A Unified Form of Exact-MSR Codes via Product-Matrix Frameworks

Sian-Jheng Lin, Wei-Ho Chung, *Member, IEEE*, Yunghsiang S. Han, *Fellow, IEEE*, and Tareq Y. Al-Naffouri, *Member, IEEE*

***Abstract*— Regenerating codes represent a class of block codes applicable for distributed storage systems. The $[n, k, d]$ regenerating code has data recovery capability while possessing arbitrary k out of n code fragments, and supports the capability for code fragment regeneration through the use of other arbitrary d fragments, for $k \leq d \leq n-1$. Minimum storage regenerating (MSR) codes are a subset of regenerating codes containing the minimal size of each code fragment. The first explicit construction of MSR codes that can perform exact regeneration (named exact-MSR codes) for $d \geq 2k-2$ has been presented via a product-matrix framework. This paper addresses some of the practical issues on the construction of exact-MSR codes. The major contributions of this paper include as follows. A new product-matrix framework is proposed to directly include all feasible exact-MSR codes for $d \geq 2k-2$. The mechanism for a systematic version of exact-MSR code is proposed to minimize the computational complexities for the process of message-symbol remapping. Two practical forms of encoding matrices are presented to reduce the size of the finite field.**

***Index Terms*— Distributed storage, maximum-distance-separable (MDS) codes, MSR codes.**

I. Introduction

In A cloud storage system, data are stored in distributed storage nodes in the network. Reliability is one of the major challenges in the design of distributed storage systems. A robust distributed system can tolerate one or more node crashes. To improve system reliability, a well-known solution is to replicate data files on multiple distinct storage nodes [1]. However, the replicate-and-store operation demands substantial storage space. Thus, erasure codes are employed in the design of distributed storage systems to provide data reliability and storage efficiency. One popular class of erasure codes are maximum-distance-separable (MDS) codes, such as Reed-Solomon (RS) codes [2]. RS codes have been applied on Google [3] and Facebook distributed storage systems. In (n, k) RS codes, k data symbols are encoded and then n encoded symbols are distributed to n storage nodes. A client or a data collector (DC) can recover the original data by downloading encoded symbols from any k among all the storage nodes, by applying a data reconstruction procedure.

Because a distributed storage system is built on an unstable network environment, any storage node may fail, crash, or become disconnected. The data stored in the failed nodes must be reconstructed to maintain functionality of the system. The process of reconstructing the data in the failed node is called data regeneration. To reconstruct the failed node, a new (blank) replacement node is placed in the storage network. This replacement node connects to a subset of other active nodes, and then downloads the symbols necessary to restore the lost code fragment. A simple method for data regeneration is that the replacement node downloads all the data stored in k nodes, and then converts all the data to obtain the lost code fragment. However, it is not efficient to retrieve the entire set of B data symbols, to recover only a much smaller code fragment stored in a failed node. The pioneer work [4], [5] addressed this issue and introduced a new class of codes for distributed storage systems, termed "regenerating codes."

There exists a trade-off between the downloaded amount of data for node repairing and the size of each fragment stored in a node in regenerating codes. For the $[n, k, d]$ regenerating code, B message symbols are encoded into n code fragments, and the size of each code fragment is defined by α symbols. Each code fragment is distributed to and stored in its corresponding storage node in the network. A data collector (DC) can reconstruct the lost data by collecting k code fragments from various nodes. When a node failure occurs, the replacement node accesses d surviving nodes and downloads β symbols from each of them to perform code regeneration. For a well designed regenerating coding scheme, the data amount necessary to repair a failed node βd is expected to be much smaller than the size of message B. The regenerating code over $GF(q)$ is associated with the following parameters

$$\{n, k, d, \alpha, \beta, B\}$$

and the parameters follow the inequality

$$k \leq d \leq n-1.$$

Manuscript received January 27, 2014; revised August 28, 2014; accepted November 16, 2014. Date of publication December 5, 2014; date of current version January 16, 2015. This work was supported by the Ministry of Science and Technology, Taiwan, under Grants NSC 102-2221-E-001-006-MY2, MOST 103-3113-E-110-002, and NSC 101-2221-E-011-069-MY3. Part of this work was presented at the 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC 2013).

S.-J. Lin was with the Research Center for Information Technology Innovation, Academia Sinica, Taipei 11574, Taiwan. He is now with the Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia (e-mail: sjlin@citi.sinica.edu.tw).

W.-H. Chung is with the Research Center for Information Technology Innovation, Academia Sinica, Taipei 11529, Taiwan (e-mail: whc@citi.sinica.edu.tw).

Y. S. Han is with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 10607, Taiwan (e-mail: yshan@mail.ntust.edu.tw).

T. Y. Al-Naffouri is with the Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia, and also with the Department of Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran 34464, Saudi Arabia (e-mail: tareq.alnaffouri@kaust.edu.sa).

Communicated by O. Milenkovic, Associate Editor for Coding Theory.
Digital Object Identifier 10.1109/TIT.2014.2378255

There exists a trade-off between the storage space for each node and the total amount of downloaded data to repair a node. The theoretical storage-bandwidth bound is given by the cut-set bound of network coding [6]:

$$B \leq \sum_{i=0}^{k-1} min\{\alpha, (d-i)\beta\}. \quad (1)$$

Two extreme cases, termed as minimum bandwidth regenerating (MBR) and minimum storage regenerating (MSR), have been adequately investigated in prior studies [7]–[9]. For MBR, one first minimizes β and then α such that

$$\beta = 2B/\left(k(2d-k+1)\right),$$
$$\alpha = d\beta.$$

Similarly, by first minimizing α and then β, for MSR, we have

$$\alpha = B/k,$$
$$\beta = B/\left(k(d-k+1)\right).$$

In the case where the reconstructed fragment is exactly the same as the fragment stored in the failed node, we refer to this data regeneration as *exact regeneration*. This is a highly desired property in practical implementations. However, any effective and systematic approach to construct the exact-regeneration codes at the interior points (or region) of a storage-bandwidth bound curve [10] is typically unknown except at MSR and MBR points.

There have been many studies that focus on the design of regenerating codes [11]–[15]. In [16], the authors presented the Exact-MSR code for $n = d + 1$ and $k = 2$. In [17], practical Exact-MSR codes at $[n = 4, k = 2, d = 3]$ and $[n = 5, k = 3, d = 4]$ via computer search algorithms were discovered. The $[n = d + 1, k, d]$ Exact-MBR codes with repair-by-transfer ability were presented in [10]. The repair-by-transfer ability is a property where the node-repairing process does not require any arithmetic operations. Rashmi et al. [7] presented a general construction of exact regenerating codes at MSR and MBR points. By product-matrix form, they proposed a class of Exact-MBR codes for all feasible parameters $[n, k, d]$, and a class of Exact-MSR codes for the set of parameters $[n, k, d \geq 2k - 2]$. (Notice that the construction of Exact-MSR codes for $d < 2k - 3$ is non-achievable [18] for $\beta = 1$.). [19] and [20] investigate the inequality for parameters k and α on MSR codes at $d = n - 1$. Recently, a number of exact regenerating codes between MSR and MBR points [21]–[23] had been proposed. Those codes [21]–[23] improve the storage-space-sharing scheme [10] based on MSR and MBR codes. [24] showed a class of minimum-storage cooperative regenerating (MSCR) code to repair two failures cooperatively. The MSCR codes [24] is constructed directly from Exact-MSR codes.

There are critical challenges for designing regenerating codes, such as computational complexity, space complexity, and size of the finite field. These issues directly influence the overall efficiency of distributed storage systems. In this work, we present a new class of $[n, k, d \geq 2k - 2]$ Exact-MSR codes with lower computational complexity that operate using smaller finite fields. The contributions of this work are listed as follows.

1) A unified product-matrix form for $[n, k, d \geq 2k - 2]$ Exact-MSR codes is presented. The proposed product-matrix form can directly calculate the codes via matrix products without applying shortening techniques in code construction [7]. This property allows us to construct Exact-MSR codes with $d > 2k - 2$ without first constructing codes with $d' = 2k' - 2$.
2) A mechanism for a systematic version of Exact-MSR code is proposed. First, a conversion method is proposed to transform a valid instance of the encoding matrix into another matrix whose systematic part includes the minimum amount of non-zero entries. Second, with this converted encoding matrix, the proposed MSR codes can achieve lower leading factor in computational complexity and lower update complexity (See Section V-B for more details).
3) Two feasible forms of encoding matrices are presented. The relationship between field sizes q and code parameters $[n, k, d]$ is studied. In particular, the second form of encoding matrix requires a finite field $\mathbb{F}_q$ with $q \geq n$ for q a power of two. This reduces the field size of the product-matrix approach $q \geq n(n-k+1)$ shown in [7].
4) If the Vandermonde matrix is chosen as the encoding matrix, a new decoding algorithm is proposed to reduce the memory space used in the decoding process.

The rest of this paper is organized as follows. Section II reviews the Exact-MSR codes constructed in [7]. Section III presents the proposed product-matrix framework for Exact-MSR code and the node-repairing procedure with data reconstruction procedure on the proposed codes. Section IV discusses two practical forms of encoding matrices, with an analysis of the lower bounds of field sizes. Section V presents the conversion method and the message-symbol remapping procedure for a systematic version of the proposed codes. Section VI compares existing Exact-MSR codes with the proposed codes, in terms of time complexities, field sizes, and update complexities. Section VII concludes this work.

II. Preliminary Background

Throughout this paper, we use the superscript t to denote the transpose of a vector/matrix. For a matrix X, $x^{(i)}$ indicates the ith column of matrix X, and $x_{(i)}$ indicates the ith row of matrix X. The notation

$$\underset{m\times n}{\text{Vander}}(a_1, a_2, \ldots, a_n)$$

denotes a Vandermonde matrix of size $m \times n$, where the ith column is given by

$$\begin{bmatrix} 1 & a_i & a_i^2 & \ldots & a_i^{m-1} \end{bmatrix}^t.$$

In the following, we review the Exact-MSR codes presented in [7].

By applying a data striping technique, $[n, k, d, \alpha, \beta = 1, B]$ regenerating codes can be used to construct $[n, k, d, \alpha' = \alpha \times i, \beta' = \beta \times i, B' = B \times i]$ regenerating codes for any positive integer i. Thus, one only needs to consider the

constructions of Exact-MSR codes at $\beta = 1$ over $GF(q)$, and thus

$$\alpha = d - k + 1; \quad \beta = 1; \quad B = k(d - k + 1). \tag{2}$$

In [7], the authors present the construction of Exact-MSR code at $d = 2(k-1) = 2\alpha$. The construction can be represented as the matrix product

$$U \cdot G = C \tag{3}$$

of an $(k-1) \times d$ message matrix U and a $d \times n$ encoding matrix G. The information sequence $m = [m_0, m_1, \ldots, m_{B-1}]$ is arranged into

$$U = \begin{bmatrix} \underbrace{Z_1}_{(k-1)\times(k-1)} & \underbrace{Z_2}_{(k-1)\times(k-1)} \end{bmatrix}, \tag{4}$$

where Z_1 and Z_2 are symmetric matrices with dimension $(k-1) \times (k-1)$. The encoding matrix

$$G = \begin{bmatrix} \underbrace{\bar{G}}_{(k-1)\times n} \\ \underbrace{\bar{G}\Lambda}_{(k-1)\times n} \end{bmatrix} \tag{5}$$

should satisfy the following conditions:

1) Any d columns of G are linearly independent;
2) Λ is a diagonal matrix and its n diagonal elements are distinct;
3) Any $k-1$ columns of $\bar{G}$ are linearly independent.

A feasible encoding matrix with G as a Vandermonde matrix is suggested by [7]. In this case, the $\bar{G}$ is given by

$$\bar{G} = \underset{(k-1)\times n}{\text{Vander}}(x_1, x_2, \ldots, x_n), \tag{6}$$

where $\{x_i\}_{i=1}^n$ are $n \leq q$ distinct elements in $GF(q)$. Each element of

$$\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

is defined as

$$\lambda_i = x_i^{k-1}.$$

Since Λ requires all n elements of it to be distinct such that n is the lower bound of the field size q. [7] indicates that a field size $n(d-k+1)$ or higher suffices to make the Vandermonde matrix a feasible solution of G. After obtaining C, node i stores the ith column of C given by

$$c^{(i)} = U \cdot \begin{bmatrix} \bar{g}^{(i)} \\ \bar{g}^{(i)}\lambda_i \end{bmatrix} = Z_1\bar{g}^{(i)} + Z_2\bar{g}^{(i)}\lambda_i.$$

A. Node-Repairing Procedure

Assume node h_0 fails, and its stored data are

$$c^{(h_0)} = U \cdot \begin{bmatrix} \bar{g}^{(h_0)} \\ \bar{g}^{(h_0)}\lambda_{h_0} \end{bmatrix} = Z_1\bar{g}^{(h_0)} + Z_2\bar{g}^{(h_0)}\lambda_{h_0}. \tag{7}$$

To maintain the functionality of the storage system, a replacement node is placed into the network to replace the role of the failed node. The replacement node downloads a total of d symbols

$$\upsilon_{h_j} = \mu_{h_0} c^{(h_j)}, \quad \forall j \in \{j\}_{j=1}^d \tag{8}$$

from d helper nodes $\{h_j\}_{j=1}^d$, where

$$\mu_{h_0} = (\bar{g}^{(h_0)})^t. \tag{9}$$

In the replacement node, the d downloaded symbols form a d-element row vector $\Upsilon_{\text{repair}} = [\, \upsilon_{h_1}\ \upsilon_{h_2} \cdots \upsilon_{h_d}\,]$, that possesses the equality

$$\begin{aligned} \Upsilon_{\text{repair}} &= \mu_{h_0} U G_{\text{repair}} \\ &= \left[\bar{g}_{h_0} Z_1 \quad \bar{g}_{h_0} Z_2\right] G_{\text{repair}}, \end{aligned} \tag{10}$$

where $G_{\text{repair}} = [\, g_{h_1}\ g_{h_2} \cdots g_{h_d}\,]$ is the submatrix of G corresponding to d helper nodes. As G_{repair} is invertible (By condition (1) of the encoding matrix),

$$\Upsilon_{\text{repair}} G_{\text{repair}}^{-1} = \left[\, \bar{g}_{h_0} Z_1\ \bar{g}_{h_0} Z_2\,\right]. \tag{11}$$

The (7) can be computed by

$$\left(\bar{g}_{h_0} Z_1 + \lambda_{h_0} \bar{g}_{h_0} Z_2\right)^t.$$

B. Data Reconstruction Procedure

To reconstruct the message, DC respectively downloads k vectors $\{c^{(i_j)}\}_{j=1}^k$ from k active nodes, where $c^{(i_j)}$ is the i_jth column of C. Let $C_{\text{DC}} = \left[c^{(i_1)}\ c^{(i_2)}\ \ldots\ c^{(i_k)}\right]$ denote an $(k-1) \times k$ matrix, and G_{DC} is a $d \times k$ matrix including the corresponding k encoding columns in G. By definition, we have

$$C_{\text{DC}} = U G_{\text{DC}} = Z_1 \bar{G}_{\text{DC}} + Z_2 \bar{G}_{\text{DC}} \Lambda_{\text{DC}}. \tag{12}$$

The C_{DC} is pre-multiplied by the $k \times (k-1)$ matrix G_{DC}^t to obtain a $k \times k$ matrix

$$\bar{G}_{\text{DC}}^t C_{\text{DC}} = \bar{G}_{\text{DC}}^t Z_1 \bar{G}_{\text{DC}} + \bar{G}_{\text{DC}}^t Z_2 \bar{G}_{\text{DC}} \Lambda_{\text{DC}}. \tag{13}$$

(13) can be expressed as

$$P = \tilde{Z}_1 + \tilde{Z}_2 \Lambda_{\text{DC}}, \tag{14}$$

where

$$\begin{aligned} P &= \bar{G}_{\text{DC}}^t C_{\text{DC}}, \\ \tilde{Z}_1 &= \bar{G}_{\text{DC}}^t Z_1 \bar{G}_{\text{DC}}, \quad \tilde{Z}_2 = \bar{G}_{\text{DC}}^t Z_2 \bar{G}_{\text{DC}}. \end{aligned} \tag{15}$$

Since Z_1 (and Z_2) is symmetric, the congruence $\tilde{Z}_1$ (and $\tilde{Z}_2$) is also symmetric. From Condition 2) of the encoding matrix, all the non-diagonal entries of $\tilde{Z}_1$ and $\tilde{Z}_2$ can be solved.

Let $\bar{G}_{\text{DC}}$ denote the k column vectors $\{\bar{g}^{(1)}, \bar{g}^{(2)}, \ldots, \bar{g}^{(k)}\}$. Since all non-diagonal elements of $\tilde{Z}_1$ are known, the $k-1$ non-diagonal entries in the ith column are given by

$$\begin{bmatrix} (\bar{g}^{(1)})^t \\ \vdots \\ (\bar{g}^{(i-1)})^t \\ (\bar{g}^{(i+1)})^t \\ \vdots \\ (\bar{g}^{(k)})^t \end{bmatrix} Z_1 \bar{g}^{(i)}. \tag{16}$$

The left-most matrix in (16) is a $(k-1)\times(k-1)$ matrix that is invertible by Condition 3) of the encoding matrix. Thus, we can obtain

$$\left\{Z_1\bar{g}^{(i)} | 1 \le i \le k\right\}.$$

Then, one can take the first $k-1$ terms of $Z_1\bar{g}^{(i)}$ to obtain

$$Z_1[\bar{g}^{(1)} \dots \bar{g}^{(k-1)}].$$

The right matrix is also invertible such that we can solve Z_1. By following similar process, Z_2 can be extracted from $\tilde{Z}_2$.

III. Exact-MSR Code via Unified Product-Matrix Framework

In this section, we present the proposed product-matrix framework for $[n, k, d \ge 2k-2]$ Exact-MSR codes. In code construction, the B message symbols in information sequence m are arranged into an $\alpha \times d$ message matrix U that is multiplied by the $d \times n$ encoding matrix G to obtain a $\alpha \times n$ matrix $C = U \cdot G$. The ith column of C, denoted as $c^{(i)} = U \cdot g^{(i)}$, is stored in node i. The U and G matrices should satisfy a number of conditions elaborated as follows.

The message matrix is given by

$$U = \begin{bmatrix} \underbrace{Z_1}_{(k-1)\times(k-1)} & \underbrace{Z_2}_{(k-1)\times(k-1)} & \underbrace{T}_{(k-1)\times\omega} \\ \underbrace{\mathbf{0}}_{\omega\times(k-1)} & \underbrace{T^t}_{\omega\times(k-1)} & \underbrace{S}_{\omega\times\omega} \end{bmatrix}, \tag{17}$$

where $\omega = d - 2k + 2$, $\mathbf{0}$ is a $\omega \times (k-1)$ zero matrix. Z_1 and Z_2 are $(k-1)\times(k-1)$ symmetric matrices, and each matrix includes $\binom{k}{2}$ distinct message symbols. Specifically, the entries of the upper triangular part of Z_1 and Z_2, respectively, are filled with message symbols, and other entries are filled with corresponding values such that the symmetric property holds. T is a $(k-1)\times\omega$ matrix filled with message symbols. S is defined as a $\omega \times \omega$ matrix expressed as

$$S = \begin{bmatrix} s_0 & s_1 \\ s_1^t & \mathbf{0} \end{bmatrix}, \tag{18}$$

where $\mathbf{0}$ is a $(\omega-1)\times(\omega-1)$ zero matrix, s_0 is an information symbol, and s_1 is a $(\omega-1)$-element row vector. The message matrix includes a total of

$$\binom{k}{2} + \binom{k}{2} + (k-1)(d-2k+2) + (d-2k+2) = k(d-k+1)$$

information symbols,equal to the message size B given in (2).

The $d \times n$ encoding matrix is given by

$$G = \begin{bmatrix} \underbrace{\bar{G}\Lambda}_{(k-1)\times n} \\ \underbrace{\bar{G}}_{(k-1)\times n} \\ \underbrace{\Delta}_{\omega\times n} \end{bmatrix}, \tag{19}$$

where the sizes of matrices $\bar{G}$, Δ, and Λ are $(k-1)\times n$, $\omega \times n$, and $n \times n$, respectively. A feasible encoding matrix is required to satisfy the following conditions:

1) Any d columns of G are linearly independent;
2) Λ is a diagonal matrix and its n diagonal elements are distinct;
3) Any $k-1$ columns of $\bar{G}$ are linearly independent;
4) For $d > 2k-2$, any k columns of
$$\hat{G} = \begin{bmatrix} \bar{G} \\ \delta_{(1)} \end{bmatrix}, \tag{20}$$
are linearly independent, where $\delta_{(1)}$ denotes the first row of Δ.

Notice that the first three conditions are the same as the conditions [7] shown in Section II. When $d = 2k-2$, the encoding matrix and message matrix reduce to

$$U = \begin{bmatrix} Z_1 & Z_2 \end{bmatrix}, \quad G = \begin{bmatrix} \bar{G}\Lambda \\ \bar{G} \end{bmatrix}, \tag{21}$$

and are the same as those given in (4) and (5). With the above four conditions, the node-repairing procedure and data reconstruction procedure are presented as follows.

A. Node-Repairing Procedure

Assume the failed node is h_0 whose stored data are

$$\begin{aligned} c^{(h_0)} &= U \cdot g^{(h_0)} = U \begin{bmatrix} \bar{g}^{(h_0)}\lambda_{h_0} \\ \bar{g}^{(h_0)} \\ \delta^{(h_0)} \end{bmatrix} \\ &= \begin{bmatrix} Z_1\bar{g}^{(h_0)}\lambda_{h_0} + Z_2\bar{g}^{(h_0)} + T\delta^{(h_0)} \\ T^t\bar{g}^{(h_0)} + S\delta^{(h_0)} \end{bmatrix}. \end{aligned} \tag{22}$$

To reconstruct $c^{(h_0)}$, the replacement node downloads a total of d symbols

$$v_{h_j} = \mu_{h_0} c^{(h_j)} \tag{23}$$

from d helper nodes $\{h_j\}_{j=1}^{d}$, where

$$\mu_{h_0} = \begin{bmatrix} \bar{g}^{(h_0)} \\ \delta^{(h_0)} \end{bmatrix}^t = \begin{bmatrix} \bar{g}_{(h_0)} & \delta_{(h_0)} \end{bmatrix}. \tag{24}$$

In the replacement node, the d downloaded symbols form a d-element row vector $\Upsilon_{\text{repair}} = [\, v_{h_1}\ v_{h_2} \cdots v_{h_d}\,]$, that possesses the equality

$$\begin{aligned} &\Upsilon_{\text{repair}} \\ &= \mu_{h_0} U G_{\text{repair}} \\ &= \begin{bmatrix} \bar{g}_{(h_0)} & \delta_{(h_0)} \end{bmatrix} \begin{bmatrix} Z_1 & Z_2 & T \\ \mathbf{0} & T^t & S \end{bmatrix} G_{\text{repair}} \\ &= \begin{bmatrix} \bar{g}_{(h_0)}Z_1 & \bar{g}_{(h_0)}Z_2 + \delta_{(h_0)}T^t & \bar{g}_{(h_0)}T + \delta_{(h_0)}S \end{bmatrix} G_{\text{repair}}, \end{aligned} \tag{25}$$

where $G_{\text{repair}} = [\, g_{h_1}\ g_{h_2} \cdots g_{h_d}\,]$ is the submatrix of G corresponding to d helper nodes. From condition 1) of the encoding matrix, G_{repair} is invertible, and

$$\begin{aligned} &\Upsilon_{\text{repair}} G_{\text{repair}}^{-1} \\ &\quad = \begin{bmatrix} \bar{g}_{(h_0)}Z_1 & \bar{g}_{(h_0)}Z_2 + \delta_{(h_0)}T^t & \bar{g}_{(h_0)}T + \delta_{(h_0)}S \end{bmatrix}. \end{aligned} \tag{26}$$

From (26), the upper part of (22) can be solved by

$$\left(\lambda_{h_0}(\bar{g}_{(h_0)}Z_1) + (\bar{g}_{(h_0)}Z_2 + \delta_{(h_0)}T^t)\right)^t,$$

and the lower part of (22) is the transpose of $\bar{g}_{(h_0)}T + \delta_{(h_0)}S$.

B. Data Reconstruction Procedure

To reconstruct the message, DC respectively downloads k vectors $\{c^{(i_j)}\}_{j=1}^{k}$ from k active nodes, where $c^{(i_j)}$ is the i_jth column of C. Let $C_{\mathrm{DC}} = \left[c^{(i_1)}\ c^{(i_2)}\ \ldots\ c^{(i_k)}\right]$ denote an $(k-1)\times k$ matrix, and G_{DC} is a $d\times k$ matrix including the corresponding k encoding columns in G. By definition, we have

$$C_{\mathrm{DC}} = UG_{\mathrm{DC}} = \begin{bmatrix} Z_1 & Z_2 & T \\ \mathbf{0} & T^t & S \end{bmatrix} \begin{bmatrix} \bar{G}_{\mathrm{DC}}\Lambda_{\mathrm{DC}} \\ \bar{G}_{\mathrm{DC}} \\ \Delta_{\mathrm{DC}} \end{bmatrix}. \quad (27)$$

(27) can be split into two parts:

$$C_{\mathrm{DC}}^{\mathrm{U}} = \begin{bmatrix} Z_1 & Z_2 & T \end{bmatrix} \begin{bmatrix} \bar{G}_{\mathrm{DC}}\Lambda_{\mathrm{DC}} \\ \bar{G}_{\mathrm{DC}} \\ \Delta_{\mathrm{DC}} \end{bmatrix}, \quad (28)$$

$$C_{\mathrm{DC}}^{\mathrm{L}} = \begin{bmatrix} T^t & S \end{bmatrix} \begin{bmatrix} \bar{G}_{\mathrm{DC}} \\ \Delta_{\mathrm{DC}} \end{bmatrix}, \quad (29)$$

where $C_{\mathrm{DC}}^{\mathrm{U}}$ denotes the upper part of C_{DC} with $k-1$ rows, and $C_{\mathrm{DC}}^{\mathrm{L}}$ denotes the lower part of C_{DC} with $d-2k+2$ rows. We first solve (29) to obtain T and S. Then T is utilized for solving (28) to obtain Z_1 and Z_2. Notice that when $d = 2k-2$, there is no need to solve (29).

1) Solving (29): (29) can be reformulated as

$$C_{\mathrm{DC}}^{\mathrm{L}} = \begin{bmatrix} \hat{T}^t & \check{S} \end{bmatrix} \begin{bmatrix} \hat{G}_{\mathrm{DC}} \\ \check{\Delta}_{\mathrm{DC}} \end{bmatrix}, \quad (30)$$

where $\hat{G}_{\mathrm{DC}}$ denotes G_{DC} after appending the first row of Δ_{DC}, and $\check{\Delta}_{\mathrm{DC}}$ denotes Δ_{DC} after removing its first row. Similarly, $\hat{T}^t$ denotes T^t after appending the first column of S, and $\check{S}$ denotes S after removing its first column. Thus, by (18), we have $\check{S} = \begin{bmatrix} s_1 \\ \mathbf{0} \end{bmatrix}$, where the column vector $\begin{bmatrix} s_0 \\ s_1^t \end{bmatrix}$ is moved to the last column of $\hat{T}^t$.

Since all rows of $\check{S}$ are zero except the first row, the ith row, $i \neq 1$, of $C_{\mathrm{DC}}^{\mathrm{L}}$ can then be formulated as

$$(\hat{t}^{(i)})^t \hat{G}_{\mathrm{DC}},$$

where $\hat{t}^{(i)}$ denotes the the ith column of $\hat{T}$. By Condition 4) of the encoding matrix, $\hat{G}_{\mathrm{DC}}$ is invertible. Thus, we can solve all columns of $\hat{T}$ except the first column. The first row of $C_{\mathrm{DC}}^{\mathrm{L}}$ is formulated as

$$(\hat{t}^{(1)})^t \hat{G}_{\mathrm{DC}} + s_1 \check{\Delta}_{\mathrm{DC}}. \quad (31)$$

Since s_1 has been solved in the last row of $\hat{T}$, we can solve (31) to get $\hat{t}^{(1)}$.

2) Solving (28): (28) can be reformulated as

$$C_{\mathrm{DC}}^{\mathrm{U}} - T\Delta_{\mathrm{DC}} = Z_1\bar{G}_{\mathrm{DC}}\Lambda_{\mathrm{DC}} + Z_2\bar{G}_{\mathrm{DC}}. \quad (32)$$

By the obtained T, the left-hand side can be computed directly. A feasible way of solving (32) had been addressed in Section II-B. However, if $\bar{G}_{\mathrm{DC}}$ is a Vandermonde matrix, Appendix B shows an alternative method to solve Z_1 and Z_2. As compared with Section II-B, the alternative method does not need to split P into two matrices $\tilde{Z}_1$ and $\tilde{Z}_2$. Instead, the decoding operations are only applied to a single unique matrix so that the arithmetic complexity and space complexity can be reduced.

IV. Encoding Matrices

This section presents two versions of valid encoding matrices based on Vandermonde matrices and its variants. Each version of the encoding matrix gives a relationship between finite field size q and code parameters $[n,k,d]$, that will also be discussed. Notice that the lower bounds of the field sizes given in this section are for the constructions under our approaches, rather than the lower bound for all Exact-MSR codes by any constructions.

A. First Approach (Vandermonde Matrix)

In [7], the authors suggested to choose the Vandermonde matrix as G, for $d = 2k-2$. Here, we show that the Vandermonde matrix can also be chosen as the encoding matrix of proposed code. In this case, $\bar{G}$ is defined as

$$\bar{G} = \underset{(k-1)\times n}{\mathrm{Vander}}(x_1, x_2, \ldots, x_n),$$

that is the same with (6). Hence Condition 3) holds. Δ is defined as

$$\Delta = \underset{\omega\times n}{\mathrm{Vander}}(x_1, x_2, \ldots, x_n) \times \mathrm{diag}(x_1^{k-1}, x_2^{k-1}, \ldots, x_n^{k-1}).$$

$\hat{G}$ is given by combining $\bar{G}$ with the first row of Δ. It can be seen that $\hat{G}$ is also a Vandermonde matrix, so Condition 4) holds. Λ is defined as

$$\Lambda = \mathrm{diag}(x_1^{d-k+1}, x_2^{d-k+1}, \ldots, x_n^{d-k+1}). \quad (33)$$

By the above definitions, it can be seen that G is a Vandermonde matrix with moving $\bar{G}\Lambda$ to the last component of G. Hence, Condition 1) holds.

To satisfy Condition 2), we should carefully choose the elements $\{x_i\}_{i=1}^{n}$ such that $\{x_i^{d-k+1}\}_{i=1}^{n}$ are mutually distinct. This is the major bottleneck of the field size and Section IV-C will discuss this issue.

B. Second Approach (Shuffling Vandermonde Matrix)

In this approach, the encoding matrix is chosen as a modified Vandermonde matrix with reordering the rows in a specific order. In this case, the diagonal matrix is defined as

$$\Lambda = \mathrm{diag}(x_1, x_2, \ldots, x_n),$$

where $\{x_i\}_{i=1}^{n}$ are n distinct elements in $GF(q)$, to meet Condition 2). $\bar{G}$ is defined as

$$\bar{G} = \underset{(k-1)\times n}{\mathrm{Vander}}(x_1^2, x_2^2, \ldots, x_n^2).$$

We should carefully choose the elements $\{x_i\}_{i=1}^{n}$ such that $\{x_i^2\}_{i=1}^{n}$ are mutually distinct. Then $\bar{G}$ is a Vandermonde matrix and thus Condition 3) holds. Details will be discussed in Section IV-C. It can be shown that $\hat{G}$ is a Vandermonde matrix, so Condition 4) holds. The Δ is defined as

$$\Delta = \underset{\omega\times n}{\mathrm{Vander}}(x_1, x_2, \ldots, x_n) \times \mathrm{diag}(x_1^{2k-2}, x_2^{2k-2}, \ldots, x_n^{2k-2}).$$

Finally, G is expressed as

$$G = \begin{bmatrix} \bar{G}\Lambda \\ \bar{G} \\ \Delta \end{bmatrix} = \left[\begin{array}{cccc} x_1 & x_2 & \dots & x_n \\ x_1^3 & x_2^3 & \dots & x_n^3 \\ & & \vdots & \\ x_1^{2k-3} & x_2^{2k-3} & \dots & x_n^{2k-3} \\ \hline 1 & 1 & \dots & 1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ & & \vdots & \\ x_1^{2k-4} & x_2^{2k-4} & \dots & x_n^{2k-4} \\ \hline x_1^{2k-2} & x_2^{2k-2} & \dots & x_n^{2k-2} \\ x_1^{2k-1} & x_2^{2k-1} & \dots & x_n^{2k-1} \\ & & \vdots & \\ x_1^{d-1} & x_2^{d-1} & \dots & x_n^{d-1} \end{array}\right].$$

The rows in G can be reordered by the exponents of x_i, to obtain the geometric progression in each column $[\,1\ x_i\ x_i^2\ \dots\ x_i^{d-1}\,]^t$. Thus, G is a shuffled Vandermonde matrix, and Condition 1) holds.

C. Size of Finite Fields

In both encoding matrices we should choose n distinct elements $\{x_i\}_{i=1}^n$ in $GF(q)$ such that $\{x_i^b\}_{i=1}^n$ are also mutually distinct for $b = d-k+1$ (the first version) or $b = 2$ (the second version). The lower bound of field size q can be proven with the following proposition.

Proposition 1: Let a be a generator of $GF(q)$. Given a constant $b \in [2, q-1]$, all elements of $GF(q)$ are divided into $\frac{q-1}{g}+1$ non-overlapping sets, where $g = \gcd(b, q-1)$, as follows:

$$A_{-1} = \{0\};$$
$$A_i = \left\{a^{i+j(q-1)/g} | 0 \le j \le g-1\right\}, \quad \forall i = 0, 1, \dots, \frac{q-1}{g} - 1.$$

These individual sets possess the following properties:

1. *For any $x \in A_i$ and $i \ge 0$, we have $x^b = a^{ib}$.*
2. *Let $A = \{0\} \cup \{a^{ib}\}_{i=0}^{(q-1)/g-1}$. Then $|A| = \frac{q-1}{g} + 1$, i.e., all elements in A are distinct.*

Proof: Please see Appendix A. □

From Proposition 1, $\{x_i\}_{i=1}^n$ cannot include two distinct elements in each set A_i; otherwise, the two elements will equal to the same value a^{ib} after taking them to the power of b. Thus, the maximal size of $\{x_i\}_{i=1}^n$ is constructed by choosing only one element from each set A_i. Moreover, 0 can also be included in $\{x_i\}_{i=1}^n$, since each $a^{ib} \neq 0$. From the above observation, a corollary is obtained.

Corollary 1: Let $\{x_i | x_i \in GF(q)\}_{i=1}^n$ denote a set subject to

$$x_i \neq x_j, \ \text{and} \ \ x_i^b \neq x_j^b, \forall i \neq j.$$

The n is thus bounded by

$$n \le \frac{q-1}{\gcd(b, q-1)} + 1. \tag{34}$$

With Corollary 1, we plug $b = d-k+1$ and $b = 2$ to determine the field sizes of encoding matrices. For the first approach, substituting $b = d-k+1$ into (34) to get

$$n \le \frac{q-1}{\gcd(d-k+1, q-1)} + 1. \tag{35}$$

If $(d-k+1)$ and $(q-1)$ are co-prime, the field size $q \ge n$ suffices for the code construction based on the first approach. For the second approach, substituting $b = 2$ into (34) to get

$$n \le \frac{q-1}{\gcd(2, q-1)} + 1. \tag{36}$$

If field size q is a power of two, (36) implies $n \le q$; otherwise $n \le (q+1)/2$. In practical applications, codes are usually constructed over characteristic-2 finite fields, and thus $q \ge 2^{\lceil \log_2 n \rceil}$ suffices for the second approach.

The (35) can also be used to determine the field size of [7], as the proposed code with the first approach encoding matrix at $[d = 2k-2]$ is equivalent to the code [7]. By substituting $d = 2k-2$ into (35), the field size of MSR code [7] is given by

$$n \le \frac{q-1}{\gcd(k-1, q-1)} + 1, \tag{37}$$

which is tighter than the $q \ge n(n-k+1)$ shown in [7].

V. Systematic Version of Exact-MSR Codes

This section addresses the construction of the systematic version of Exact-MSR codes. In [7], the authors indicated that the systematic code can be constructed by applying a message-symbol remapping procedure to the non-systematic code. Let G_{SN} denote a $d \times k$ matrix consisting of k columns of encoding matrix G corresponding to the systematic positions in the systematic node, and $\bar{U}$ denote an $(\alpha \times k)$ information matrix containing B message symbols. The message-symbol remapping procedure is to solve U by

$$\bar{U} = UG_{\text{SN}}. \tag{38}$$

Notice that U should follow the form (17). Clearly, the procedure of solving (38) is equivalent to the data-reconstruction procedure. Thus, U and $\bar{U}$ are related in a linear way. After obtaining U, it can be used to calculate the code matrix $C = UG$. As G_{SN} is a submatrix of G, C will include a part that is exactly equivalent to the information matrix $\bar{U}$, and thus the code is systematic. Conversely, in the data reconstruction process, we need to compute the information matrix $\bar{U}$ by (38) after recovering U.

If G_{SN} is sparse, the conversion (compute $\bar{U}$ by given U) or the inversion (solve U by given $\bar{U}$) can take lower computational cost. Moreover, a sparse G_{SN} has lower update complexity, and this issue will be discussed in Section VI-C. In this section, we present a method to construct an encoding matrix for systematic code, where the G_{SN} embedded in the encoding matrix is sparse. Section V-A will present a conversion mechanism to convert a given valid encoding matrix $G_{(0)}$ into an encoding matrix of the systematic code, that we denote as G_{SC}. Here the "valid" means that the encoding matrix meets the conditions of the encoding matrix. The result G_{SC} is also valid and the G_{SN} embedded in G_{SC} is sparse and

achieves a minimum number of non-zero entries. If there is no confusion, we also use G_{SN} to indicate the systematic part of G_{SC}. With the result G_{SC}, Section V-B presents a new message symbol-remapping procedure with complexity of the order $O(B)$.

A. Conversion of Encoding Matrix

Given a valid encoding matrix

$$G_{(0)} = \begin{bmatrix} \bar{G}_{(0)}\Lambda_{(0)} \\ \bar{G}_{(0)} \\ \Delta_{(0)} \end{bmatrix},$$

this sub-section provides the conversion to generate a new matrix G_{SC} that includes the fewest possible non-zero entries in the first k columns. Here we assume that the systematic part is in the first k columns of G_{SC}. The conversion consists of two stages: the first stage converts the input $G_{(0)}$ into $G_{(1)}$, and the second stage converts the $G_{(1)}$ into G_{SC}. In each stage, we show that the result matrix is valid as long as the input matrix is valid.

1) First Stage: This stage converts $G_{(0)}$ into

$$G_{(1)} = \begin{bmatrix} \bar{G}_{(1)}\Lambda_{(1)} \\ \bar{G}_{(1)} \\ \Delta_{(1)} \end{bmatrix}.$$

In the following, we present the procedure that converts $G_{(0)}$ into $G_{(1)}$ such that the first $k-1$ columns of $\bar{G}_{(1)}$ is an identity matrix, the first $k-1$ columns of $\Delta_{(1)}$ is a zero matrix, and the kth column of $\bar{G}_{(1)}\Lambda_{(1)}$ is a zero column.

First, given

$$\Lambda_{(0)} = \mathrm{diag}(\lambda_1^{(0)}, \lambda_2^{(0)}, \ldots, \lambda_n^{(0)})$$

in $G_{(0)}$, the $\Lambda_{(1)}$ is defined as

$$\Lambda_{(1)} = \mathrm{diag}(\lambda_1^{(1)}, \lambda_2^{(1)}, \ldots, \lambda_n^{(1)}), \text{ where } \lambda_i^{(1)} = (\lambda_i^{(0)} - \lambda_k^{(0)}).$$

Notice that $\lambda_k^{(1)} = 0$.

Second, let $\bar{G}_{(0)}^{\mathrm{k-1}}$ denote the first $k-1$ columns of $\bar{G}_{(0)}$. $\bar{G}_{(0)}^{\mathrm{k-1}}$ is a $(k-1)\times(k-1)$ square matrix and invertible from Condition 3). Then, $\bar{G}_{(1)}$ is defined as

$$\bar{G}_{(1)} = (\bar{G}_{(0)}^{\mathrm{k-1}})^{-1}\bar{G}_{(0)},$$

whose first $k-1$ columns are formed as a $(k-1)\times(k-1)$ identity matrix.

Third, let $\Delta_{(0)}^{(\mathrm{k-1})}$ denote the first $k-1$ columns of $\Delta_{(0)}$. The size of $\Delta_{(0)}^{(\mathrm{k-1})}$ is $(d-2k+2)\times(k-1)$. $\Delta_{(1)}$ is defined as

$$\Delta_{(1)} = \Delta_{(0)} - \Delta_{(0)}^{(\mathrm{k-1})}\bar{G}_{(1)}.$$

As the first $k-1$ columns of $\bar{G}_{(1)}$ constitute an identity matrix, the first $k-1$ columns of $\Delta_{(0)}^{(\mathrm{k-1})}\bar{G}_{(1)}$ are $\Delta_{(0)}^{(\mathrm{k-1})}$. Hence, the first $k-1$ columns of $\Delta_{(1)}$ are filled with zeros. The validity of $G_{(1)}$ is demonstrated in Lemma 1 in Appendix C.

2) Second Stage: In this stage, we convert $G_{(1)}$ into a new encoding matrix

$$G_{\mathrm{SC}} = \begin{bmatrix} \bar{G}_{(1)}\Lambda_{(1)} \\ \bar{G}_{(1)} \\ \Delta_{(2)} \end{bmatrix}.$$

In the following, we present the procedure that converts $G_{(1)}$ into G_{SC} such that the kth column of $\Delta_{(2)}$ is $i^{(1)} = [10\ldots0]^t$.

Assume the kth column of $\Delta_{(1)}$ is $\left[\delta_k[1]\ \delta_k[2]\ \cdots\ \delta_k[d-2k+2]\right]^t$. The $\Delta_{(2)}$ is computed through

$$\Delta_{(2)} = \Delta_{(2)}^{\mathrm{TM}}\Delta_{(1)}, \tag{39}$$

where

$$\Delta_{(2)}^{\mathrm{TM}} = \begin{bmatrix} 1/\delta_k[1] & 0 & \ldots & 0 \\ \delta_k[2]/\delta_k[1] & -1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_k[d-2k+2]/\delta_k[1] & 0 & \ldots & -1 \end{bmatrix}. \tag{40}$$

In (40), we should show that $\delta_k[1] \neq 0$. If $\delta_k[1] = 0$, the first row of $\Delta_{(1)}$ will include k zero elements in the first, and Condition 4) cannot be satisfied. Thus, $\delta_k[1] \neq 0$. Then, G_{SC} is the desired encoding matrix. the validity is demonstrated in Lemma 2 in Appendix C.

Notably, by (59) and (62) in Appendix C, G_{SC} is linearly transformed from $G_{(0)}$ as

$$G_{\mathrm{SC}} = G_{\mathrm{SC}}^{\mathrm{TM}}G_{(1)}^{\mathrm{TM}}G_{(0)}.$$

In Appendix C, Theorem 1 shows that the first k columns of G_{SC} achieve the lower bound of the number of non-zero entries. The G_{SN} embedded in G_{SC} is expressed as

$$G_{\mathrm{SN}} = \begin{bmatrix} \Lambda_{\mathrm{SN}} & \mathbf{0} \\ I & \bar{g}_k^{(\mathrm{k})} \\ \mathbf{0} & i^{(1)} \end{bmatrix}, \tag{41}$$

where $\Lambda_{\mathrm{SN}} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{k-1})$, $\bar{g}_k^{(\mathrm{k})}$ is the kth column of $\bar{G}_{(1)}$, and column vector $i^{(1)}$ has 1 in the first position and 0 elsewhere.

B. Message-Symbol Remapping Procedure

Given the information matrix $\bar{U}$, this subsection presents the method to solve $\bar{U} = UG_{\mathrm{SN}}$, where G_{SN} is defined in (41), and the solved U is in the form (17). Based on (41), $\bar{U}$ is divided into four non-overlapping parts

$$\bar{U} = \begin{bmatrix} \bar{U}_{00} & \bar{U}_{01} \\ \bar{U}_{10} & \bar{U}_{11} \end{bmatrix}, \tag{42}$$

where the sizes of $\bar{U}_{00}$, $\bar{U}_{01}$, $\bar{U}_{10}$, $\bar{U}_{11}$ are $(k-1)\times(k-1)$, $(k-1)\times 1$, $\omega\times(k-1)$, and $\omega\times 1$, respectively. We have

$$\bar{U}_{00} = Z_1\Lambda_{(1)} + Z_2, \tag{43}$$

$$\bar{U}_{01} = Z_2\bar{g}_k^{(\mathrm{k})} + Ti^{(1)}, \tag{44}$$

$$\bar{U}_{10} = T^t, \tag{45}$$

$$\bar{U}_{11} = T^t\bar{g}_k^{(\mathrm{k})} + Si^{(1)}. \tag{46}$$

TABLE I

ARITHMETIC COMPLEXITIES OF [7] AND PROPOSED CODES IN SYSTEMATIC CASE, WHERE $i = d - 2k + 2$

	Encoding of systematic ver. (No including symbol remap.)	Message-symbol remapping	Data reconstruction (Without symbol remap.)	Node regeneration
[7]	$2(k+i-1)(n-k)(d+i)$	$10(k+i)^3$	$10(k+i)^3$	$2(d+i)^2$
Proposed codes	$2(k+i)(n-k)d$	$4k(k+i) = O(B)$	$2k^2(5k+i)$	$2d^2$

TABLE II

UPDATE COMPLEXITIES OF [7], [25], AND THE PROPOSED CODES

		Non-Systematic Ver.	Systematic Ver.
[7]	Update complexity	$2n$	$(k-1)(n-k)+1$
	♯ Nodes Updated	n	$n-k+1$
[26]	Update complexity	$2n-2k+4$	-
	♯ Nodes Updated	$n-k+3$	-
Ours	Update complexity	$2n-2k+4$	When $d = 2k-2$: $(2n-2k)+1$ Else: $3n-3k+1$
	♯ Nodes Updated	$n-k+3$	When $d = 2k-2$: $n-k$ Else: $n-k+1$

The steps are as follows.

1) Get T by (45).
2) Compute S via (46).
3) Compute non-diagonal entries of Z_1 and Z_2 by (43), where

$$Z_1[i,j] = \frac{-\bar{U}_{00}[i,j] + \bar{U}_{00}[j,i]}{\lambda_i - \lambda_j}, \forall i \neq j, \quad (47)$$

$$Z_2[i,j] = \frac{\lambda_i}{\lambda_i - \lambda_j}\bar{U}_{00}[i,j] + \frac{-\lambda_j}{\lambda_i - \lambda_j}\bar{U}_{00}[j,i], \quad \forall i \neq j. \quad (48)$$

4) Compute diagonal entries of Z_2 by (44).
5) Compute diagonal entries of Z_1 by (43), where

$$Z_1[i,i] = \frac{\bar{U}_{00}[i,i] - Z_2[i,i]}{\lambda_i}. \quad (49)$$

The computational complexity of the above procedure is analyzed as follows. Step 1) does not need any computational cost to solve T consisting of $\omega(k-1)$ symbols. In Step 2), we solve ω symbols in S via computing $\bar{U}_{11} - T^t\bar{g}_k^{(k)}$, which requires about $\omega(k-1)$ additions/multiplications. Thus, in Steps 1) and 2), each symbol requires about 2 finite field operations on average.

Step 3) solves $(k-1)(k-2)$ symbols. (47) requires 1 addition and 1 multiplication, and (48) requires 1 addition and 2 multiplications. Thus, each symbol takes about 2.5 finite field operations. In step 4), we compute $\bar{U}_{01} - Ti^{(1)} - Z_2\bar{g}_k^{(k)}$, where the diagonal entries of Z_2 are assigned to zeros. Then each element of above result vector is divided by each element of $\bar{g}_k^{(k)}$ to get the diagonal entries of Z_2. This step needs about $(k-1)^2$ additions/multiplications. In step 5), (49) only requires 1 addition/multiplication. In steps 3), 4) and 5), we solve a total of $(k-1)^2$ entries, and take $O((k-1)(k-2) + (k-1)^2 + (k-1)) = O((k-1)^2)$ operations. Thus, each symbol solved in Steps 3), 4), and 5) also takes constant operations. Thus, the computational complexity is proportional to the size of message B, and subsequently the computational complexity is $O(B)$.

VI. DISCUSSIONS

This section presents comparisons of Exact-MSR codes in [7] and the proposed version, in terms of field sizes, time complexities, and update complexities. By of [7, Corollary 8] the $[n, k, d \geq 2k-2]$ Exact-MSR codes [7] are the shortening codes of $[n' = n+i, k' = k+i, d' = d+i]$ codes, where $i = d - 2k + 2 \geq 0$. Thus, the size of the encoding matrix is $n \times (2d - 2k + 2)$, and the size of the message matrix is $(2d - 2k + 2) \times (d - k + 1)$. The analysis of code [7] is for the parameter $[n', k', d']$. Table I tabulates the results for time complexities, and Table II tabulates the results for update complexities.

A. Size of Finite Fields

In (2), the definition of $[\alpha, \beta, B]$ is in the number of symbols over field $GF(q)$. For each symbol of $GF(q)$, the symbol should take $\log_2(q)$ bits to record the value. Thus, the bit sizes of the $[n, k, d]$ Exact-MSR code at $\beta = 1$ over $GF(q)$ is given by

$$\alpha'' = \alpha \cdot \log_2(q); \quad \beta'' = \log_2(q); \quad B'' = k(d-k+1) \cdot \log_2(q). \quad (50)$$

Hence, the field size q is important for the bit sizes of codes. In of [7, Sec. V] the authors indicate that the field size $q \geq n(d-k+1)$ suffices for the Exact-MSR code at $d = 2k-2$. However, this bound is not tight, and a tight bound is shown in (37). For $d > 2k-2$, a precise value of the required field size is obtained by plugging $[n', k', d']$ into (37). In contrast, by using the encoding matrix defined in Section IV-B, the field size n suffices for the proposed code. The improvement is from two techniques: First, the unified framework can directly generate the $[n, k, d \geq 2k-2]$ codes without applying shortening technique. Second, a new encoding matrix (Section IV-B) is proposed to satisfy the four conditions over smaller fields.

B. Time Complexities

In this subsection, we study the number of finite field operations in encoding, Node-regeneration, and Data-reconstruction procedures.

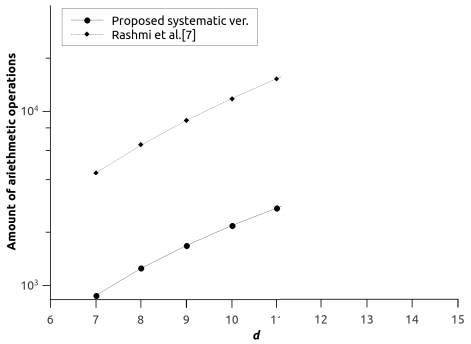

Fig. 1. Average number of arithmetic operations (additions and multiplications) used in [7] and proposed codes, where $n = 16$, $k = 4$.

1) Encoding Procedure: As the points of the complexities in systematic and non-systematic versions are the same, we only discuss the complexities of systematic version. For the systematic case, we do not need to calculate the systematic part of codeword, since this part is known. The following discussion is based on the observation.

The encoding procedure [7] is represented as a matrix product $C = UG$, where the sizes of U and G are $(d'-k'+1)\times d'$ and $d' \times n'$, respectively. If the columns corresponding to systematic part are removed, the size of encoding matrix is $d' \times (n-k)$. Thus, [7] requires a total of $(d'-k'+1)(n-k)d' = (d-k+1)(n-k)(d+i)$ multiplications and $(d'-k'+1)(n-k)(d'-1) = (d-k+1)(n-k)(d+i-1)$ additions. For the proposed code, the sizes of U and G are $(d-k+1)\times d$ and $d \times n$, respectively. After removing the systematic part, the size of encoding matrix is $d \times (n-k)$. Thus, it requires a total of $(d-k+1)(n-k)d$ multiplications and $(d-k+1)(n-k)(d-1)$ additions.

The message-symbol remapping procedure is an essential part for systematic codes, that takes around $5k'^3$ multiplications and $5k'^2(k'-1)$ additions (the details will be addressed later). For the proposed codes, the message-symbol remapping procedure in Sec. V requires the complexity of order $O(B)$, which is lower than that for data-reconstruction $O(k'^3)$ in [7]. Furthermore, for $d > 2k-2$, the code shortening process in [7] requires performing message-symbol remapping procedure to convert the longer code into systematic form. Hence, the message-symbol remapping procedure is required.

As shown in Table I, the gain of complexity reduction depends on the value of $i = d-2k+2$. Moreover, the ratio of leading constant is about 6 : 1. The primary cause is the high cost of the message-symbol remapping procedure in [7].

Figure 1 gives an example for the number of arithmetic operations (additions and multiplications) used in the encoding of [7] and the proposed approach when $n = 16$, $k = 4$, and $7 \le d \le 15$. Since the number of arithmetic operations is dependent on the information U, we average the number over randomly generated U. In this example, the number of operations used in the proposed algorithm is approximately 18% of [7] on average for $d = 7, 8, \ldots, 15$.

2) Node-Regeneration Procedure: In node regeneration, the replacement node receives d symbols from the connected nodes. Then this node computes the code fragment stored in the failed code. In [7], the complexity of solving a code fragment is of the order $O(d'^2)$, where $d' = d+i = 2d-2k+2$. In the proposed codes, the complexity of solving a code fragment is $O(d^2)$.

3) Data-Reconstruction Procedure: The data reconstruction procedure in [7] is elaborated upon in Section III-B2. The domination of the complexity is the matrix multiplications in the data-reconstruction procedure. (13) needs a matrix multiplication. Moreover, after obtaining $\tilde{Z}_1$ and $\tilde{Z}_2$, one need to solve (15) to get Z_1 and Z_2. Here, we treat this step as by two matrix multiplications. In summary, the data-reconstruction procedure requires about five matrix multiplications, where the size of matrix is $k' \times k'$. Thus, it takes about $5k'^3$ multiplications and $5k'^2(k'-1)$ additions.

For the proposed codes, data reconstruction consists of two parts, as given in Sec. III-B1 and Sec. III-B2. To solve (29) in Sec. III-B1, the number of arithmetic operations is no more than $(d-2k+2)(k^2+1)$ multiplications and $(d-2k+2)(k^2+1)$ additions. In Sec. III-B2, as the size of the matrix is $k \times k$, those matrix multiplications take approximately $5k^3$ multiplications and $5k^3$ additions. In summary, the proposed codes require approximately $k^2(d+3k+2)$ multiplications and $k^2(d+3k+2)$ additions. To emphasize the differences, we use $i = d-2k+2 \ge 0$; then, we have $5k'^3 = 5(k+i)^3$ and $k^2(d+3k+2) = k^2(5k+i)$. Thus, the proposed code has lower computational cost.

C. Update Complexities

Update complexity is defined as the maximal number of codeword symbols that should be updated while a single source symbol is modified. Consider Reed-Solomon codes, the systematic versions have lower update complexities than non-systematic versions. This is because the codeword in systematic part does not need to be updated, except for the source symbol we want to modify. Thus, [26], [25] proposed a class of update-efficient exact regenerating codes based on systematic Reed-Solomon codes. This section analyzes update complexities of three practical versions of Exact-MSR codes: [7], [25], and ours. Table II tabulates the results.

Assume the source message is a zero sequence, and thus, U and the generated code $C = UG$ are zero matrices. Then, a source symbol is assigned to non-zero to get a updated massage matrix $U' = [Z_1', Z_2']$, and the number of non-zero entries in codeword $C' = U'G$ is the update complexity of this code. W.L.O.G., assume the nonzero source symbol is arranged in Z_1'. Owing to the symmetry condition imposed on Z_1', there are two non-zero entries $Z_1'[i,j]$ and $Z_1'[j,i]$ in Z_1', for $i \ne j$. Otherwise, there is only one nonzero entry $Z_1'[i,i]$ in Z_1', for $i = j$.

1) Update Complexity of [7]: [7] presents the non-systematic codes (at $d = 2k-2$) and their systematic version (for $d \ge 2k-2$). For the non-systematic versions, suppose the non-zero entry is at $Z_1'[i,j]$ and $Z_1'[j,i]$, then the code matrix $C' = U'G$ has non-zero entries at the ith and jth rows. Thus, the number of non-zero entries in C' is $2n$ stored in n nodes. Notably, when the modified element is at the diagonal of Z_1', the update complexity is reduced to n.

The systematic version [7] is the shortening code of $[n'; k'; d' = 2k' - 2]$. In this case, U' is computed from $\bar{U}' = U'G_{\mathrm{SN}}$, where $\bar{U}'$ is filled with zeroes excluding a non-zero entry somewhere, and G_{SN} is a Vandermonde matrix. Application of the data reconstruction procedure shows that most of the entries of U' are non-zero. Thus, the code matrix $C' = U'G$ has numerous non-zero entries in the non-systematic part. Hence, we should update up to a total of $(k-1)(n-k)+1$ entries in $(n-k+1)$ nodes.[1] The analysis results have been included in Table II.

2) Update Complexity of [25]: Based on $[n, k-1]$ systematic Reed-Solomon codes, [25] presents a class of update-efficient Exact-MSR codes at $d = 2k-2$. [25] showed that its proposed codes have lower update complexity. As shown in [26, Sec. III] of, the encoding matrix is defined as

$$G = \begin{bmatrix} \bar{G}\Lambda \\ \bar{G} \end{bmatrix},$$

where $\bar{G}$ is the $(k-1) \times n$ encoding matrix of an $[n, k-1]$ systematic Reed-Solomon code. Since each row of G is with $n-k+2$ non-zero entries, the number of updated entries in C' is $2(n-k+2)$ or $n-k+2$ when the modified element is a diagonal entry in Z'_1. As the codes in [25] are not systematic codes, we list the analysis result in non-systematic column of Table II.

3) Update Complexity of Proposed Code: The update complexity of the proposed code is determined by the position of the modified symbol. For the non-systematic version, we choose the encoding matrix as the G_{SC} shown in Section V. By definition, the message matrix consists of four sub-matrices Z_1, Z_2, T, and S. In the following, we consider the update complexity by the position of the modified source symbol in each submatrix:

- $Z'_1[i, j]$ and $Z'_1[j, i]$: In this case, the code matrix $C' = U'G$ has non-zero entries at row i and row j. By definition of G_{SN}, there are two non-zero entries in the first $k-1$ columns of C', and the kth column of C' is a zero column. Thus, we should update $2(n-k)+2$ entries stored in $n-k+2$ nodes. Notably, if the modified entry is at the diagonal position $Z'_1[i, i]$, then $C' = U'G$ has non-zero entries in row i. In this case, we should update $n-k+1$ entries stored in $n-k+1$ nodes.
- $Z'_2[i, j]$ and $Z'_2[j, i]$: In this case, the code matrix $C' = U'G$ has non-zero entries at row i and row j. By definition of G_{SN}, there are two non-zero entries in the first $k-1$ columns of C', and two non-zero entries in the kth column of C'. Thus, we should update $2(n-k)+4$ entries stored in $n-k+3$ nodes. Notably, if the modified entry is at the diagonal position $Z'_2[i, i]$, then $C' = U'G$ has non-zero entries in row i. In this case, we should update $n-k+2$ entries stored in $n-k+2$ nodes.
- $T'[i, j]$ and $(T^t)'[j, i]$: In this case, the code matrix $C' = U'G$ has non-zero entries at row i and row $j+k-1$. By definition of G_{SN}, there is a non-zero entry in the first $k-1$ columns of C'. If $j = 1$, the kth column of C' has two non-zero entries; otherwise, the kth column of C' constitute a non-zero entry. Thus, we should update $2(n-k)+2$ or $2(n-k)+3$ entries stored in $n-k+2$ nodes.
- $S'[i, 1]$ and $S'[1, i]$: In this case, the code matrix $C' = U'G$ has non-zero entries at row $k-1+i$. By definition of G_{SN}, it can be seen that the first $k-1$ columns of C' is a zero matrix. Thus, we should update $n-k+1$ entries stored in $n-k+1$ nodes.

Table II records the worst case when the modified source symbol is at $Z'_2[i, j]$ and $Z'_2[j, i]$, for $i \neq j$. In this case, we need to update $2(n-k)+4$ entries in $n-k+3$ nodes.

For the systematic version, $\bar{U}$ in (42) can be divided into four parts $\bar{U}_{00}$, $\bar{U}_{01}$, $\bar{U}_{10}$, and $\bar{U}_{11}$. In the following, we consider the update complexity by the position of the modified source symbol in each submatrix.

- $\bar{U}'_{00}[i, j]$: By (47) and (48), the computed Z'_1 and Z'_2 in U' have four non-zero entries $Z'_1[i, j]$, $Z'_1[j, i]$, $Z'_2[i, j]$, and $Z'_2[j, i]$. Hence, the code matrix $C' = U'G$ has non-zero entries at row i and row j. Then by (44), both $Z'_2[i, i]$ and $Z'_2[j, j]$ are non-zeros entries. By (49), $Z'_1[i, i]$ and $Z'_1[j, j]$ are also non-zeros entries. As the code is systematic, the systematic part of C' does not include non-zero entries, except at the position of the modified message symbol. Thus, we should update $2(n-k)+1$ entries stored in $n-k+1$ nodes. Notably, if the modified entry is a diagonal entry in $\bar{U}'_{00}[i, i]$, then $Z'_1[i, i]$ is non-zero, and $C' = U'G$ has non-zero entries in row i. Thus, we should update $n-k+1$ entries stored in $n-k+1$ nodes in this case.
- $\bar{U}'_{01}[i]$: In this case, we have non-zero entries at $Z'_2[i, i]$ and $Z'_1[i, i]$. With this U', the code $C' = U'G$ has non-zero entries at row i, and hence we should update $n-k+1$ entries stored in $n-k+1$ nodes.
- $\bar{U}'_{11}[i]$: If $i \neq 1$, we have two non-zero entries at $S'[1, i]$ and $S'[i, 1]$. Thus, the code matrix $C' = U'G$ has non-zero entries at row $k-1+i$ and row k, then we should update $2(n-k)+1$ entries stored in $n-k+1$ nodes. If $i = 1$, we have a non-zero entry at $S'[1, 1]$. The code matrix $C' = U'G$ has non-zero entries at row k, and we should then update $n-k+1$ entries stored in $n-k+1$ nodes.
- $\bar{U}'_{10}[i, j]$: In this case, we have four non-zero entries at $T^t[i, j]$, $T[j, i]$, $S'[1, i]$, and $S'[i, 1]$ in U'. By this U', the code matrix $C' = U'G$ has non-zero entries in row $k-1+i$, row k, and row j. Thus, we should update $3(n-k)+1$ entries stored in $n-k+1$ nodes.

Table II records the worst case of update complexity. For $d = 2k-2$, $\bar{U}$ includes only $\bar{U}_{00}$ and $\bar{U}_{01}$ such that the update complexity is $2n-2k+1$ stored in $n-k$ nodes. For $d > 2k-2$, the worst case is when the modified source symbol is at $\bar{U}'_{10}[i, j]$. In this case, we need to update $3(n-k)+1$ symbols in $n-k+1$ nodes.

VII. Concluding Remarks

Regenerating codes have good performance to repair the node failures. However, some regenerating codes require quite expensive computational cost, as compared with common

[1] This has been verified by simulations for some MSR codes.

MDS codes, such as Reed-Solomon codes. To facilitate practical implementations of regenerating codes, we investigated several important issues pertaining to the construction of Exact-MSR codes. First, a unified framework of an $[n, k, d \geq 2k-2]$ Exact-MSR code is proposed. The unified framework is operated on smaller encoding/message matrices without having to apply the shortening technique. This gives a clear representation about the code constructions of Exact-MSR codes at $[n, k, d \geq 2k-2]$. Second, two practical forms of the encoding matrix are proposed. By the proposed encoding matrix, the field size is only n over the extended binary field, and thus the bit sizes (50) of Exact-MSR codes can be reduced. Third, an encoding matrix conversion method is proposed. This reduces the update complexity and the leading constant (around 1 vs 6) hidden in big-O encoding complexity. In some practical distribution storage systems, the number of nodes n is not very large, and thus the leading constant is also quite important. Finally, we presented a new data reconstruction algorithm in the case of an encoding matrix in the form of a Vandermonde matrix, with the aim of reducing the space cost and encoding cost. These proposed techniques facilitate practical implementations. Note that Exact-MSR code for $d < 2k-3$ is non-achievable for $\beta = 1$ [18] such that the code rate for Exact-MSR is roughly less than $1/2$. The proposed unified framework does not give constructions for any new set of parameters, and thus the construction for $[n, k, d = 2k-3]$ Exact-MSR code is still unknown.

APPENDIX A
PROOF OF PROPOSITION 1

It is clear that each element of $GF(q)$ is exactly assigned to a set of $\{A_i\}_{i=-1}^{(q-1)/g-1}$. By definition, two properties are proven as follows:

1) Note that $a^{q-1} = 1$. For $x \in A_i$ and $i \geq 0$, then $x = a^{i+j(q-1)/g} \in A_i$, and we have
$$(a^{i+j(q-1)/g})^b = a^{ib+jb(q-1)/g} = a^{ib+j\cdot \mathrm{lcm}(b,q-1)} = a^{ib}.$$
2) Because 0 is evidently excluded from $\{a^{ib}\}_{i=0}^{(q-1)/g-1}$, we only consider two elements $a^{i_1 b}$ and $a^{i_2 b}$ taken from the set $\{a^{ib}\}_{i=0}^{(q-1)/g-1}$, where $0 \leq i_1, i_2 \leq (q-1)/g - 1$ and $i_1 \neq 1_2$. We prove it by contradiction. Suppose $a^{i_1 b} = a^{i_2 b}$ and $i_1 > i_2$, then we have
$$\begin{aligned} a^{i_1 b} &= a^{i_2 b} \\ &\Rightarrow a^{(i_1-i_2)b \pmod{(q-1)}} = 1 \\ &\Rightarrow (q-1) | (i_1-i_2)b \\ &\Rightarrow (q-1)/g | (i_1-i_2)(b/g) \\ &\Rightarrow (q-1)/g | (i_1-i_2) \end{aligned}$$
since $(q-1)/g$ and b/g are co-prime. However, $(i_1 - i_2) \leq (q-1)/g - 1$ is defined such that $(q-1)/g$ cannot divide $(i_1 - i_2)$ and it leads to a contradiction. This completes the proof.

APPENDIX B
SOLVING $\tilde{Z}_1$ AND $\tilde{Z}_2$ IN (32)

As shown in Section III-B2, if the decoding of (16) is achieved by multiplying inverse matrices, it needs to store a total of k inverse matrices with size $(k-1) \times (k-1)$ for each or calculate inverses for two Vandermonde matrices. This section presents another decoding algorithm that needs to store only a $k \times k$ inverse matrix. The proposed algorithm has a condition that $\tilde{Z}_1$ (and $\tilde{Z}_2$) can be viewed as the results of polynomial evaluations. For example, when $\bar{G}_{\mathrm{DC}}$ is a Vandermonde matrix, $\tilde{Z}_1 = \bar{G}_{\mathrm{DC}}^t Z_1 \bar{G}_{\mathrm{DC}}$ can be treated as polynomial evaluations. In this case, the information polynomial for Z_1 is defined as
$$Z_1(x, y) = \sum_{1 \leq i,j \leq k-1} Z_1[i, j] x^{i-1} y^{j-1}, \tag{51}$$
where the degree of $Z_1(x, y)$ is at most $k-2$ in both x and y. Consequently, each entry of $\tilde{Z}_1$ can be viewed as the evaluation result $Z_1(\bar{x}_i, \bar{x}_j) = \tilde{Z}_1[i, j]$ for $1 \leq i, j \leq k$, where the set of evaluation points $\{\bar{x}_i\}_{i=1}^k$ is given by $\bar{G}_{\mathrm{DC}}$. Notice that the results of $Z_1(z, z)$ at $z \in \{\bar{x}_i\}_{i=1}^k$ are unknown, since we do not have the diagonal values of $\tilde{Z}_1$. Similarly, the information polynomial for Z_2 is given by
$$Z_2(x, y) = \sum_{1 \leq i,j \leq k-1} Z_2[i, j] x^{i-1} y^{j-1}, \tag{52}$$
and we have the evaluation results $Z_2(\bar{x}_i, \bar{x}_j) = \tilde{Z}_2[i, j]$ for $1 \leq i, j \leq k$ and $i \neq j$.

By combining $Z_1(x, y)$ and $Z_2(x, y)$, an information polynomial is defined as
$$Z_{\mathrm{SS}}(x, y) = Z_1(x, y) y + Z_2(x, y). \tag{53}$$
The degree of $Z_{\mathrm{SS}}(x, y)$ is at most $k-2$ in x and $k-1$ in y. The coefficients of $Z_{\mathrm{SS}}(x, y)$ are denoted as a $(k-1) \times k$ matrix Z_{SS}, where $Z_{\mathrm{SS}}[i, j]$ is the coefficient of the term $x^{i-1} y^{j-1}$ in $Z_{\mathrm{SS}}(x, y)$. We will show that Z_1 and Z_2 can be extracted from Z_{SS}, and thus the goal is to solve the coefficients of $Z_{\mathrm{SS}}(x, y)$. By $\tilde{Z}_1$ and $\tilde{Z}_2$, the evaluations of $Z_{\mathrm{SS}}(x, y)$ at $x, y \in \{\bar{x}_i\}_{i=1}^k$ can be easily calculated as
$$Z_{\mathrm{SS}}(\bar{x}_i, \bar{x}_j) = \tilde{Z}_1[i, j] \bar{x}_j + \tilde{Z}_2[i, j], \quad \forall i, j = 1, 2, \ldots, k. \tag{54}$$
However, the evaluations $Z_{\mathrm{SS}}(z, z)$ at $z \in \{\bar{x}_i\}_{i=1}^k$ are still unknown. These unknown values will increase the difficulty of solving $Z_{\mathrm{SS}}(x, y)$.

To handle these unknown values, $Z_{\mathrm{SS}}(x, y)$ is multiplied with $(x - y)$ to get
$$Z_{\mathrm{S}}(x, y) = Z_{\mathrm{SS}}(x, y)(x - y), \tag{55}$$
where the degree of $Z_{\mathrm{S}}(x, y)$ is at most $k-1$ in x and k in y. Let $\tilde{Z}_{\mathrm{S}}$ denote a $k \times k$ matrix consisting of the evaluation results $Z_{\mathrm{S}}(x, y)$ for $x, y \in \{\bar{x}_i\}_{i=1}^k$. By $\tilde{Z}_1$ and $\tilde{Z}_2$, $\tilde{Z}_{\mathrm{S}}$ can be easily calculated as
$$\tilde{Z}_{\mathrm{S}}[i, j] = (\tilde{Z}_1[i, j] \bar{x}_j + \tilde{Z}_2[i, j])(\bar{x}_i - \bar{x}_j), \quad \forall i, j = 1, 2, \ldots, k. \tag{56}$$
It is noteworthy that $\tilde{Z}_{\mathrm{S}}$ does not include any unknown values. Specifically, the diagonal of $\tilde{Z}_{\mathrm{S}}$ is assigned to zeroes $\tilde{Z}_{\mathrm{S}}[i, i] = 0$, unlike the unknown values in the diagonals of $\tilde{Z}_1$ and $\tilde{Z}_2$.

Next, we solve the coefficients of polynomials $\{Z_{\mathrm{S}}(x, \bar{x}_j)\}_{j=1}^k$, where $Z_{\mathrm{S}}(x, \bar{x}_j)$ is a polynomial with

$y = \bar{x}_j$ on $Z_S(x, y)$. By the known values $\{Z_S(\bar{x}_i, \bar{x}_j)\}_{i=1}^k$ in the ith row of $\tilde{Z}_S$, we can interpolate $Z_S(x, \bar{x}_j)$ through Lagrange polynomials, because $\deg(Z_S(x, \bar{x}_j)) \leq (k-1)$. Then, the obtained $Z_S(x, \bar{x}_j)$ is divided by $(x - \bar{x}_j)$ to get the coefficients of the polynomial

$$\begin{aligned} Z_{SS}(x, \bar{x}_j) &= Z_S(x, \bar{x}_j)/(x - \bar{x}_j) \\ &= Z_1(x, \bar{x}_j)\bar{x}_j + Z_2(x, \bar{x}_j). \end{aligned} \tag{57}$$

After $\{Z_{SS}(x, \bar{x}_j)\}_{j=1}^k$ has been obtained, the next step is to calculate the coefficients of $Z_{SS}(x, y)$. This step can be performed by applying a polynomial interpolation to $\{Z_{SS}(x, \bar{x}_j)\}_{j=1}^k$.

Finally, we split Z_{SS} into two $(k-1) \times (k-1)$ symmetry matrices, Z_1 and Z_2. By (53), the relationship of Z_{SS} with Z_1 and Z_2 is given by

$$Z_{SS}[i, j] = Z_1[i, j-1] + Z_2[i, j].$$

By symmetry of Z_1 and Z_2, an iterative algorithm is derived as presented below:

1) Obtain the first row/column of Z_2 as

$$Z_2[1, j] = Z_2[j, 1] = Z_{SS}[j, 1], \quad \forall j = 1, 2, \ldots, k-1.$$

The first row/column of Z_1 is computed by

$$\begin{aligned} Z_1[1, j] &= Z_1[j, 1] \\ &= Z_{SS}[1, j+1] - Z_2[1, j+1], \\ &\qquad \forall j = 1, 2, \ldots, k-2. \\ Z_1[1, k-1] &= Z_1[k-1, 1] = Z_{SS}[1, k]. \end{aligned}$$

Set $i = 2$.

2) Obtain the ith row/column of Z_2 as

$$\begin{aligned} Z_2[i, j] &= Z_2[j, i] \\ &= Z_{SS}[j, i] - Z_1[j, i-1], \\ &\qquad \forall j = i, i+1, \ldots, k-1. \end{aligned}$$

3) Obtain the ith row/column of Z_1 as

$$\begin{aligned} Z_1[i, j] &= Z_1[j, i] \\ &= Z_{SS}[i, j+1] - Z_2[i, j+1], \\ &\qquad \forall j = i, i+1, \ldots, k-2. \\ Z_1[i, k-1] &= Z_1[k-1, i] = Z_{SS}[i, k]. \end{aligned}$$

4) If $i \leq k-2$, set $i = i+1$ and goto step 2. Otherwise, terminate the algorithm and output Z_1 and Z_2.

It can be shown that each entry of $Z_1[i, j]$ ($Z_2[i, j]$) requires only 1 addition. Thus, this algorithm requires complexity of the order $O(k^2)$.

In summary, the proposed method consists of four major stages. The first stage is to calculate (56) with the complexity of the order $O(k^2)$. The second stage is to interpolate k polynomials $\{Z_{SS}(x, \bar{x}_j)\}_{j=1}^k$. Polynomial interpolation can be performed using the matrix product $(\hat{G}_{DC}^{-1})^t \tilde{Z}_S$. In this case, $\hat{G}_{DC}$ is a $k \times k$ Vandermonde matrix with a row appended at the bottom of $\bar{G}_{DC}$. The computation of $\hat{G}_{DC}$ requires complexity of the order of $O(k^2)$ [27]. After obtaining the result $\tilde{Z}_S \hat{G}_{DC}^{-1}$, we compute a $(k-1) \times k$ matrix Z_{SS}^{tmp} by dividing the ith column, $1 \leq i \leq k$, of the result matrix by $(x - \bar{x}_j)$. This stage requires $O(k^3)$ operations in the matrix product and $O(k^2)$ operations in polynomial division on each matrix column.

The third stage is to solve the coefficients of $Z_{SS}(x, y)$. The result is computed by the matrix product $Z_{SS} = Z_{SSL}^{tmp} \hat{G}_{DC}^{-1}$ with complexity of order $O(k^3)$ operations. Finally, the last stage is to split Z_{SS} into two matrices Z_1 and Z_2 with $O(k^2)$ additions.

This method has several advantages over the decoding of (16). First, we do not need to independently solve two equations $\tilde{Z}_1 = \bar{G}_{DC}^t Z_1 \bar{G}_{DC}$ and $\tilde{Z}_2 = \bar{G}_{DC}^t Z_2 \bar{G}_{DC}$. Instead, the decoding operations are only operated on a unique matrix, and so, the computational cost and memory cost can be reduced. Second, we do not need to consider the unknown entries in the diagonals of $\tilde{Z}_1$ and $\tilde{Z}_2$. In the proposed technique, the decoding procedure needs to only store matrix $\hat{G}_{DC}^{-1}$.

Appendix C
Proofs of the Vailidity of the Encoding Matrix Conversion

The proof utilizes the following trivial proposition.

Proposition 2 [28]: Let X denote a $b \times n$ matrix and X^{TM} denote a $b \times b$ non-singular matrix, where $b \leq n$. If any b columns of X are linearly independent, any b columns of product $X^{TM} \cdot X$ are also linearly independent. □

Lemma 1: If $G_{(0)}$ *is valid, then* $G_{(1)}$ *is also valid.*

Proof: The validity of the each condition of encoding matrix is discussed separately.

- *Condition 2):* From Condition 2), $\{\lambda_i^{(0)}\}_{i=1}^n$ are mutually distinct. Clearly, $\{\lambda_i^{(1)} = \lambda_i^{(0)} - \lambda_k^{(0)}\}_{i=1}^n$ are also mutually distinct such that Condition 2) holds.
- *Condition 3):* From Condition 3), any $k-1$ columns of $\bar{G}_{(0)}$ are linearly independent. Since matrix $(\bar{G}_{(0)}^{(k-1)})^{-1}$ is non-singular, by Proposition 2, any $k-1$ columns of $\bar{G}_{(1)} = (\bar{G}_{(0)}^{(k-1)})^{-1} \bar{G}_{(0)}$ are linearly independent.
- *Condition 1):* To verify this condition, the first term of $G_{(1)}$ is reformulated as

$$\begin{aligned} &\bar{G}_{(1)} \Lambda_{(1)} \\ &\quad = (\bar{G}_{(0)}^{(k-1)})^{-1} \bar{G}_{(0)} (\Lambda_{(0)} - \mathrm{diag}(\lambda_k, \ldots, \lambda_k)) \\ &\quad = (\bar{G}_{(0)}^{(k-1)})^{-1} \bar{G}_{(0)} \Lambda_{(0)} - (\bar{G}_{(0)}^{(k-1)})^{-1} \bar{G}_{(0)} \\ &\qquad \times \mathrm{diag}(\lambda_k, \ldots, \lambda_k) \\ &\quad = (\bar{G}_{(0)}^{(k-1)})^{-1} \bar{G}_{(0)} \Lambda_{(0)} - \lambda_k (\bar{G}_{(0)}^{(k-1)})^{-1} \bar{G}_{(0)}. \end{aligned}$$

The third term of $G_{(1)}$ can be reformulated as

$$\begin{aligned} \Delta_{(1)} &= \Delta_{(0)} - \Delta_{(0)}^{(k-1)} \bar{G}_{(1)} \\ &= \Delta_{(0)} - \Delta_{(0)}^{(k-1)} (\bar{G}_{(0)}^{(k-1)})^{-1} \bar{G}_{(0)}. \end{aligned} \tag{58}$$

Substituting (58) and (58) into $G_{(1)}$, we have

$$\begin{aligned} G_{(1)} &= G_{(1)}^{TM} G_{(0)} \\ &= \begin{bmatrix} (\bar{G}_{(0)}^{(k-1)})^{-1} & -\lambda_k (\bar{G}_{(0)}^{(k-1)})^{-1} & \mathbf{0} \\ \mathbf{0} & (\bar{G}_{(0)}^{(k-1)})^{-1} & \mathbf{0} \\ \mathbf{0} & -\Delta_{(0)}^{(k-1)} (\bar{G}_{(0)}^{(k-1)})^{-1} & I \end{bmatrix} \begin{bmatrix} \bar{G}_{(0)} \Lambda_{(0)} \\ \bar{G}_{(0)} \\ \Delta_{(0)} \end{bmatrix}. \end{aligned} \tag{59}$$

Notice that $G_{(1)}^{\mathrm{TM}}$ is non-singular because of

$$\det(G_{(1)}^{\mathrm{TM}}) = \begin{vmatrix} (\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} & -\lambda_k(\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} & \mathbf{0} \\ \mathbf{0} & (\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} & \mathbf{0} \\ \mathbf{0} & -\Delta_{(0)}^{(\mathrm{k}-1)}(\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} & I \end{vmatrix}$$
$$= \begin{vmatrix} (\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} & -\lambda_k(\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} \\ \mathbf{0} & (\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} \end{vmatrix}$$
$$= \det((\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1})^2 \neq 0 \tag{60}$$

As any d columns of $G_{(0)}$ are linearly independent, by Proposition 2, any d columns of $G_{(1)} = G_{(1)}^{\mathrm{TM}} G_{(0)}$ are also linearly independent.

- *Condition 4):* Let $\delta^{(0)}$ denote the first row of $\Delta_{(0)}$, and let row vector $\delta_{(0)}^{(\mathrm{k}-1)}$ contain the first $k-1$ elements of $\delta^{(0)}$. $\hat{G}_{(1)}$ denotes the combination $\bar{G}_{(1)}$ with the first row of $\Delta_{(1)}$. By (59), $\hat{G}_{(1)}$ can be obtained from $\bar{G}_{(0)}$ as

$$\hat{G}_{(1)} = \hat{G}_{(1)}^{\mathrm{TM}} \hat{G}_{(0)} = \begin{bmatrix} (\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} & \mathbf{0} \\ -\delta_{(0)}^{(\mathrm{k}-1)}(\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1} & 1 \end{bmatrix} \begin{bmatrix} \bar{G}_{(0)} \\ \delta^{(0)} \end{bmatrix}. \tag{61}$$

Notice that $\hat{G}_{(1)}^{\mathrm{TM}}$ is non-singular because of

$$\det(\hat{G}_{(1)}^{\mathrm{TM}}) = \det((\bar{G}_{(0)}^{(\mathrm{k}-1)})^{-1}) \neq 0.$$

As any k columns of $\hat{G}_{(0)}$ are linearly independent, by Proposition 2, any k columns of $\hat{G}_{(1)} = \hat{G}_{(1)}^{\mathrm{TM}} \hat{G}_{(0)}$ are also linearly independent. □

Lemma 2: If $G_{(1)}$ *is valid, then* G_{SC} *is also valid.*

Proof: Since $G_{(1)}$ is valid, Conditions 2) and 3) are satisfied inherently. Hence, we only consider the validity for Conditions 1) and 4).

- *Condition 1):* G_{SC} can be reformulated as

$$G_{\mathrm{SC}} = G_{\mathrm{SC}}^{\mathrm{TM}} G_{(1)} = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Delta_{(2)}^{\mathrm{TM}} \end{bmatrix} \begin{bmatrix} \bar{G}_{(1)}\Lambda_{(1)} \\ \bar{G}_{(1)} \\ \Delta_{(1)} \end{bmatrix}. \tag{62}$$

As

$$\det(G_{\mathrm{SC}}^{\mathrm{TM}}) = \det(\Delta_{(2)}^{\mathrm{TM}}) = 1/\delta_k[1] \neq 0,$$

$G_{\mathrm{SC}}^{\mathrm{TM}}$ is non-singular. Since any d columns of $G_{(1)}$ are linearly independent, by Proposition 2, any d columns of $G_{\mathrm{SC}} = G_{\mathrm{SC}}^{\mathrm{TM}} G_{(1)}$ are also linearly independent.

- *Condition 4):* By (39), the first row of $\Delta_{(2)}$ is the first row of $\Delta_{(1)}$ through scaling of a non-zero factor $1/\delta_k[1]$. Thus, the transformation is given by

$$\hat{G}_{\mathrm{SC}} = \hat{G}_{\mathrm{SC}}^{\mathrm{TM}} \hat{G}_{(1)} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & 1/\delta_k[1] \end{bmatrix} \hat{G}_{(1)}.$$

Clearly, $\hat{G}_{\mathrm{SC}}^{\mathrm{TM}}$ is non-singular; then, Condition 4) holds. □

Theorem 1: The number of non-zero entries of G_{SN} *attains the lower bound, under the conditions of the encoding matrix.*

Proof: By its construction, G_{SN} can be viewed as the encoding matrix of an $[n = k]$ Exact-MSR code, under the conditions of an encoding matrix. To demonstrate the optimality, we first prove that the $[n = k]$ Exact-MSR code includes at least $3k-2$ non-zero entries in the encoding matrix, and then show that G_{SN} attaches this bound.

As shown in (19), the encoding matrix of the $[n = k]$ code consists of three parts, termed $\bar{G}_{[n=k]}\Lambda_{[n=k]}$, $\bar{G}_{[n=k]}$, and $\Delta_{[n=k]}$. In the following, we analyze the minimum number of non-zero entries in each part, separately.

- $\Delta_{[n=k]}$ includes at least one non-zero entry:
  This claim can be easily proved by contradiction: if $\Delta_{[n=k]}$ is a zero matrix, $\hat{G}_{[n=k]}$ will have a zero row in the bottom; thus, the encoding matrix cannot satisfy Condition 4).
- $\bar{G}_{[n=k]}$ includes at least $2k-2$ non-zero entries:
  By Condition 3), $\bar{G}_{[n=k]}$ can be viewed as the encoding matrix of $(k, k-1)$ MDS code with minimum Hamming distance 2, and thus the number of non-zero entries is at least $2(k-1)$.
- $\bar{G}_{[n=k]}\Lambda_{[n=k]}$ includes at least $k-1$ non-zero entries:
  This claim can be proved by contradiction. Assume that $\bar{G}_{[n=k]}\Lambda_{[n=k]}$ includes less than $k-1$ non-zero entries. Then there exists at least two zero columns in $\bar{G}_{[n=k]}\Lambda_{[n=k]}$ that are denoted as $\bar{g}_i\lambda_i$ and $\bar{g}_j\lambda_j$. Since $\lambda_i \neq \lambda_j$ by Condition 2), there exists at least one non-zero element between λ_i and λ_j. W.L.O.G, assume that λ_i is the non-zero entry. Since $\bar{g}_i\lambda_i = 0$, $\bar{g}_i$ should be a zero column such that Condition 3) cannot be satisfied. This completes this claim.

After all three numbers of the non-zero entries have been added up, the encoding matrix of the $[n = k]$ Exact-MSR code has at least $1 + 2k - 2 + (k-1) = 3k-2$ non-zero entries, which is equivalent to the number of non-zero entries of G_{SN}. Hence, G_{SN} is with the minimum number of non-zero entries. □

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. 19th ACM SIGOPS Symp. Oper. Syst. Principles*, Bolton Landing, NY, USA, Oct. 2003, pp. 29–43.

[2] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960.

[3] A. Fikes, "Storage architecture and challenges," in *Proc. Google Faculty Summit*, 2010.

[4] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Anchorage, AK, USA, May 2007, pp. 2000–2008.

[5] 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.

[6] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. 45th Annu. Allerton Conf. Control, Comput., Commun.*, Urbana, IL, USA, Sep. 2007, pp. 242–249.

[7] 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.

[8] V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li, "Optimal repair of MDS codes in distributed storage via subspace interference alignment," *CoRR*, vol. abs/1106.1250, 2011.

[9] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, "Repair optimal erasure codes through Hadamard designs," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3021–3037, May 2013.
[10] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1837–1852, Mar. 2012.
[11] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. 47th Annu. Allerton Conf. Control, Comput., Commun.*, Urbana, IL, USA, Sep./Oct. 2009, pp. 1243–1249.
[12] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 277–288, Feb. 2010.
[13] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6734–6753, Oct. 2011.
[14] F. Oggier and A. Datta, "Byzantine fault tolerance of regenerating codes," in *Proc. IEEE Int. Conf. Peer-to-Peer Comput. (P2P)*, Kyoto, Japan, 2011, pp.112–121.
[15] Y. S. Han, R. Zheng, and W. H. Mow, "Exact regenerating codes for Byzantine fault tolerance in distributed storage," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Orlando, FL, USA, Mar. 2012, pp. 2498–2506.
[16] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Seoul, Korea, Jun./Jul. 2009, pp. 2276–2280.
[17] D. F. Cullina, "Searching for minimum storage regenerating codes," M.S. thesis, Dept. Elect. Eng., California Inst. Technol., Pasadena, CA, USA, 2009.
[18] 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.
[19] S. Goparaju, I. Tamo, and R. Calderbank, "An improved sub-packetization bound for minimum storage regenerating codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2770–2779, May 2014.
[20] S. Goparaju and R. Calderbank, "A new sub-packetization bound for minimum storage regenerating codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1616–1620.
[21] C. Tian, V. Aggarwal, and V. A. Vaishampayan, "Exact-repair regenerating codes via layered erasure correction and block designs," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1431–1435.
[22] T. Ernvall, "Exact-regenerating codes between MBR and MSR points," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Sep. 2013, pp. 1–5.
[23] B. Sasidharan and P. V. Kumar, "High-rate regenerating codes through layering," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1611–1615.
[24] J. Li and B. Li, "Cooperative repair with minimum-storage regenerating codes for distributed storage," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr./May 2014, pp. 316–324.
[25] Y. S. Han, H.-T. Pai, R. Zheng, and P. K. Varshney, "Update-efficient regenerating codes with minimum per-node storage," *CoRR*, vol. abs/1301.2497, 2013.
[26] Y. S. Han, H.-T. Pai, R. Zheng, and P. K. Varshney, "Update-efficient regenerating codes with minimum per-node storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1436–1440.
[27] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1988.
[28] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra*, 4th ed. London, U.K.: Pearson, 2002.

Sian-Jheng Lin received the B.Sc., M.Sc., and Ph.D. degrees in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2004, 2006, and 2010, respectively. From 2010 to 2014, he was a postdoc at the Research Center for Information Technology Innovation, Academia Sinica. He was a part-time lecturer at Yuanpei University from 2007 to 2008, and at Hsuan Chuang University From 2008 to 2010. He is currently a postdoc with the Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia. His recent research interests include erasure codes and distributed storage codes.

Wei-Ho Chung (M'10) received the B.Sc. and M.Sc. degrees in Electrical Engineering from the National Taiwan University, Taipei, Taiwan, in 2000 and 2002, respectively, and the Ph.D. degree in Electrical Engineering from the University of California, Los Angeles, in 2009. From 2002 to 2005, he was a system engineer at ChungHwa Telecommunications Company, where he worked on data networks. In 2008, he worked on CDMA systems at Qualcomm, Inc., San Diego, CA. His research interests include communications, signal processing, and networks. Dr. Chung received the Taiwan Merit Scholarship from 2005 to 2009 and the Best Paper Award in IEEE WCNC 2012, and has published over 40 journal articles and over 50 conference papers. Since January 2010, Dr. Chung has been an assistant research fellow, and promoted to the rank of associate research fellow in January 2014 in Academia Sinica. He leads the Wireless Communications Lab in the Research Center for Information Technology Innovation, Academia Sinica, Taiwan.

Yunghsiang S. Han (S'90–M'93–SM'08–F'11) was born in Taipei, Taiwan, 1962. He received B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and a Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, in 1993.

He was from 1986 to 1988 a lecturer at Ming-Hsin Engineering College, Hsinchu, Taiwan. He was a teaching assistant from 1989 to 1992, and a research associate in the School of Computer and Information Science, Syracuse University from 1992 to 1993. He was, from 1993 to 1997, an Associate Professor in the Department of Electronic Engineering at Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering at National Chi Nan University, Nantou, Taiwan from 1997 to 2004. He was promoted to Professor in 1998. He was a visiting scholar in the Department of Electrical Engineering at University of Hawaii at Manoa, HI from June to October 2001, the SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and CASE center at Syracuse University, NY from September 2002 to January 2004 and July 2012 to June 2013, and the visiting scholar in the Department of Electrical and Computer Engineering at University of Texas at Austin, TX from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering at National Taipei University, Taipei, Taiwan from August 2004 to July 2010. From August 2010, he is with the Department of Electrical Engineering at National Taiwan University of Science and Technology as Chair professor. His research interests are in error-control coding, wireless networks, and security.

Dr. Han was a winner of the 1994 Syracuse University Doctoral Prize and a Fellow of IEEE.

Tareq Y. Al-Naffouri (M'10) received the B.S. degrees in mathematics and electrical engineering (with first honors) from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, in 1994, the M.S. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, in 1998, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2004.

He was a visiting scholar at California Institute of Technology, Pasadena, CA, from January to August 2005 and during summer 2006. He was a Fulbright scholar at the University of Southern California from February to September 2008. He has held internship positions at NEC Research Labs, Tokyo, Japan, in 1998, Adaptive Systems Lab, University of California at Los Angeles in 1999, National Semiconductor, Santa Clara, CA, in 2001 and 2002, and Beceem Communications Santa Clara, CA, in 2004. He is currently an Associate Professor at the Electrical Engineering Department, King Fahd University of Petroleum and Minerals, Saudi Arabia, and jointly at the Electrical Engineering Department, King Abdullah University of Science and Technology (KAUST). His research interests lie in the areas of adaptive and statistical signal processing and their applications to wireless communications, seismic signal processing, and in multiuser information theory. He has recently been interested in compressive sensing and random matrix theory and their applications. He has over 80 publications in Journal and conference proceedings, 9 standard contributions, 4 issued patents, and 4 pending.

Dr. Al-Naffouri is the recipient of a 2001 Best Student Paper Award at the IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP) 2001 for his work on adaptive filtering analysis, the IEEE Education Society Chapter Achievement Award in 2008, and Al-Marai Award for innovative research in communication in 2009. Dr. Al-Naffouri has also been serving as an Associate Editor of TRANSACTIONS ON SIGNAL PROCESSING since August 2013.