geq k^1 \)) surviving nodes (i.e. helper nodes) and downloads \( \beta_1 \) symbols from each. Note that different newcomers may choose different \( d \) helper nodes. Secondly, each newcomer downloads \( \beta_2 \) symbols from each of the other \( t - 1 \) newcomers. Therefore, the bandwidth for repairing one failed node is \( \gamma = d\beta_1 + (t - 1)\beta_2 \). In [12] and [14] a cut-set bound was derived for the cooperative repair, i.e.,

\[
B \leq \sum_{i=1}^{t} l_i \min\{\alpha, (d - \sum_{h=1}^{i-1} l_h)\beta_1 + (t - l_i)\beta_2\},
\]

where \( B \) is the size of the original data file, \( \alpha \) is the size of data stored in each node (also called the sub-packetization level), and \( l_1, \ldots, l_s, s \) are the integers satisfying \( l_1 + \cdots + l_s = k \) and \( 1 \leq l_1, \ldots, l_s \leq t \). In more detail, the bound (1) is derived from an information flow graph where \( k \) nodes are connected by the data collector in \( s \) stages and each \( l_i \), \( 1 \leq i \leq s \), represents the number of connected nodes in the \( i \)-th stage. The cut-set bound (1), along with other necessary conditions on the parameters, define a \( \alpha-\gamma \) tradeoff curve.

1In regenerating codes, it always assumes that \( d \geq k \) because \( d \) helper nodes are sufficient to repair any other node in the system, thus they are enough to recover the original data file, however, it is predefined that the data file can be recovered by connecting at least \( k \) nodes.
and codes with parameters lying on this curve are called cooperative regenerating codes. In particular, the two extreme points on the tradeoff curve respectively correspond to the MBCR (minimum bandwidth cooperative regenerating) codes and the MSCR (minimum storage cooperative regenerating) codes. Specifically, it can be computed that for MBCR codes,
\[
\alpha = \frac{B(2d + t - 1)}{k(2d - k + t)}, \quad \beta_1 = 2\beta_2 = \frac{2B}{k(2d - k + t)},
\]
and for MSCR codes,
\[
\alpha = \frac{B}{k}, \quad \beta_1 = \beta_2 = \frac{B}{k(d - k + t)}.
\]
Most of the studies on cooperative regenerating codes are concerned with the two extreme points.

It can be seen from (2) and (3) that for any fixed parameters \(n, k, d \geq k, t\), the rest parameters \(B, \alpha, \beta_1\) are all integral multiples of \(\beta_2\), which implies that an MBCR or MSCR code \(C_0\) with parameters \((B, \alpha, \beta_1, \beta_2 = 1)\) can be easily extended to the parameters \((mB, ma, m\beta_1, \beta_2 = m)\) for any \(m \in \mathbb{N}\) by dividing the data file into \(m\) stripes and applying \(C_0\) to each stripe. In other words, the code with \(\beta_2 = 1\) can serve as a building block for constructing cooperative regenerating codes. Adopting the definition in [15], we call the code with \(\beta_2 = 1\) as a scalar cooperative regenerating code. Unfortunately, scalar regenerating codes may not exist for some \(n, k, d, t\). For example, in the MSR case (i.e., \(t = 1\)), it is proved in [19] that linear scalar MSR codes do not exist for \(d < 2k - 3\). However, things are optimistic for the MBR and MBCR codes because the scalar codes have been explicitly constructed for all \(n, k, d, t\) in [3] and [13] respectively. Therefore, in this work we focus on the construction of MSCR codes, especially on scalar MSCR codes.

In [14] Shum et al. first explicitly constructed scalar MSCR codes for \(d = k\). Then in [15] the author built scalar MSCR codes for the case \(k = 2\) and \(d = n - t\). Later, Chen and Shum [16] designed a scalar \((n = 2k, k, d = 2k - 2, t = 2)\) MSCR code, and then they generalized the construction to a scalar \((n = 2k, k, d = n - t, 2 \leq t \leq n - k)\) MSCR code where the failed nodes must be systematic nodes [17]. Although these existing constructions of MSCR codes are all scalar, an obvious drawback is that they all restrict to very limited parameters. Until recently, Ye and Barg [9] gave an explicit construction of MSCR codes (not scalar) for all admissible \(n, k, d, t\). However, the sub-packetization of their codes is extraordinarily large, i.e., \(a = \{(d - k + t)(d - k)^{t-1}\} \approx \exp(br)\). By contrast, one can see scalar MSCR codes have the sub-packetization \(a = d - k + t\) by letting \(\beta_2 = 1\) in (3). Since the sub-packetization directly determines the smallest file size (i.e., \(B = ka\)) that is admissible for the code implementation, and also greatly influences the computational complexity, Ye and Barg’s MSCR codes are almost infeasible in practice.

In this paper, we present an explicit construction of scalar MSCR codes for all \(d \geq \max(2k - 1 - t, k)\), which almost covers all parameters except for an augment of \(d\) when \(k > t + 1\). More specifically, when \(2k - 1 - t \leq k\), i.e., \(k \leq t + 1\), our construction applies to all \(d \geq k\) which covers all admissible \(n, k, d, t\) for cooperative regenerating codes. When \(2k - 1 - t > k\), i.e., \(k > t + 1\), our construction restricts to the case \(d \geq 2k - 1 - t\). Furthermore, for the latter case, we continue to prove that there exist no linear scalar MSCR codes with invariant repair space for \(k < d < 2k - 2 - t\). It’s worth noting that the nonexistence result is restricted to the scalar MSCR codes with invariant repair spaces, while for general scalar MSCR codes in the range \(k < d \leq 2k - 2 - t\) no results have been given so far. Moreover, our construction can be viewed as an extension of the product matrix construction for MSR codes proposed by Rashmi et al. [3] to cooperative regenerating codes.

The remaining of this paper is organized as follows. First, a brief introduction to the product matrix framework is given in Section II. Then the construction of MSCR codes with any \(2 \leq t < k - 1\) and \(d = 2k - 1 - t\) is presented in Section III. Section IV further extends to all \(2 \leq t \leq n - k\) and \(d \geq \max(2k - 1 - t, k)\). Section V proves the nonexistence of linear scalar MSCR codes for \(k < d < 2k - 2 - t\). Finally, Section VI concludes the paper.

II. THE PRODUCT MATRIX FRAMEWORK

Let \([n]\) stand for \([1, 2, \ldots, n]\). We recall the product matrix framework proposed in [3]. First, each codeword is represented by an \(n \times \alpha\) matrix \(C\) with the \(i\)-th row stored in the \(i\)-th node for all \(i \in [n]\). Moreover, the codeword is generated as a product of two matrices, i.e.,
\[
C = GM,
\]
where \(G\) is an \(n \times d\) encoding matrix, and \(M\) is a \(d \times \alpha\) message matrix. More specifically, the matrix \(M\) contains all message symbols where some entries may be linear combinations of the message symbols, and \(G\) is a predefined matrix which is independent of the message. To store a data file (i.e., a message) consisting of \(B\) symbols, we firstly arrange the \(B\) symbols into the message matrix \(M\) properly, and then calculate \(C = GM\) to obtain a codeword which will be stored across \(n\) nodes. Denote by \(\psi_i\) the \(i\)-th row of \(G\), then the content stored in node \(i\) is represented by
\[
c_i = \psi_i M,
\]
for \(i \in [n]\). Throughout this paper, we use bold letters (e.g. \(\psi, \varphi, e\), etc.) to denote vectors and capital letters (e.g. \(C, M, \Phi\), etc.) denote matrices.

Next we use the construction of scalar MSR codes to describe how this framework works for building regenerating codes. For simplicity, we just consider the case \(d = 2k - 2\).

First, the parameters of the scalar MSR code with \(d = 2k - 2\) are as follows:
\[
\alpha = d - k + 1 = k - 1, \quad \beta_1 = 1, \quad B = ka = k(k - 1).
\]
Let \(a_1, \ldots, a_n\) be \(n\) distinct nonzero elements in a finite field \(\mathbb{F}_q\) such that \(a_i^{k-1} \neq a_j^{k-1}\) for any \(i \neq j\). Then the \(n \times d\) encoding matrix is set to be
\[
G = \begin{pmatrix}
1 & a_1 & \cdots & a_1^{2k-3} \\
1 & a_2 & \cdots & a_2^{2k-3} \\
\vdots & \vdots & \ddots & \vdots \\
1 & a_n & \cdots & a_n^{2k-3}
\end{pmatrix}.
\]
The $d \times \alpha$ message matrix is defined as

$$M = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix},$$

where $S_1$ and $S_2$ are two $(k-1) \times (k-1)$ symmetric matrices each filled with $\frac{k(k-1)}{2}$ message symbols. Thus the total number of symbols contained in $M$ is $k(k-1) = B$. For $i \in [n]$, the $i$-th node stores a row vector of length $\alpha = k - 1$:

$$c_i = \psi_i M = \begin{pmatrix} 1 & a_i & \cdots & a_i^{2k-3} \end{pmatrix} M.$$

Then we illustrate the above construction gives a scalar MSR code. Actually, we need to verify the following two properties:

1) Node repair: Suppose the $i$-th node fails, then a newcomer can recover the content stored in the $i$-th node by connecting to any $d$ helper nodes and downloading $\beta = 1$ symbol from each. Let $R \subseteq [n] \setminus \{i\}$ be the set of $d$ helper nodes and define a row vector $\phi_i = (1, a_i, \ldots, a_i^{k-2})$. Then each node $j \in R$ sends the symbol

$$e_j \phi_i^\tau = \psi_j M \phi_i^\tau$$

to the newcomer, where $\tau$ denotes the transpose. Thus the newcomer obtains the symbols: $\Psi_{\text{repair}} M \phi_i^\tau$, where $\Psi_{\text{repair}}$ is the matrix $G$ restricted to the rows indexed by the elements in $R$. Since $\Psi_{\text{repair}}$ is a $d \times d$ Vandermonde matrix, by multiplying the inverse of $\Psi_{\text{repair}}$ the newcomer obtains

$$M \phi_i^\tau = \begin{pmatrix} S_1 \phi_i^\tau \\ S_2 \phi_i^\tau \end{pmatrix}.$$

Since $S_1$ and $S_2$ are both symmetric, then $\phi_i S_1$ and $\phi_i S_2$ are obtained by a transposition. Finally, the newcomer computes $\phi_i S_1 + a_i^{k-1} \phi_i S_2 = \psi_i M = c_i$ which are exactly the contents stored in the $i$-th node.

2) Data reconstruction: A data collector can recover the original data file by reading the contents stored in any $k$ nodes.

Suppose the data collector connects to $k$ nodes $i_1, i_2, \ldots, i_k \in [n]$. Let $\Psi$ be the $k \times d$ sub-matrix of $G$ consisting of the rows $\psi_{i_1}, \ldots, \psi_{i_k}$, then the data collector reads the $k \alpha$ symbols

$$\Psi M = \Psi \begin{pmatrix} S_1 \\ S_2 \end{pmatrix} = \Phi S_1 + \Delta \Phi S_2,$$

where $\Phi$ is the $k \times (k-1)$ matrix consisting of the first $k-1$ columns of $\Psi$, i.e.,

$$\Phi = \begin{pmatrix} 1 & a_{i_1} & \cdots & a_{i_1}^{k-2} \\ 1 & a_{i_2} & \cdots & a_{i_2}^{k-2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_{i_k} & \cdots & a_{i_k}^{k-2} \end{pmatrix}, \quad (5)$$

and $\Delta = \text{diag}[a_{i_1}^{k-1}, a_{i_2}^{k-1}, \ldots, a_{i_k}^{k-1}]$ is a $k \times k$ diagonal matrix. Obviously, $\Phi$ is a Vandermonde matrix and $\Psi = (\Phi \ \Delta \Phi)$. The data collector can recover the data file due to the following lemma which is also a result in [3].

**Lemma 1.** [3] Let $\Phi$ be a $k \times (k-1)$ Vandermonde matrix as defined in (5), and $\Delta$ be a $k \times k$ diagonal matrix with distinct and nonzero diagonal elements. Suppose

$$X = \Phi S + \Delta \Phi T,$$

where $S$ and $T$ are two $(k-1) \times (k-1)$ symmetric matrices. Then $S$ and $T$ can be uniquely computed from $X, \Phi$ and $\Delta$.

III. MSCR codes with $d = 2k - 1 - t$ and $2 \leq t \leq k - 1$

In this section, we construct a scalar MSCR code with $2 \leq t \leq k - 1$ and $d = 2k - 1 - t$. The restrictions on the parameters are explained in Remark 2. By the parameters of MSCR codes given in (3), we know in this case

$$\alpha = d - k + t = k - 1, \ B = k \alpha = k(k-1).$$

Since we are to give the construction using the product matrix framework, the key point is to design the encoding matrix $G$ and the message matrix $M$.

For simplicity, denote $\mu = k - t$. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be $n$ distinct nonzero elements in $\mathbb{F}_q$ such that $\alpha_i^\mu \neq \alpha_j^\mu$ for $1 \leq i \neq j \leq n$. In particular, set $q \geq (d - k + 1)n = \mu n$, then we can choose $\alpha_i = \xi_i^\mu$ where $\xi$ is a primitive element in $\mathbb{F}_q$. Our encoding matrix is defined as follows, which is an $n \times d$ Vandermonde matrix.

$$G = \begin{pmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{2k-2-t} \\ 1 & \alpha_2 & \cdots & \alpha_2^{2k-2-t} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \cdots & \alpha_n^{2k-2-t} \end{pmatrix}.$$ 

The message matrix $M$ is a $d \times \alpha$ matrix which has the following form.

$$M = \begin{pmatrix} S_{n-1} \\ \vdots \\ S_0 \end{pmatrix} + \begin{pmatrix} 0_\mu \\ \vdots \\ T_{n-1} \end{pmatrix},$$

where $S$ and $T$ are two $(k-1) \times (k-1)$ symmetric matrices each filled with $\frac{k(k-1)}{2}$ message symbols chosen from $\mathbb{F}_q$. Thus the total number of message symbols contained in $M$ is $k(k-1) = B$. The subscript $k-1$ or $\mu$ denotes the number of rows, thus here $0_\mu$ means a $\mu \times (k-1)$ all-zero matrix. Note that $k - 1 + \mu = k - 1 + k - t = d$, thus $M$ is a $d \times \alpha$ matrix.

**Remark 1.** Note that the symmetric matrices $S$ and $T$ respectively form the first and the last $\mu$ rows of $M$, while interweave in the mediate $k - 1 - \mu = t - 1$ rows of $M$. This partial interweaving structure is the key idea in our design of MSCR codes. Actually, comparing with MSR codes which have $\alpha = d - k + 1$, MSCR codes have $\alpha = d - k + t$ for $t \geq 2$. That is, fixing $\alpha = k - 1$, the MSCR codes have $d$ less than the MSR codes by $t - 1$. Since the message matrix has $d$ rows, the reduction on $d$ for MSCR codes is realized by interweaving $S$ and $T$ in $t - 1$ rows. On the other hand, the interweaved message symbols just can be unpicked through the exchanging phase between $t$ newcomers in the cooperative repair.
Next, we illustrate the data reconstruction for the construction.

**Theorem 2.** (Data reconstruction) The B message symbols can be recovered from any k nodes.

**Proof:** For any k nodes $i_1, i_2, ..., i_k \in [n]$, denote by $\Psi$ the sub-matrix of $G$ restricted to the rows corresponding to the k nodes. Then the data collector can obtain the symbols

$$\Psi M = \Psi \begin{pmatrix} S \\ 0 \end{pmatrix} + \Psi \begin{pmatrix} 0 \\ T \end{pmatrix} = \Phi S + \Delta \Phi T,$$

where $\Phi$ denotes the $k \times (k-1)$ matrix formed by the first $k-1$ columns of $\Psi$, and $\Delta = \text{diag}[a_{i_1}^{k-1}, a_{i_2}^{k-1}, ..., a_{i_k}^{k-1}]$. The diagonal elements of $\Delta$ are all nonzero and distinct by our construction. For the second equality above, one needs to note the sub-matrix formed by the last $k-1$ columns of $\Psi$ equals $\Delta \Phi$.

Then the theorem follows from Lemma 1.

**Remark 2.** The reason $d$ is restricted to $d = 2k - 1 - t$ is that we want to keep $a = d-k+t = k-1$ unchanged so that Lemma 1 still can be used to ensure the data reconstruction. On the other hand, in this section we have $2 \leq t \leq k-1$ rather than the most general condition $2 \leq t \leq n-k$, because $t \leq k-1$ implies $t-1 < k-1$ which means the two symmetric matrices $S$ and $T$ are not totally interweaved and thus the diagonal matrix $\Delta$ has $k$ distinct diagonal elements as required by Lemma 1.

Before illustrating the cooperative repair of t nodes, we first give an example for the case of $t = 2$.

**Example 1.** Set $t = 2$, then it has $d = 2k - 3$, $a = k - 1$ and $B = k(k-1)$. Moreover, $\mu = k-2$ in this case. The encoding matrix $G$ and message matrix $M$ are as follows.

$$G = \begin{pmatrix} 1 & a_1 & \cdots & a_1^{2k-4} \\ 1 & a_2 & \cdots & a_2^{2k-4} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_n & \cdots & a_n^{2k-4} \end{pmatrix}, \quad M = \begin{pmatrix} S_{i-1} \\ 0_{k-2} \end{pmatrix} + \begin{pmatrix} 0_{k-2} \\ T_{k-1} \end{pmatrix}.$$  

We explain the cooperative repair process of any two failed nodes in this example by the following lemma.

**Lemma 3.** Suppose node $i_1$ and $i_2$ fail. Then two newcomers can regenerate node $i_1$ and $i_2$ respectively through a cooperative repair in two phases:

- Phase 1: Each newcomer connects to any $d = 2k - 3$ surviving nodes as helper nodes and downloads one symbol from each helper node.

- Phase 2: Two newcomers exchange one symbol with each other.

**Proof:** For simplicity, the two newcomers are also called node $i_1$ and $i_2$ respectively. For $i \in [n]$, denote by $\psi_i$ the i-th row of $G$, then the data stored in the i-th node is $c_i = \psi_i M = (c_{i1}, ..., c_{ik-1})$. Moreover, define $\phi_i = (1, a_i, ..., a_i^{k-2})$, then

$$c_i = \psi_i M = \phi_i S + a_i^{k-2} \phi_i T.$$

Therefore, node $i_1$ needs to recover $c_{i_1} = \phi_{i_1} S + a_{i_1}^{k-2} \phi_{i_1} T$, and node $i_2$ needs to recover $c_{i_2} = \phi_{i_2} S + a_{i_2}^{k-2} \phi_{i_2} T$.

In Phase 1, suppose the set of helper nodes connected by node $i_1$ is $R_1 \subseteq [n] \setminus \{i_1, i_2\}$ with $|R_1| = 2k - 3$. Then for each $j \in R_1$, node $j$ sends the symbol

$$c_{i_1} \phi_{i_1}^T = \psi_j M \phi_{i_1}^T$$

to node $i_1$. Thus node $i_1$ obtains the symbols $\Psi_{\text{repair}} M \phi_{i_1}^T$, where $\Psi_{\text{repair}}$ is a $(2k - 3) \times (2k - 3)$ invertible matrix consisting of the rows $\psi_j$, $j \in R_1$. Therefore, node $i_1$ can derive the symbols $M \phi_{i_1}^T$ by multiplying the inverse of $\Psi_{\text{repair}}$.

Furthermore, we will show the symbols $M \phi_{i_1}^T$ can be used to derive partial data of $c_{i_1}$. For convenience, denote

$$M = \begin{pmatrix} S \\ 0 \\ T \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_{k-2} \end{pmatrix} + \begin{pmatrix} 0 \\ \psi_1 \\ \vdots \end{pmatrix} = \begin{pmatrix} u_1 \vdots \ u_{k-2} \psi_1 \vdots \ \vdots \end{pmatrix},$$

where $u_i$ and $\psi_i$ are the i-th row of $S$ and $T$ respectively, $i \in [k-1]$. Due to the symmetry of $S$ and $T$, $u_i^T$ and $\psi_i^T$ are also the i-th column of $S$ and $T$ respectively. Therefore,

$$M \phi_{i_1}^T = \begin{pmatrix} u_1 \cdot \phi_{i_1}^T \\ \vdots \\ u_{k-2} \cdot \phi_{i_1}^T \end{pmatrix} = \begin{pmatrix} \phi_{i_1} \cdot u_1^T \\ \vdots \\ \phi_{i_1} \cdot u_{k-2}^T \end{pmatrix} = \begin{pmatrix} \phi_{i_1} \cdot (u_1^T + \psi_1^T) \\ \vdots \\ \phi_{i_1} \cdot (u_{k-2}^T + \psi_{k-2}^T) \end{pmatrix} = \begin{pmatrix} \omega_{i_1} \\ \vdots \\ \omega_{k-2} \end{pmatrix},$$

where the second equality comes from the transposition, and $\omega_i$ denotes the i-th coordinate of $M \phi_{i_1}^T$ for $i \in [2k-3]$. Recall that node $i_1$ is to recover $c_{i_1} = (c_{i_1,1}, ..., c_{i_1,k-1}) = \phi_{i_1} S + a_{i_1}^{k-2} \phi_{i_1} T$. That is, for $j \in [k-1]$, $c_{i_1,j} = \phi_{i_1} u_j^T + a_{i_1}^{k-2} \phi_{i_1} \psi_j$.

Actually, after obtaining $M \phi_{i_1}^T = (\omega_{1}, ..., \omega_{2k-3})^T$ node $i_1$ computes the following

$$\begin{cases} \omega_{1} + a_{i_1}^{k-2} \omega_{k-1} + a_{i_1}^{2(k-2)} \omega_{2k-3}, \\ \omega_{2} + a_{i_1}^{k-2} \omega_{k}, \\ \omega_{3} + a_{i_1}^{k-2} \omega_{k+1}, \\ \vdots \\ \omega_{2k-2} + a_{i_1}^{k-2} \omega_{2k-4} \end{cases}.$$  

It can be verified that

$$\begin{align*} &\omega_{1} + a_{i_1}^{k-2} \omega_{k-1} + a_{i_1}^{2(k-2)} \omega_{2k-3} \\ &= \phi_{i_1} \cdot u_1^T + a_{i_1}^{k-2} \phi_{i_1} \cdot (u_1^T + \psi_1^T) + a_{i_1}^{2(k-2)} \phi_{i_1} \cdot \psi_{k-1}^T \\ &= \phi_{i_1} \cdot (u_1^T + \psi_1^T) + a_{i_1}^{k-2} \phi_{i_1} \cdot (u_1^T + \psi_1^T) \\ &= c_{i_1,1} + a_{i_1}^{k-2} c_{i_1,k-1}. \end{align*}$$
while for \( j \in \{2, \ldots, k-2\}, \)
\[
\omega_j + \alpha_{i_1}^{-k-2} \omega_{j+k-2} = \varphi_{i_1} \cdot u^*_j + \alpha_{i_1}^{-k-2} \varphi_{i_1} \cdot v^*_j = c_{i_1, j}.
\]
That is, after Phase 1 node \( i_1 \) gets \( d \) symbols \( M \varphi^*_1 \), from which it further recovers \( c_{i_1,2}, \ldots, c_{i_1,k-2} \) and \( c_{i_1,1} + \alpha_{i_1}^{-k-2} c_{i_1,k-1} \). Similarly, by connecting to \( d \) helper nodes and downloading \( c_j \varphi^*_j \) from each helper node \( j \), node \( i_2 \) can get \( d \) symbols \( M \varphi^*_2 \), from which it can further recover \( c_{i_2,2}, \ldots, c_{i_2,k-2} \) and \( c_{i_2,1} + \alpha_{i_2}^{-k-2} c_{i_2,k-1} \).

In Phase 2, node \( i_1 \) sends the symbol \( \psi_{i_1} M \varphi^*_1 \) to \( i_2 \), and node \( i_2 \) sends the symbol \( \psi_{i_2} M \varphi^*_2 \) to \( i_1 \). So, node \( i_1 \) gets
\[
\psi_{i_1} M \varphi^*_2 = c_{i_1} \varphi^*_2 = (c_{i_1,1}, \ldots, c_{i_1,k-1})(1)_{a_{i_2}} \cdot (\alpha_{i_2})^{a_{i_2}^{-k-2}} \cdot c_{i_1,k-1}
\]
from which node \( i_1 \) can obtain the value of \( c_{i_1,1} + \alpha_{i_1}^{-k-2} c_{i_1,k-1} \) because node \( i_1 \) has already obtained \( c_{i_1,j} \) for \( j \in \{2, \ldots, k-2\} \) after Phase 1. Thus \( c_{i_1,1} \) and \( c_{i_1,k-1} \) can be solved from
\[
\begin{cases}
    c_{i_1,1} + \alpha_{i_1}^{-k-2} c_{i_1,k-1} \\
    c_{i_1,1} + \alpha_{i_2}^{-k-2} c_{i_1,k-1}
\end{cases}
\]
where the second equality comes from the transposition. Therefore, for each \( l \in [2k-1-t] \),
\[
\omega_l = \begin{cases}
    \varphi_{i_1} u^*_l, & 1 \leq l \leq \mu \\
    \varphi_{i_2} v^*_{l-\mu}, & \mu + 1 \leq l \leq k-1 \\
    \varphi_{i_2} v^*_{l-k-1}, & k \leq l \leq 2k-1-t
\end{cases}
\]
Then we do some calculations on the symbols \( \omega_1, \ldots, \omega_{2k-1-t} \) to help recover \( c_i \). In particular, denote \( d = 2k-1-t = z \mu + r \) for some integers \( z \) and \( r \) where \( 0 \leq r \leq \mu - 1 \). Then \( \alpha = k-1 = d-\mu = (z-1)\mu + r \). Next node \( i \) computes the following \( \mu \) symbols
\[
\begin{aligned}
    \omega_1 + \alpha_i^\mu \omega_{\mu+1} + \alpha_i^{2\mu} \omega_{2\mu+1} + \cdots + \alpha_i^{(z+1)\mu} \omega_{(z+1)\mu+1} \\
    + \alpha_i^{z\mu} \omega_{z\mu+1} \\
    \vdots \\
    \omega_r + \alpha_i^\mu \omega_{r+\mu} + \alpha_i^{2\mu} \omega_{2r+\mu} + \cdots + \alpha_i^{(z+1)\mu} \omega_{(z+1)\mu+r} \\
    + \alpha_i^{r\mu} \omega_{r\mu+r} \\
    \vdots \\
    \omega_{2r+1} + \alpha_i^{2\mu} \omega_{2r+1+\mu} + \alpha_i^{3\mu} \omega_{2r+1+\mu+r} + \cdots + \alpha_i^{(z+1)\mu} \omega_{(z+1)\mu+r+1} \\
    + \alpha_i^{(z+1)\mu} \omega_{(z+1)\mu+r+1} \\
    \vdots \\
    \omega_{\mu} + \alpha_i^\mu \omega_{\mu+\mu} + \alpha_i^{2\mu} \omega_{2\mu+\mu} + \cdots + \alpha_i^{(z+1)\mu} \omega_{(z+1)\mu+\mu}
\end{aligned}
\]
Recall that from (8) it has \( c_{i,j} = \varphi_{i} u^*_j + \alpha_{i}^\mu \varphi_{i} v^*_j \) for \( j \in [k-1] \). Then, it can be verified that for \( 1 \leq l \leq r \),
\[
\begin{aligned}
    \omega_l + \alpha_i^\mu \omega_{l+\mu} + \cdots + \alpha_i^{(z+1)\mu} \omega_{(z+1)\mu+l} + \alpha_i^{z\mu} \omega_{z\mu+l} \nonumber \\
    = \varphi_{i} (u^*_l + \alpha_{i}^\mu (u^*_{l+\mu} + v^*_j)) + \cdots + \alpha_i^{z\mu} \omega_{z\mu+l} \\
    \vdots \\
    = \varphi_{i} (u^*_l + \alpha_{i}^\mu v^*_l) + \alpha_i^{\mu} (u^*_l + \alpha_{i}^{\mu} (u^*_{l+\mu} + v^*_j)) + \cdots + \alpha_i^{(z-1)\mu} (u^*_{(z-1)\mu+l} + \alpha_{i}^{(z-1)\mu} v^*_j)) + \cdots + \alpha_i^{(z+1)\mu} (u^*_{(z+1)\mu+l} + \alpha_{i}^{(z+1)\mu} v^*_j)) \\
    = c_{i,l} + \alpha_{i}^\mu c_{i,\mu+l} + \cdots + \alpha_i^{(z+1)\mu} c_{i,(z+1)\mu+l},
\end{aligned}
\]
while for $r + 1 \leq l \leq \mu$, 
$$
\omega_l + a_i^\mu \omega_{l-1} + \cdots + a_i^{(z-2)\mu} \omega_{l-(z-2)} + a_i^{(z-1)\mu} \omega_{l-(z-1)} + \cdots + a_i^{\mu} \omega_{l-1} + a_i^{(z-2)\mu} v_{l-(z-2)}^{(2)} + a_i^{(z-1)\mu} v_{l-(z-1)}^{(2)}
$$

(1)

$$\varphi\left(\begin{array}{c}
  a_i^{\mu} v_{l}^{(2)} \\
  \end{array}
\right)
$$

For $\mu$, this linear system has coefficient matrix 
$$
H = 
\begin{pmatrix}
  1 & a_i & a_i^2 & \cdots & a_i^{\mu-1} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  1 & a_i & a_i^2 & \cdots & a_i^{\mu-1} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  1 & a_i & a_i^2 & \cdots & a_i^{\mu-1} \\
\end{pmatrix}
$$

(2)

Therefore, after the two phases, node $i$ obtains $\mu + t - 1 = k - t + t - 1 = k - 1$ linear equations on $c_{i,1}, c_{i,2}, \ldots, c_{i,k-1}$. Moreover, this linear system has coefficient matrix 
$$
H =
\begin{pmatrix}
  H_{i,1} \\
  H_{i,2} \\
\end{pmatrix}
$$

(3)

$$
H_{i,1} = \begin{pmatrix}
  I_{\mu} & a_i I_{\mu} & a_i^{(2)\mu} I_{\mu} & \cdots & a_i^{(z-2)\mu} I_{\mu} & a_i^{(z-1)\mu} I_{\mu} \\
\end{pmatrix}^{(r)}
$$

(4)

where $I_{\mu}$ denotes the $\mu \times \mu$ identity matrix, and $I_{\mu}^{(r)}$ denotes the $\mu \times r$ matrix consisting of the first $r$ columns of $I_{\mu}$.

Now we turn to Phase 2 and still fix some $i \in [r]$. Since each node $j \in [r] \setminus [i]$ has known $M\phi_j^i$ after Phase 1, then node $j$ sends the symbol $\psi_j M\phi_j^i = c_i \phi_j^i$ to node $i$. Thus in Phase 2 node $i$ receives $t - 1$ symbols $[c_i \phi_j^i]_j \in [r]$, $j \neq i$ which correspond to $t - 1$ linear equations on $c_{i,1}, c_{i,2}, \ldots, c_{i,k-1}$. Write these $t - 1$ linear equations in the matrix form $H_{i,2} \phi_i$, then the coefficient matrix $H_{i,2}$ has the form 
$$
H_{i,2} = 
\begin{pmatrix}
  1 & a_i & a_i^2 & \cdots & a_i^{k-2} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  1 & a_i & a_i^2 & \cdots & a_i^{k-2} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  1 & a_i & a_i^2 & \cdots & a_i^{k-2} \\
\end{pmatrix}
$$

(5)

IV. MSCR CODES WITH $d \geq \max\{2k - 1 - t, k\}$ AND $2 \leq t \leq n - k$

In this section, we first show that any MSCR code can be transformed into a systematic MSCR code with the same parameters. In particular, the codes constructed in Section III could have a systematic form. Then by applying a shortening technique to these codes, we build scalar MSCR codes for all $2 \leq t \leq n - k$ and $d \geq \max\{2k - 1 - t, k\}$.

A. Systematic MSCR codes

Since $B = ka$ for MSCR codes, the original data file can be denoted as $(m_1, ..., m_k)$ where each $m_i, i \in [k]$, consists of $a$ symbols. An MSCR code is called systematic if there exist $k$ nodes (called systematic nodes) which store $m_i, i \in [k]$, respectively. In fact, through a reverse application of the data reconstruction, we can turn any MSCR code into a systematic one.

Theorem 5. Suppose there exists an MSCR code $C$ with parameters $(n, k, d, t, a, \beta)$, then for any $I \subseteq [n]$ with $|I| = k$, there exists an $(n, k, d, t, a, \beta)$ systematic MSCR code $C'$ taking the nodes in $I$ as systematic nodes.

Proof: In general, we define the MSCR code $C$ by using its encoding map $E : F^B \rightarrow (F^a)^n$, $E(m) = (c_1, ..., c_n)$, that is, for a data file $m$ of size $B$, node $i$ stores $c_i$, for $i \in [n]$.

By the data reconstruction property, the content stored in any $k$ nodes uniquely determines the data file. In particular,
for $l \leq [n]$ with $|l| = k$, there exists a reconstruction function $R_I : (F^n)^k \to F^B$ such that

$$R_I(E'|_I(m)) = m, \forall m \in F^B, \quad (12)$$

where $E'|_I(m)$ denotes $E(m)$ restricted to the nodes in $I$. Moreover, $(12)$ implies that both $R_I$ and $E'|_I$ are one-to-one maps and $E'|_I = R_I^{-1}$.

For any data file $m \in F^B = F^{ka}$, denote $m = (m_1, \ldots, m_k)$ where $m_i \in F^a$ for $i \in [k]$. Then we define a new MSCR code by the encoding function $E' : F^B \to (F^a)^m$, such that

$$E'(m) = E(R_I(m_1, \ldots, m_k)).$$

Then we will say that $E'$ actually defines the systematic MSCR code $C'$ as required by the theorem.

First, because $R_I$ and $E'|_I$ are one-to-one maps, it follows that $E'|_I$ is a one-to-one map for any $|I| \leq [n]$ with $|I| = k$. Define the inverse map of $E'|_I$ as the reconstruction function for $I'$, then $C'$ satisfies the data reconstruction property. Moreover,

$$E'|_I(m) = E|_I(R_I(m_1, \ldots, m_k)) = R_I^{-1}(E_R(m_1, \ldots, m_k)) = (m_1, \ldots, m_k),$$

so the $k$ nodes in $I$ are systematic nodes. For the repair property, since $C'$ and $C$ have the same codeword space except that they have different encoding maps, the repair property of $C$ are maintained in $C'$. It is easy to verify that $C'$ and $C$ have the same parameters. The theorem is proved.

In particular, for the $(n, k, d, t)$ scalar MSCR code constructed under the product matrix framework, if we want the nodes $\{1, \ldots, k\}$ to be systematic nodes, then for any data file $m = (m_1, \ldots, m_k) \in (F^a)^k$ we first solve the message matrix $M(m)$ from $E' |_I(m) = W(m)$ through the data reconstruction process, where $\Psi$ denotes the encoding matrix $G$ restricted to the first $k$ rows, and $W(m)$ is the $k \times a$ matrix whose $k$ rows are exactly $m_1, \ldots, m_k$. Thus we obtain a systematic $(n, k, d, t)$ scalar MSCR code which encodes the data file $m$ as $C = GM(m)$.

B. Scalar MSCR codes with $d \geq \max[2k - 1 - t, k]$ and $2 \leq t \leq n - k$

In Section III, we have constructed scalar MSCR codes for any $2 \leq t \leq k - 1$ and $d = 2k - 1 - t$. Next we show by proper shortening from these codes, one can derive scalar MSCR codes for all $d \geq \max[2k - 1 - t, k]$ and $2 \leq t \leq n - k$. First, Theorem 6 states the relations between the parameters of the original MSCR code and the shortened code. The shortening technique is specifically described in the proof of Theorem 6. Then in Corollary 7, applying the shortening technique to a previously constructed MSCR code, it produces the code that is desired in this Section.

**Theorem 6.** If there exists an $(n', k', d', t)$ scalar MSCR code $C'$ for some $\delta \geq 0$, then there must exist an $(n, k, d, t)$ scalar MSCR code $C$.

**Proof:** By Theorem 5, we can assume that $C'$ is an $(n', k', d', t)$ systematic scalar MSCR code with systematic nodes $1, \ldots, k'$. From $(3)$ we know that

$$a' = d' - k' + t = d - k + t$$

and

$$B' = k'a' = (k + \delta)(d - k + t),$$

while a scalar MSCR code with parameters $(n, k, d, t)$ has

$$a = d - k + t, \quad B = k\alpha.$$ 

Thus it has

$$a' = a, \quad B' = B + \delta a.$$ 

Now consider all the codewords in $C'$ that have zeros in the first $\delta$ nodes and then puncture these codewords in the first $\delta$ nodes, it gives the desired $(n, k, d, t)$ MSCR code $C$.

More specifically, in the data reconstruction any $k$ nodes in $C$ plus $\delta$ imaginary systematic nodes that store all zeros correspond to $k'$ nodes in $C'$ which uniquely determines a data file of length $B'$ with the first $\delta a$ symbols being zeros, therefore any $k$ nodes in $C$ uniquely determines a data file of size $B' = \delta a$. The cooperative repair of any $t$ nodes in $C$ with each connecting to $d$ helper nodes can be done as the cooperative repair of the $t$ nodes in $C'$ with each connecting to the $d$ helper nodes and $\delta$ imaginary nodes that store all zeros. Therefore, one can see that $C$ is an $(n, k, d, t)$ scalar MSCR code.

**Corollary 7.** For any $2 \leq t \leq n - k$ and $d \geq \max[2k - 1 - t, k]$, there exists an $(n, k, d, t)$ scalar MSCR code.

**Proof:** Define $\delta = d - (2k - 1 - t) \geq 0$, and let $n' = n + \delta, k' = k + \delta, d' = d + \delta$. It is easy to verify that $d' = 2k' - 1 - t$. Since $t \leq d - k + t = k' - 1$, then we can obtain an $(n', k', d', t)$ scalar MSCR code from the construction in Section III. Thus the desired $(n, k, d, t)$ scalar MSCR code can be constructed as in Theorem 6.

As a result, when $2k - 1 - t \leq k$, i.e., $k \leq t + 1$, our construction presents scalar MSCR codes for all $d \geq k$ which covers all possible parameters for $(n, k, d, t)$ cooperative regenerating codes. When $2k - 1 - t > k$, i.e., $k > t + 1$, our construction restricts to the case $d \geq 2k - 1 - t$. As a complementary result, for the latter case, we will prove the nonexistence of a family of linear scalar MSCR codes for $k < d < 2k - 2 - t$ in the next section. Recall that, the existence of scalar MSCR codes for $d = k$ has been ensured in [14].

V. NONEXISTENCE OF LINEAR SCALAR MSCR CODES FOR $k < d < 2k - 2 - t$

The nonexistence result relies on an assumption that the linear MSCR codes have invariant repair spaces. In the following, we first describe the linear model for MSCR codes and explain the property of invariant repair space. Then we derive the interference alignment property for such MSCR codes and prove the nonexistence result under the condition $k < d < 2k - 2 - t$. 


A. Linear MSCR codes with invariant repair space

Suppose there exists an \((n, k, d, t, \alpha, \beta)\) linear MSCR code \(C\) over \(\mathbb{F}_q\). By Theorem 5, we can always assume that \(C\) is systematic and has the following generator matrix

\[
G = \begin{pmatrix}
I_{\alpha} & I_{\alpha} & \cdots & I_{\alpha} \\
A_{1,1} & A_{1,2} & \cdots & A_{1,k} \\
\vdots & \vdots & \ddots & \vdots \\
A_{r,1} & A_{r,2} & \cdots & A_{r,k}
\end{pmatrix},
\]

where \(I_{\alpha}\) denotes the \(\alpha \times \alpha\) identity matrix, \(A_{i,j}\) is a \(\alpha \times \alpha\) matrix over \(\mathbb{F}_q\) for \(i \in [r]\), \(j \in [k]\), and \(r = n - k\). For simplicity, every \(\alpha\) consecutive rows of \(G\) are regarded as a thick row, thus \(G\) has \(n\) thick rows which exactly correspond to the \(n\) storage nodes. For any data vector \(m \in \mathbb{F}_q^{ka}\), node \(i\) stores an \(\alpha\)-dimensional vector \(e_i = G_i m^T\), where \(G_i\) denotes the \(i\)-th thick row of \(G\) and \(i \in [n]\). Since \(m\) is independently and uniformly chosen from \(\mathbb{F}_q^{ka}\), we can also view node \(i\) as storing the linear space spanned by the rows of \(G_i\), denoted by \(\langle G_i \rangle\), which is a subspace of \(\mathbb{F}_q^{ka}\). Then the data reconstruction requirement can be restated as follows.

Data reconstruction. The subspaces stored in any \(k\) nodes can generate the entire space, namely, for any \(i_1, \ldots, i_k \in [n]\), \(\sum_{j=1}^k \langle G_{i_j} \rangle = \mathbb{F}_q^{ka}\).

Obviously, the data reconstruction requirement implies that each \(A_{i,j}\) is invertible for all \(i \in [r]\) and \(j \in [k]\).

Then we describe the cooperative repair process. Suppose \(\mathcal{F} \subseteq [n]\) is the set of failed nodes and \(|\mathcal{F}| = t\). For each \(i \in \mathcal{F}\), let \(\mathcal{H}_i \subseteq [n] \setminus \mathcal{F}\) denote the set of helper nodes for repairing node \(i\) and \(|\mathcal{H}_i| = d\). In the first phase of the repair process, each node \(i \in \mathcal{H}\) transmits \(S_{j-i,\mathcal{F},\mathcal{H}_i} e_i^T = S_{j-i,\mathcal{F},\mathcal{H}_i} G_i m^T\) to repair node \(i\), where \(S_{j-i,\mathcal{F},\mathcal{H}_i}\) is a \(\beta_1 \times \alpha\) matrix corresponding to the linear transformation performed on node \(j\) and \(\mathcal{H} = \langle \mathcal{H}_i \rangle_{i \in \mathcal{F}}\). From the view of linear spaces, we call \(\langle S_{j-i,\mathcal{F},\mathcal{H}_i} G_j \rangle\) as the repair space of node \(j\) for repairing node \(i\) with respect to the failed node set \(\mathcal{F}\) and the helper node set \(\mathcal{H}\). In the second phase of the repair process, the nodes in \(\mathcal{F}\) exchange data with each other. Specifically, for any node \(i \in \mathcal{F}\), each node \(i' \in \mathcal{F} \setminus \{i\}\) transmits to node \(i\) a \(\beta_2\)-dimensional repair space \(\gamma_{i'-i,\mathcal{F},\mathcal{H}_i}\), where \(\gamma_{i'-i,\mathcal{F},\mathcal{H}_i}\) is a \(\beta_2 \times k\alpha\) matrix generated from \(\sum_{i \in \mathcal{H}_i} \langle S_{j-i,\mathcal{F},\mathcal{H}_i} G_j \rangle\). Then the node repair requirement can be restated as follows.

Cooperative repair. For any \(i \in \mathcal{F}\), the space stored by node \(i\) can be recovered from the repair spaces collected by node \(i\) in the two phases of the cooperative repair process, i.e., \(\langle G_i \rangle \subseteq \sum_{i \in \mathcal{H}_i} \langle S_{j-i,\mathcal{F},\mathcal{H}_i} G_j \rangle + \sum_{i' \in \mathcal{F} \setminus \{i\}} \langle \gamma_{i'-i,\mathcal{F},\mathcal{H}_i} \rangle\).

Definition 8. A linear MSCR code with the generator matrix defined in (13) is said to have invariant repair spaces if for any \(i, j \in [n]\), the repair space \(\langle S_{j-i,\mathcal{F},\mathcal{H}_i} G_j \rangle\) is independent of \(\mathcal{F}\) and \(\mathcal{H}\), or equivalently, the repair matrix \(S_{j-i,\mathcal{F},\mathcal{H}_i}\) is independent of \(\mathcal{F}\) and \(\mathcal{H}\).

As a result, for a linear MSCR code with invariant repair spaces, we denote the repair matrix by \(S_{j-i,\mathcal{F},\mathcal{H}_i}\) instead of \(S_{j-i,\mathcal{F},\mathcal{H}_i}\), which means so long as node \(j\) is connected to repair node \(i\) in a cooperative repair of \(t\) failed nodes containing \(i\), node \(j\) always performs the same linear transformation \(S_{j-i,\mathcal{F},\mathcal{H}_i}\) on its stored data in spite of the identity of other failed nodes and other helper nodes. This property brings great convenience to the repair process in practice. Actually, most of the existing scalar MSCR codes have invariant repair spaces, such as the codes constructed in [16]–[18]. Moreover, the construction in this paper also satisfies this property.

B. Interference alignment and nonexistence result

Next we consider only linear scalar MSCR codes that have invariant repair spaces. Since for scalar MSCR codes, it holds \(\beta_1 = \beta_2 = 1\) and \(\alpha = d - k + t\), the repair spaces \(\langle S_{j-i,\mathcal{F},\mathcal{H}_i} G_j \rangle\) and \(\langle \gamma_{i'-i,\mathcal{F},\mathcal{H}_i} \rangle\) are both \(1\)-dimensional subspaces, thus are denoted as \(\langle s_{j-i,\mathcal{F},\mathcal{H}_i} G_j \rangle\) and \(\langle \gamma_{i'-i,\mathcal{F},\mathcal{H}_i} \rangle\) respectively. Moreover, we always assume the scalar MSCR code is systematic and has a generator matrix as defined in (13). Note that the property of having invariant repair spaces is maintained after transforming a linear MSCR code to a systematic one as illustrated in Theorem 5.

Lemma 9. (Interference alignment) For a linear scalar MSCR code as described above, we have,

(a) For any \(i, j \in [k], i \neq j\),

\[
s_k+i-1 \sim_{A_1,i} \sim_{A_1,i} \cdots \sim_{A_1,i} \sim_{A_1,i} = \alpha,
\]

where \(s_{k+i-1} \sim_{A_1,i} \cdots \sim_{A_1,i} = \alpha = c x^T\) for some nonzero \(c \in \mathbb{F}_q\).

(b) Suppose \(k \geq t\), then for any \(i \in [k]\) and any \(j_1, \ldots, j_{t-1} \in [k] \setminus \{i\}\),

\[
\begin{pmatrix}
\gamma_{j_1-i,\mathcal{F},\mathcal{H}_i} \\
\gamma_{j_1-i,\mathcal{F},\mathcal{H}_i} \\
\vdots \\
\gamma_{j_{t-1}-i,\mathcal{F},\mathcal{H}_i}
\end{pmatrix} = \alpha,
\]

where \(\gamma_{j-i,\mathcal{F},\mathcal{H}_i} \in \mathbb{F}_q^{t}\) means the \(i\)-th component of \(\gamma_{j-i,\mathcal{F},\mathcal{H}_i}\), that is, \(\gamma_{j-i,\mathcal{F},\mathcal{H}_i} = \langle \gamma_{j-i,\mathcal{F},\mathcal{H}_i} \rangle_1, \ldots, \langle \gamma_{j-i,\mathcal{F},\mathcal{H}_i} \rangle_{t-1}\) for \(1 \leq i \leq t-1\), \(\mathcal{F} = \{i, j_1, \ldots, j_{t-1}\} \subseteq [k]\) and \(\mathcal{H}_i = [k+a] \setminus \{i\}\) for all \(j' \in \mathcal{F}\).

Proof: Note that \(k + \alpha = k + (d - k + t) = d + t \leq n\) and \(\alpha = d - k + t \leq n\). The lemma is proved by considering the node repair requirement in different repair patterns.

(a) Let \(\mathcal{F} = \{1, k + \alpha - t + 2, \ldots, k + \alpha\}\), and \(\mathcal{H}_i = \{2, \ldots, k\} \cup \{k+1, \ldots, k + \alpha - t + 1\}\) for all \(i \in \mathcal{F}\). That is, one systematic node and \(t - 1\) parity nodes fail, and the remaining \(k - 1\) systematic nodes and other \(d - k + 1\) parity nodes are helper nodes. Then after the repair process, node 1 collects the space \(\Omega_1\) as displayed in (14).

The node repair requirement implies that there exists an \(\alpha \times (d + t - 1)\) matrix \(B = (b_1^T \cdots b_k^T)\) such that

\[
B \Omega_1 = G_1 = \begin{pmatrix}
I_{\alpha} & 0 & \cdots & 0
\end{pmatrix},
\]
It follows from (16) that rank 
\[
\begin{pmatrix}
  s_{j+1} & \cdots & s_k \\
  s_{j+1} & \cdots & s_k \\
  \vdots & \vdots & \vdots \\
  s_{j+1} & \cdots & s_k \\
\end{pmatrix}
\] 
= I_a,
\]
and for \( j \in \{2, \ldots, k \} \),
\[
\begin{pmatrix}
  b_j^r & \cdots & b_{k+1}^r \\
  b_j^r & \cdots & b_{k+1}^r \\
  \vdots & \vdots & \vdots \\
  b_j^r & \cdots & b_{k+1}^r \\
\end{pmatrix}
\] 
= 0. (17)

It follows from (16) that rank \( (b_j^r \cdots b_{k+1}^r) = a \). As a result, the matrix
\[
(b_j^r \cdots b_{k+1}^r)
\]
is an \( a \times (a + 1) \) matrix of rank \( a \). Then from the equality (17) we have for \( 2 \leq j \leq k \),
\[
\begin{pmatrix}
  s_{j+1} & \cdots & s_k \\
  s_{j+1} & \cdots & s_k \\
  \vdots & \vdots & \vdots \\
  s_{j+1} & \cdots & s_k \\
\end{pmatrix}
\] 
\[
\begin{pmatrix}
  s_{j+1} & \cdots & s_k \\
  s_{j+1} & \cdots & s_k \\
  \vdots & \vdots & \vdots \\
  s_{j+1} & \cdots & s_k \\
\end{pmatrix}
\]
\[
= I_a,
\]
and for \( j \in \{2, \ldots, k \} \),
\[
\begin{pmatrix}
  b_j^r & \cdots & b_{k+1}^r \\
  b_j^r & \cdots & b_{k+1}^r \\
  \vdots & \vdots & \vdots \\
  b_j^r & \cdots & b_{k+1}^r \\
\end{pmatrix}
\] 
\[= 0. (17)\]

Consider a new repair pattern by exchanging the positions of node \( k+1 \) and \( k+\alpha-t+1 \), i.e., let \( k+1 \) be a failed node and \( k+\alpha-t+1 \) be a helper node, then we can derive
\[
s_j \sim s_{k+\alpha-t+1}, \quad j \in \{2, \ldots, k \}.
\]
Continue this way, then we finally obtain
\[
s_j \sim s_{k+\alpha-t+1}, \quad 2 \leq j \leq k.
\]
Substituting node 1 by an arbitrary \( i \in [k] \), then (a) is obtained.
(b) Without loss of generality, suppose \( i = 1 \) and \( \{j_1, \ldots, j_{t-1}\} = \{2, \ldots, t\} \). With respect to the \( F \) and \( H \) defined in (b), node 1 receives the repair space \( \Omega_2 \) as in (15).
Since \( (G_1) \subseteq (\Omega_2) \), it is easy to see that (b) holds.

\[\text{Lemma 10. Suppose } d \leq 2k - 1 - t, \text{ then for any } p \in \{k + 1, \ldots, k + \alpha\}, \text{ any } a \text{ out of the } k \text{ vectors } \{s_p \rightarrow 1, \ldots, s_p \rightarrow k\} \text{ are linearly independent.}\]
Proof: Assume on the contrary that for some \( p \in \{k + 1, \ldots, k + \alpha\} \), there exist \( \alpha \) linearly dependent vectors in \( \{s_{p-1}, \ldots, s_{p-k}\} \). Without loss of generality, assume \( p = k + 1 \) and
\[
s_{k+1} \in \langle s_{k+1-2}, \ldots, s_{k+1-a} \rangle,
\]
where the notation \( \langle s_{k+1-2}, \ldots, s_{k+1-a} \rangle \) denotes the space spanned by \( \{s_{k+1-2}, \ldots, s_{k+1-a}\} \). In the following, we will show the linear dependence in (18) can be extended to all \( k + j \) for \( 1 \leq j \leq \alpha \), which will then lead to a contradiction to Lemma 9 (b).

Since \( d \leq 2k - 1 - t \), then \( k \geq d - k + t + 1 \geq \alpha + 1 \), which means that there exists the \( (\alpha + 1) \)-th component. Multiplying the invertible matrix \( A_{1,1} \), on both sides of (18), we have\[
s_{k+1} \in \langle s_{k+2-1}, \ldots, s_{k+1-a} \rangle \forall j \leq \alpha.
\]

By Lemma 9 (a), we can further obtain that for \( j \in \{1, \ldots, \alpha\} \),
\[
s_{k+j} \in \langle s_{k+j+2-1}, \ldots, s_{k+j-a} \rangle.
\]

Note that all the \( A_{j,1} \)'s are invertible from the data reconstruction requirement. Multiplying the inverse \( A_{j,1}^{-1} \), we have\[
s_{k+j} \in \langle s_{k+j+2-1}, \ldots, s_{k+j-a} \rangle, \quad 1 \leq j \leq \alpha.
\]

From \( d \leq 2k - 1 - t \) it also follows \( k \geq d - k + t + 1 > t \), thus consider the repair pattern \( F = [t] \subset [k] \) and \( H = [k + \alpha] \setminus F \) for all \( i \in F \). From Lemma 9 (b), we know the space
\[
\langle s_{k+j-1} \rangle \leq \alpha \cup \{y_{2-1,F,H}^{(1)} \cup \cdots \cup y_{1-1,F,H}^{(1)} \}
\]
has dimension \( \alpha \). However, by the definition of \( y_{2-1,F,H}^{(1)} \) it has
\[
y_{2-1,F,H}^{(1)} \in \langle s_{k+1-2} A_{1,1}, \ldots, s_{k+a-2} A_{1,1} \rangle = \langle s_{k+1-2} A_{1,1} \rangle
\]
where the equality follows from Lemma 9 (a). In a similar way, we get \( y_{1-1,F,H}^{(1)} \in \langle s_{k+1-1} A_{1,1}, \ldots, s_{k+a-1} A_{1,1} \rangle = \langle s_{k+1-1} A_{1,1} \rangle \) for \( 2 \leq i \leq t \). Therefore,
\[
\langle s_{k+j-1} \rangle \leq \alpha \cup \{y_{2-1,F,H}^{(1)} \cup \cdots \cup y_{1-1,F,H}^{(1)} \}
\]
\[
\subseteq \langle s_{k+2-1} A_{1,1}, \ldots, s_{k+a-1} A_{1,1} \rangle \quad 1 \leq j \leq \alpha \]
\[
\subseteq \langle s_{k+2-1} A_{1,1}, \ldots, s_{k+1-a} A_{1,1} \rangle
\]
where (20) come from (19), and (21) comes from Lemma 9 (a) and the fact that \( \alpha = d - k + t \geq t \). So, (20) implies that \( \langle s_{k+j-1} \rangle \) has dimension at most \( \alpha - 1 \) which contradicts to Lemma 9 (b).

Theorem 11. For \( k < d < 2k - 2 - t \), there exist no linear scalar \( (n, k, d, t) \) MSCR codes that have invariant repair spaces.

Proof: Assume on the contrary there exists such an MSCR code for some \( n, k, d, t \) with \( k < d < 2k - 2 - t \). As stated before, by Theorem 5 we can always assume this MSCR code is systematic and has a generator matrix as defined in (13). Since \( d < 2k - 2 - t \), it follows \( k > d - k + t + 2 = \alpha + 2 \), i.e. \( k > \alpha + 2 \).

We first consider the repair of node \( \alpha + 1 \) and node \( \alpha + 2 \) by the helper nodes \( k + j, 1 \leq j \leq \alpha \). In particular, we restrict to the \( (\alpha + 2) \)-th and the \( (\alpha + 3) \)-th components. That is, by Lemma 9 (a), we have
\[
\begin{align*}
s_{k+1} &\sim s_{k+2} \sim s_{k+3} \sim \cdots \sim s_{k+\alpha+2}, \\
s_{k+a} &\sim s_{k+a+1} \sim s_{k+a+2} \sim \cdots \sim s_{k+\alpha+a+2}, \\
s_{k+1} &\sim s_{k+2} \sim s_{k+3} \sim s_{k+a+2} \sim \cdots \sim s_{k+\alpha+a+2}, \\
s_{k+1} &\sim s_{k+2} \sim s_{k+a+2} \sim s_{k+\alpha+a+2} \sim \cdots \sim s_{k+\alpha+a+2}.
\end{align*}
\]

Then our proof goes along the following line. First, represent both \( s_{k+j-1} \) and \( s_{k+j-2} \) as linear combinations of \( \{s_{k+1-1}, \ldots, s_{k+j-1}\} \) (by Lemma 10). Then by Lemma 9 (a) and (22), (23) we can derive a relation between the \( (\alpha + 2) \)-th and the \( (\alpha + 3) \)-th components. With this relation we can finally substitute the \( A_{j,\alpha+3} \)'s in (24) with \( A_{j,\alpha+2} \)'s and then obtain a contradiction to Lemma 9 (b). The details are as follows.

From Lemma 10, we know that for \( 1 \leq j \leq \alpha \), the \( \alpha \) vectors \( s_{k+j-1}, \ldots, s_{k+j-\alpha} \) each of length \( \alpha \) are linearly independent, thus \( s_{k+j-\alpha+1} \) can be represented as a linear combination of \( \{s_{k+j-1}, \ldots, s_{k+j-\alpha}\} \). Specifically, suppose
\[
\begin{align*}
s_{k+j-\alpha+1} &= \lambda_{k+j,\alpha+1} \left(s_{k+j-1} \begin{array}{c}
\vdots \\
s_{k+j-\alpha}
\end{array}\right) \\
&= \left(\lambda_{k+j,\alpha+1}^{(1)} \cdots \lambda_{k+j,\alpha+1}^{(\alpha)}\right) \left(s_{k+j-1} \begin{array}{c}
\vdots \\
s_{k+j-\alpha}
\end{array}\right),
\end{align*}
\]
where \( \lambda_{k+j,\alpha+1} = (\lambda_{k+j,\alpha+1}^{(1)} \cdots \lambda_{k+j,\alpha+1}^{(\alpha)}) \in [p^q]^\alpha \). Moreover, we claim that \( \lambda_{k+j,\alpha+1}^{(i)} \neq 0 \) for \( 1 \leq i \leq \alpha \). Otherwise, it leads to a contradiction to Lemma 10. Similarly, for \( 1 \leq j \leq \alpha \), we can write \( s_{k+j-a+2} \) as
\[
\begin{align*}
s_{k+j-a+2} &= \lambda_{k+j,\alpha+2} \left(s_{k+j-1} \begin{array}{c}
\vdots \\
s_{k+j-a}
\end{array}\right) \\
&= \left(\lambda_{k+j,\alpha+2}^{(1)} \cdots \lambda_{k+j,\alpha+2}^{(\alpha)}\right) \left(s_{k+j-1} \begin{array}{c}
\vdots \\
s_{k+j-a}
\end{array}\right),
\end{align*}
\]
where \( \lambda_{k+j,\alpha+2} = (\lambda_{k+j,\alpha+2}^{(1)} \cdots \lambda_{k+j,\alpha+2}^{(\alpha)}) \in [p^q]^\alpha \).

For simplicity, we denote for \( 1 \leq j \leq \alpha \),
\[
B_j = \left(s_{k+j-1} \begin{array}{c}
\vdots \\
s_{k+j-a}
\end{array}\right), \\
C_j = \left(s_{k+j-1} \begin{array}{c}
\vdots \\
s_{k+j-a}
\end{array}\right).
\]

Obviously, all the \( B_j, C_j \)'s are invertible, and (22)-(24) can be respectively rewritten as below.
\[
\begin{align*}
\lambda_{k+1,\alpha+1} B_1 &\sim \lambda_{k+2,\alpha+1} B_2 \sim \cdots \sim \lambda_{k+a,\alpha+1} B_{\alpha}, \\
\lambda_{k+1,\alpha+1} C_1 &\sim \lambda_{k+2,\alpha+1} C_2 \sim \cdots \sim \lambda_{k+a,\alpha+1} C_{\alpha}, \\
\lambda_{k+1,\alpha+2} C_1 &\sim \lambda_{k+2,\alpha+2} C_2 \sim \cdots \sim \lambda_{k+a,\alpha+2} C_{\alpha}.
\end{align*}
\]
From Lemma 9 (a), we know that there exist diagonal matrices $\Lambda_j$ and $\Gamma_j$ such that $B_j = \Lambda_jB_1$ and $C_j = \Gamma_jC_1$ for $2 \leq j \leq \alpha$. Since $B_1$ and $C_1$ are invertible, it follows from (25)-(27) that

$$\begin{align*}
\lambda_{k+1,a+1} &\sim \lambda_{k+2,a+1}A_2 \sim \cdots \sim \lambda_{k+a,a+1}A_a, \\
\lambda_{k+1,a+1} &\sim \lambda_{k+2,a+1}A_2 \sim \cdots \sim \lambda_{k+a,a+1}A_a, \\
\lambda_{k+1,a+2} &\sim \lambda_{k+2,a+2}A_2 \sim \cdots \sim \lambda_{k+a,a+2}A_a.
\end{align*}$$

(28) 
(29) 
(30)

Then (28) and (29) imply that for $2 \leq j \leq \alpha$,

$$\lambda_{k+j,a+1} \sim \lambda_{k+j,a+1} \Lambda_j,$$

i.e.,

$$\left(\lambda_{k+j,a+1}^{(1)} \cdots \lambda_{k+j,a+1}^{(a)}\right) \Lambda_j \sim \left(\lambda_{k+j,a+1}^{(1)} \cdots \lambda_{k+j,a+1}^{(a)}\right) \Gamma_j.$$

Since all the components of $\lambda_{k+j,a+1}$ are nonzero, it follows

$$\lambda_j \sim \lambda_j, \quad \text{i.e.,} \quad \lambda_j = c \Gamma_j \text{ for some } c \in \mathbb{F}_q^*.$$ 

(31)

As a result, it follows from (30) and (31) that $\lambda_{k+1,a+2} \sim \lambda_{k+2,a+2}A_2 \sim \cdots \sim \lambda_{k+a,a+2}A_a$. Multiply each term by $B_1$ on the right, we get $k_{1,a+2}A_1B_1 \sim \lambda_{k+2,a+2}A_2B_2 \sim \cdots \sim \lambda_{k+a,a+2}B_1$, i.e.,

$$s_{k+1-a+2}A_1 \sim \cdots \sim s_{k+a-a+2}A_1.$$ 

(32)

However, by Lemma 9 (b), we know that

$$\text{rank}\left(\begin{array}{cccc}
\vdots \\
\vdots \\
\vdots \\
\vdots
\end{array}\right) \geq \alpha - t + 1 = d - k + 1 \geq 2,$$

which contradicts (32). Thus the theorem is proved. $\square$

**Remark 3.** For the case $d = 2k - 2 - t$, we get neither constructions of scalar MSCR codes nor nonexistence result of such codes. This is because our construction is confined to the product-matrix framework and the nonexistence result is restricted to the MSCR codes with invariant repair spaces. Actually, in our construction the repair space transferred from the helper node $j$ to the failed node $i$ only depends on $i$, that is, $s_{j-i} = s_i$, which is even stronger than the invariant repair spaces requirement. However, in the case $t = 1$, an $(n = 6, k = 4, d = 5)$ linear scalar MSR code is displayed in [20] where $s_{j-i}$ depends on both $i$ and $j$. Therefore, we conjecture that scalar MSCR codes exist for $d = 2k - 2 - t$ if more flexibility are introduced in the repair spaces.

**VI. CONCLUSIONS**

We explicitly construct scalar MSCR codes for all $d \geq \max(2k - 1 - t, k)$. The construction can be viewed as an extension of the product matrix code construction proposed in [3] for MSR and MBR codes. Just as in [3] where the product matrix-based MSR codes only applies when $d \geq 2k - 2$, our construction of MSCR codes also restricts to $d \geq 2k - 1 - t$. Both restrictions lead to the same limit on the information rate, i.e., $\frac{k}{d} \leq \frac{1}{2} + \frac{t}{2}$. As complementary results, the nonexistence of certain scalar MSR codes for $k < d < 2k - 3$ was presented in [19] and the nonexistence of certain scalar MSCR codes for $k < d < 2k - t - 2$ is proved in this paper. Along with this work, several results achieved so far for cooperative regenerating codes can be seen as the counterparts of the corresponding results in regenerating codes, such as the cut-set bound ([1] and [12], [14]) and the general construction of high-rate MSR codes and MSCR codes ([6] and [9]). On the one hand, the parameter bound and most constructions for cooperative regenerating codes tend to degenerate into those for regenerating codes when $t = 1$. However, on the other hand, it is not trivial to extend the results of regenerating codes to derive their counterparts in cooperative regenerating codes. An interesting question is how to generally build a cooperative regenerating code for repairing $t > 1$ erasures from regenerating codes that are designed for repairing individual node failures. Although we cannot solve this problem right now, we can predict that there should be more extensions in cooperative regenerating codes based on the fruitful research in regenerating codes.

**REFERENCES**

[1] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” IEEE Trans. Inf. Theory, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[2] C. Suh and K. Ramchandran, “Exact-repair MDS code construction using interference alignment,” IEEE Trans. Inf. Theory, vol. 57, no. 3, pp. 1425–1442, Mar. 2011.

[3] 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. Inf. Theory, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[4] S. Goparaju, A. Fazel, and A. Vardy, “Minimum storage regenerating codes for all parameters,” in Proc. IEEE Int. Symp. Inf. Theory, Oct. 2016, pp. 76–80.

[5] B. Sasidharan, G. K. Agrawal, and P. V. Kumar, “A high-rate MSR code with polynomial sub-packetization level,” in Proc. IEEE Int. Symp. Inf. Theory, Jun. 2015, pp. 2051–2055.

[6] M. Ye and A. Barg, “Explicit constructions of high-rate MDS array codes with optimal repair bandwidth,” IEEE Trans. Inf. Theory, vol. 63, no. 4, pp. 2001–2014, Apr. 2017.

[7] M. Ye and A. Barg, “Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization,” IEEE Trans. Inf. Theory, vol. 63, no. 10, pp. 6307–6317, Oct. 2017.

[8] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. M. Voelker, “Total recall: System support for automated availability management,” in Proc. 1st Conf. Networked Syst. Design Implement., San Francisco, CA, USA, Mar. 2004, p. 25.

[9] M. Ye and A. Barg, “Cooperative repair: Constructions of optimal MDS codes for all admissible parameters,” IEEE Trans. Inf. Theory, vol. 65, no. 3, pp. 1639–1656, Mar. 2019.

[10] 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 Trans. Inf. Theory, vol. 59, no. 5, pp. 2974–2987, May 2013.

[11] Z. Wang, I. Tamo, and J. Bruck, “Optimal rebuilding of multiple erasures in MDS codes,” IEEE Trans. Inf. Theory, vol. 63, no. 2, pp. 1084–1101, Feb. 2017.

[12] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, “Cooperative recovery of distributed storage systems from multiple losses with network coding,” IEEE J. Sel. Areas Commun., vol. 28, no. 2, pp. 268–276, Feb. 2010.

[13] A. Wang and Z. Zhang, “Exact cooperative regenerating codes with minimum-repair-bandwidth for distributed storage,” in Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM), Apr. 2013, pp. 400–404.

[14] K. W. Shum, “Cooperative regenerating codes for distributed storage systems,” in Proc. IEEE Int. Conf. Commun. (ICC), Kyoto, Japan, Jun. 2011, pp. 1–5.

[15] N. Le Scouarnec, “Exact scalar minimum storage coordinated regenerating codes,” in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2012, pp. 1197–1201.
[16] J. Chen and K. W. Shum, “Repairing multiple failures in the Sub-Ramchandran regenerating codes,” in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2013, pp. 1441–1445.

[17] K. W. Shum and J. Chen, “Cooperative repair of multiple node failures in distributed storage systems,” Int. J. Inf. Coding Theory, vol. 3, no. 4, pp. 299–323, 2016.

[18] J. Li and B. Li, “Cooperative repair with minimum-storage regenerating codes for distributed storage,” in Proc. IEEE Int. Conf. Comput. Commun., Apr./May 2014, pp. 316–324.

[19] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, “Interference alignment in regenerating codes for distributed storage: Necessity and code constructions,” IEEE Trans. Inf. Theory, vol. 58, no. 4, pp. 2134–2158, Apr. 2012.

[20] Z. Wang, I. Tamo, and J. Bruck, “Explicit minimum storage regenerating codes,” IEEE Trans. Inf. Theory, vol. 62, no. 8, pp. 4466–4480, Aug. 2016.

Yaqian Zhang received the B.S. degree in information and computational science from Northeastern University, Shenyang, China, in 2015. She is currently a Ph.D. student at the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. Her research interests include coding theory and information theory, with a current focus on regenerating codes.

Zhifang Zhang received the Ph.D. degree in Applied Mathematics from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, in 2007. Now she is an associate professor in the Academy of Mathematics and Systems Science, Chinese Academy of Sciences. Her research interests include cryptography, information security, and coding theory.