An Unified Form of Exact-MSR Codes via Product-Matrix Framework

Sian-Jheng Lin and Wei-Ho Chung*

***Abstract*—Regenerating code represents a class of erasure codes applicable to distributed storage systems. An $[n,k,d]$ regenerating code admits the recovery of the message from any k out of the n encoded fragments, and the recovery of an erasure fragment from other arbitrary d valid fragments, for $k \leq d \leq n-1$. Minimum Storage Regenerating (MSR) code is the regenerating code attaining the minimal storage requirement. By product-matrix framework, the $[n,k,d \geq 2k-2]$ exact-MSR code is presented by Rashmi et al. This paper presents a novel version of exact-MSR codes for the same set of parameters. As compared with previous works, the major contributions of work include: i). An approach is proposed to directly include the $d > 2k-2$ Exact-MSR codes in the product-matrix representation; ii). The required size of the finite field is reduced to n from $n(d-k+1)$. The proposed coding techniques further improve the practicability of Exact-MSR codes.**

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

I. Introduction

Erasure code is useful for the design of distributed storage systems to offer data reliability and storage efficiency. In the distributed storage system, the source data (message) is encoded into many code fragments via erasure coding techniques, and those code fragments are distributed among the data servers (nodes) in the network. A client (data collector) can obtain the source data by downloading a certain amount of code fragments from a subset of accessible nodes in the network. A well-known erasure code for distributed storage systems is the Reed-Solomon (RS) codes [1] belonging to the maximum distance separable (MDS) codes. When the distributed storage system is built on an unstable network environment, the nodes may frequently join and depart. When a node departs or crashes, the system manager will place a new blank node in the storage network to replace the functionality of the failed node. The replacement node connects to a subset of other active nodes, and then downloads the requisite information to restore the lost code fragment. A directly way is to reconstruct the entire source data and then convert the data to obtain the required fragment. However, as the size of data stored in a single node is much smaller than the entire source data, it is possible to design a new class of storage codes to reduce the amount of downloaded symbols in node-repairing process. The new class of storage codes, termed as regenerating codes, is introduced by the pioneer paper [3].

This work was supported by National Science Council of Taiwan, under grant number NSC101-2221-E-001-002, NSC101-2221-E-001-008, NSC101-2622-E-001-001-CC3, and 32T-1010721-1C.

Authors are with the Research Center for Information Technology Innovation, Academia Sinica, Taipei City, Taiwan. (e-mail: sjlin@citi.sinica.edu.tw;whc@citi.sinica.edu.tw)

Regenerating code represents a category of distributed storage codes with design criteria including joint consideration of the bandwidth requirement for node repairing and the size of each fragment stored in a node. In an $[n,k,d]$ regenerating code, the B message symbols are encoded as n fragments with fragment size as α symbols, and those fragments are respectively stored in n different nodes in the network. A client (data collector) can recover the entire message by downloading k fragments in nodes. Upon failure of a node, the fragment stored in the failed node can be reconstructed through obtaining β symbols per node from arbitrary d non-failed nodes. In a well designed regenerating code, the data amount for repairing a failed node βd is much smaller than the size of the message B. In summary, a regenerating code over $GF(q)$ is associated with those parameters

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

By the problem setup, the parameters follows the inequality

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

A storage-bandwidth bound for feasible regenerating codes has been proven in [4] by the cut-set bound of network coding:

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

Two extreme cases have been adequately investigated. The minimum bandwidth regeneration (MBR) is an extreme case where the interest of investigation is related to minimizing β:

$$\begin{aligned}&\beta = 2B/\left(k(2d-k+1)\right);\\&\alpha = d\beta.\end{aligned}$$

Another extreme case is the minimum storage regenerating (MSR) code where the interest of investigation is related to minimizing the α:

$$\begin{aligned}&\alpha = B/k;\\&\beta = B/\left(k(d-k+1)\right).\end{aligned}$$

Exact regeneration is an additional property where the reconstructed fragment is exactly the same with the original fragment stored in the failed node. This is a highly desired property in the practical implementation. However, the effective and systematic approach to construct the exact-regeneration codes at the interior points on storage-bandwidth bound curve[5] is still under searching, except at the MSR and MBR points. A number of practical exact regenerating codes have been proposed in recent years. For example, Wu and Dimakis [6] present the Exact-MSR code for $n = d+1$ and $k = 2$.

Cullina et al. [7] discover the practical Exact-MSR code at $[n = 4, k = 2, d = 3]$ and $[n = 5, k = 3, d = 4]$ via computer searching. Shah et al. [5] present the $[n = d+1, k, d]$ Exact-MBR codes with the repair-by-transfer, which is a property where the node-repairing process does not involve any computations. A remarkable work for the construction of exact regenerating codes is presented by Rashmi et al. [2]. By product-matrix form, the [2] presents 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 $[n, k, d \geq 2k-2]$. Note that the construction of Exact-MSR codes for $d < 2k-3$ is currently non-achievable [8] for $\beta = 1$.

The Exact-MBR code presented in [2] is efficient and easily realizable. In contrast, we observe that the Exact-MSR code presented in [2] can be further improved from the implementation perspective. This paper presents a new version of Exact-MSR codes under the same set of parameters $[n, k, d \geq 2k-2]$. The proposed Exact-MSR codes is built upon [2]. However, due to the length limitation, we cannot include the summary of [2]. As compared with prior Exact-MSR codes [2], the advantages of proposed Exact-MSR codes are summarized in the following. i). This paper presents an unified representation for $d \geq 2k-2$ Exact-MSR codes under product-matrix form. In [2], the non-systematic version of Exact-MSR codes at $d = 2k-2$ is firstly presented, and then the non-systematic version is converted to a systematic version through a process of message-symbol remapping. Finally, the shortening technique is applied on the systematic Exact-MSR code to construct the codes for $d > 2k-2$. In contrast, this paper presents a way of representing $d \geq 2k-2$ Exact-MSR codes directly in the product-matrix form to avoid the shortening technique in the constructions of Exact-MSR codes. Thus, the proposed Exact-MSR codes require lower computational overhead than [2]. ii). This paper gives a form of encoding matrix to reduce the size of required finite field to n from $n(d-k+1)$ or higher. The details are addressed in Section V. As the smaller finite field size can simplify the design of field arithmetic, the proposed form of encoding matrix is useful for practical applications.

The rest of this paper is organized as follows. Section II presents the proposed Exact-MSR code by product-matrix framework. Section III and Section IV respectively present the node-repairing process and data reconstruction process on the proposed Exact-MSR codes. Section V compares the [2] with the proposed codes in encoding process. Section VI concludes this work.

II. Product-matrix Exact-MSR code

This section presents the product-matrix form for $[n, k, d \geq 2k-2]$ Exact-MSR codes. Throughout this paper, the operations and symbols are drawn from the finite field $GF(q)$. By the data striping technique [2], the optimal regenerating codes for any values of β can be constructed from the optimal regenerating codes with $\beta = 1$. Thus, this paper focuses on the Exact-MSR codes at $\beta = 1$, and the corresponding parameters are

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

In encoding, the B message symbols are encoded into n row vectors, and each row has α elements. Those n vectors are respectively stored in n nodes. Let a $n \times \alpha$ matrix C denote the collection of those n row vectors. The ith row of C, denoted as c_i^t for $i = 1 \ldots n$, is stored in the corresponding node with index i. It is noted that the superscript $'t'$ denotes the transpose of a matrix. The codeword matrix is calculated by the matrix product

$$C = \Psi M \tag{3}$$

of a $n \times d$ encoding matrix Ψ and a $d \times \alpha$ message matrix M. Let ψ_i^t denote the ith row of Ψ. The (3) can be rewritten as the form of a vector c_i^t in C:

$$c_i^t = \psi_i^t M. \tag{4}$$

The structure of encoding matrix Ψ and message matrix M is organized by a specific form, and the form of Ψ should follow a number of conditions. The structure and conditions of Ψ and M are elaborated as follows.

The message matrix is constructed from four component matrices S_1, S_2, T and Z. Both S_1 and S_2 are $(k-1)\times(k-1)$ symmetric matrices, and each matrix is determined by $\binom{k}{2}$ message symbols. Precisely, the entries in the upper triangular part of each of S_1 and S_2 are filled with message symbols, and the strictly lower triangle parts of both matrices fill the corresponding values such that the symmetric property holds. The matrix T is a $(k-1) \times (d-2k+2)$ matrix filled with $(k-1) \times (d-2k+2)$ distinct message symbols. The Z is defined as a $(d-2k+2) \times (d-2k+2)$ symmetric matrix, where the entries in non-first-row and non-first-column are all filled with zeros. Precisely, the form of Z is expressed as

$$Z = \begin{bmatrix} z_0 & z_1^t \\ z_1 & \mathbf{0} \end{bmatrix},$$

where the $\mathbf{0}$ denotes a $(d-2k+1)\times(d-2k+1)$ zero matrix, the z_0 is a scalar value, and the z_1^t is a $(d-2k+1)$-element row vector. By the above notations, the layout of $d\times(d-k+1)$ message matrix M is defined as

$$M = \begin{bmatrix} S_1 & \mathbf{0} \\ S_2 & T \\ T^t & Z \end{bmatrix}, \tag{5}$$

where the $\mathbf{0}$ is a $(k-1)\times(d-2k+2)$ zero matrix. Thus, the M contains a total of

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

source symbols, which meets the setup given in (2).

For the encoding matrix Ψ, the $n \times d$ encoding matrix consists of several component matrices given by

$$\Psi = \begin{bmatrix} \Lambda\Phi & \Phi & \Delta \end{bmatrix}, \tag{6}$$

where the Φ denotes a $n \times (k-1)$ matrix, the Δ denotes a $n\times(d-2k+2)$ matrix, and the Λ is a $n\times n$ diagonal matrix. Additionally, for the case $d > 2k-2$, we define a $n \times k$ matrix $\hat{\Phi}$ which is the concatenation Φ with the first column of Δ. The encoding matrix is chosen to satisfy the following

conditions:
i). Any d rows of Ψ are linearly independent;
ii). The n diagonal elements of Λ are mutually distinct;
iii). Any $k-1$ rows of Φ are linearly independent;
iv). For $d > 2k-2$, any k rows of $\hat{\Phi}$ are linearly independent.
For $d = 2k-2$, the Ψ and M are expressed as

$$M = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}, \Psi = \begin{bmatrix} \Lambda\Phi & \Phi \end{bmatrix}, \tag{7}$$

which is isomorphic to the Exact-MSR framework designed by the [2]. Thus, the proposed framework can be treated as a generalized form of [2]. As the additional matrix $\hat{\Phi}$ cannot be properly defined at this case $d = 2k-2$, the corresponding encoding matrix only demands the (i)-(iii) conditions.

In encoding, we should choose a practical form of encoding matrix to satisfy the above four conditions. A feasible version of encoding matrix is the Vandermonde matrix with shuffling the columns by a specific order. In this case, given $n \leq q$ distinct elements $\{x_1, x_2 \dots x_n\}$ taken from $GF(q)$, the diagonal matrix is defined as $\Lambda = diag(x_1, x_2 \dots x_n)$ to meet the second condition. The Φ is chosen as a Vandermonde matrix, where the ith row of Φ is defined as $\phi_i^t = \begin{bmatrix} 1 & x_i^2 & x_i^4 & \dots & x_i^{2(k-2)} \end{bmatrix}$. By the form of Vandermonde matrix, the n elements $\{x_1^2, x_2^2 \dots x_n^2\}$ need to be mutually distinct, and this condition can be satisfied with assigning the field size q to a power of two. Thus, the third condition is certainly satisfied. For the form of Δ, the ith row of Δ is defined as $\begin{bmatrix} x_i^{2k-2} & x_i^{2k-1} & x_i^{2k} & \dots & x_i^{d-1} \end{bmatrix}$. By taking the left element x_i^{2k-2} from Δ, the ith row of $\hat{\Phi}$ is expressed as $\begin{bmatrix} 1 & x_i^2 & \dots & x_i^{2k-4} & x_i^{2k-2} \end{bmatrix}$, which meets the form of Vandermonde matrix. Therefore the fourth condition is satisfied. Finally, we concatenate those matrices to obtain the Ψ. The ith row of Ψ is expressed as

$$\psi_i^t = [x_i \dots x_i^{2k-3} | 1 \dots x_i^{2k-4} | x_i^{2k-2} \dots x_i^{d-1}].$$

Those columns of Ψ can be sorted by the exponents of x_i, resulting in a geometric progression in each row $\begin{bmatrix} 1 & x_i & x_i^2 & \dots & x_i^{d-1} \end{bmatrix}$. Thus, the Ψ is the Vandermonde matrix with shuffling the columns, and the first condition holds. By such form of encoding matrix, the smallest size of applied finite field is $q = 2^{\lceil \lg n \rceil}$.

Example: We give an example of the construction of Exact-MSR code over $GF(8)$. The sample case is at $(n, k, d) = (8, 3, 6)$, and $(\alpha, \beta, B) = (4, 1, 12)$ by the setup of Exact-MSR code. The matrices S_1, S_2, T and Z are filled with 12 source symbols $\{u_i\}_{i=1}^{12}$ as follows:

$$S_1 = \begin{bmatrix} u_1 & u_2 \\ u_2 & u_3 \end{bmatrix}, S_2 = \begin{bmatrix} u_4 & u_5 \\ u_5 & u_6 \end{bmatrix},$$
$$T = \begin{bmatrix} u_7 & u_8 \\ u_9 & u_{10} \end{bmatrix}, Z = \begin{bmatrix} u_{11} & u_{12} \\ u_{12} & 0 \end{bmatrix}.$$

By combining those matrices, the message matrix M is given by

$$M = \begin{bmatrix} u_1 & u_2 & 0 & 0 \\ u_2 & u_3 & 0 & 0 \\ u_4 & u_5 & u_7 & u_8 \\ u_5 & u_6 & u_9 & u_{10} \\ u_7 & u_9 & u_{11} & u_{12} \\ u_8 & u_{10} & u_{12} & 0 \end{bmatrix}.$$

For the encoding matrix, each entry of Ψ is represented as the 0, 1 or a power of the primitive element α of field $GF(8)$. The size of Φ is 8×2; the ith row, $1 \leq i \leq 7$, of Φ is represented as $\begin{bmatrix} 1 & \alpha^{2(i-1)} \end{bmatrix}$; and the last row is $\begin{bmatrix} 1 & 0 \end{bmatrix}$. The Λ is a 8×8 diagonal matrix $\Lambda = diag(1, \alpha, \alpha^2 \dots \alpha^6, 0)$. The size of Δ is 8×2, and the ith row, $1 \leq i \leq 7$, of Δ is represented as $\begin{bmatrix} \alpha^{4(i-1)} & \alpha^{5(i-1)} \end{bmatrix}$, and the last row is $\begin{bmatrix} 0 & 0 \end{bmatrix}$. Thus, the encoding matrix is represented as

$$\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \alpha & \alpha^3 & 1 & \alpha^2 & \alpha^4 & \alpha^5 \\ \alpha^2 & \alpha^6 & 1 & \alpha^4 & \alpha^1 & \alpha^3 \\ \alpha^3 & \alpha^9 & 1 & \alpha^6 & \alpha^5 & \alpha^1 \\ \alpha^4 & \alpha^5 & 1 & \alpha^1 & \alpha^2 & \alpha^6 \\ \alpha^5 & \alpha^1 & 1 & \alpha^3 & \alpha^6 & \alpha^4 \\ \alpha^6 & \alpha^4 & 1 & \alpha^5 & \alpha^3 & \alpha^2 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

III. Node-repairing process

Suppose the node h_0 crashes, and a replacement node is placed into the network to replace the functionality of the failed node. The replacement node can exactly restore the node data $c_{h_0}^t$ through downloading a symbol from arbitrary d non-failed nodes (helper nodes) $\{h_1, h_2 \dots h_d\}$. We also note that the node h_j corresponds to a row $c_{h_j}^t$ in C and corresponds to a row $\psi_{h_j}^t$ in Ψ. By construction, the encoding row of failed node $\psi_{h_0}^t$ is represented as $\psi_{h_0}^t = \begin{bmatrix} \lambda_{h_0}\phi_{h_0}^t & \phi_{h_0}^t & \delta_{h_0}^t \end{bmatrix}$. Thus, the $d-k+1$ symbols stored in a failed node is

$$\begin{aligned} c_{h_0}^t &= \begin{bmatrix} \lambda_{h_0}\phi_{h_0}^t & \phi_{h_0}^t & \delta_{h_0}^t \end{bmatrix} M \\ &= \begin{bmatrix} \lambda_{h_0}\phi_{h_0}^t S_1 + \phi_{h_0}^t S_2 + \delta_{h_0}^t T^t & \phi_{h_0}^t T + \delta_{h_0}^t Z \end{bmatrix}. \end{aligned} \tag{8}$$

Upon transferring a symbol to the replacement node, each helper node h_j requires a α-element column vector

$$\mu_{h_0} = \begin{bmatrix} \phi_{h_0} \\ \delta_{h_0} \end{bmatrix}, \tag{9}$$

which is a sub-vector of the transpose of $\psi_{h_0}^t$. Then each helper node h_j passes a scalar value computed via

$$\upsilon_{h_j} = c_{h_j}^t \mu_{h_0}, \tag{10}$$

on to the replacement node. Thus the replacement node gathers d downloading symbols expressed as a d-element column vector $\Upsilon_{\text{repair}} = \begin{bmatrix} \upsilon_{h_1} & \upsilon_{h_2} \dots \upsilon_{h_d} \end{bmatrix}^t$. Let $\Psi_{\text{repair}} = \begin{bmatrix} \psi_{h_1} & \psi_{h_2} \dots \psi_{h_d} \end{bmatrix}^t$ denote a sub-matrix of Ψ corresponding to the encoding rows of d helper nodes. By definition, the Υ_{repair} possesses the equality

$$\begin{aligned}\Upsilon_{\text{repair}} &= \Psi_{\text{repair}} M \mu_{h_0} \\ &= \Psi_{\text{repair}} \begin{bmatrix} S_1 & \mathbf{0} \\ S_2 & T \\ T^t & Z \end{bmatrix} \begin{bmatrix} \phi_{h_0} \\ \delta_{h_0} \end{bmatrix} \\ &= \Psi_{\text{repair}} \begin{bmatrix} S_1\phi_{h_0} \\ S_2\phi_{h_0} + T\delta_{h_0} \\ T^t\phi_{h_0} + Z\delta_{h_0} \end{bmatrix}.\end{aligned} \tag{11}$$

By the first condition of the encoding matrix, the Ψ_{repair} is non-singular, so the Υ_{repair} can be multiplied by the inverse of Ψ_{repair} to decode the three terms

$$\Psi_{\text{repair}}^{-1} \Upsilon_{\text{repair}} = \begin{bmatrix} S_1\phi_{h_0} \\ S_2\phi_{h_0} + T\delta_{h_0} \\ T^t\phi_{h_0} + Z\delta_{h_0} \end{bmatrix}.$$

By the above solved terms, the first part of (8) is computed by

$$\begin{aligned}&(\lambda_{h_0}(S_1\phi_{h_0}) + (S_2\phi_{h_0} + T\delta_{h_0}))^t \\ =&\lambda_{h_0}\phi_{h_0}^t S_1 + \phi_{h_0}^t S_2 + \delta_{h_0}^t T^t,\end{aligned}$$

and the second part of (8) is the transpose of the term $T^t\phi_{h_0} + Z\delta_{h_0}$.

IV. Data reconstruction process

In the data reconstruction, the client can retrieve the whole message via downloading the data stored in arbitrary k active nodes. Specifically, the B message symbols can be recovered from arbitrary k rows of the codeword matrix C. Suppose the client connects to k active nodes corresponding to the rows of C at $\{i_1, i_2 \dots i_k\}$. Each connected node passes the vector $c_{i_j}^t$ on to the client, so that the client accesses k vectors $\{c_{i_1}^t, c_{i_2}^t \dots c_{i_k}^t\}$, and those vectors form as a $k \times \alpha$ matrix C_{DC} whose ith row is the $c_{i_j}^t$. Besides, the corresponding k encoding rows in the Ψ is denoted as a $k \times d$ matrix Ψ_{DC}. Then we have

$$C_{\text{DC}} = \Psi_{\text{DC}} M. \tag{12}$$

By the definition of the encoding matrix, the $k \times d$ sub-matrix Ψ_{DC} can be represented as

$$\Psi_{\text{DC}} = [\ \Lambda_{\text{DC}}\Phi_{\text{DC}} \quad \Phi_{\text{DC}} \quad \Delta_{\text{DC}}\ ]. \tag{13}$$

Then the (12) is reformulated as

$$C_{\text{DC}} = [\ \Lambda_{\text{DC}}\Phi_{\text{DC}} \quad \Phi_{\text{DC}} \quad \Delta_{\text{DC}}\ ] \begin{bmatrix} S_1 & \mathbf{0} \\ S_2 & T \\ T^t & Z \end{bmatrix}.$$

To explicitly express the equality, the C_{DC} is split into two sub-matrices $C_{\text{DC}} = [\ C_{\text{DC}}^{\text{L}} \quad C_{\text{DC}}^{\text{R}}\ ]$, where the left part C_{DC}^{L} consists of $k-1$ columns referring to the left part of M, and the right part C_{DC}^{R} consists of $d-2k+2$ columns referring to the right part of M. Precisely,

$$C_{\text{DC}}^{\text{L}} = [\ \Lambda_{\text{DC}}\Phi_{\text{DC}} \quad \Phi_{\text{DC}} \quad \Delta_{\text{DC}}\ ] \begin{bmatrix} S_1 \\ S_2 \\ T^t \end{bmatrix}; \tag{14}$$

$$C_{\text{DC}}^{\text{R}} = [\ \Phi_{\text{DC}} \quad \Delta_{\text{DC}}\ ] \begin{bmatrix} T \\ Z \end{bmatrix}. \tag{15}$$

The decoding process is firstly applied on the part C_{DC}^{R} to solve T and Z. Then the extracted T is utilized on the decoding of the C_{DC}^{L} to solve S_1 and S_2. It is noted that the case $d = 2k-2$ does not have the part C_{DC}^{R}, so the process omits the decoding stage on C_{DC}^{R}.

A. The decoding process on C_{DC}^{R}

To explicitly express the decoding process, the (15) is relabeled as

$$C_{\text{DC}}^{\text{R}} = [\ \hat{\Phi}_{\text{DC}} \quad \check{\Delta}_{\text{DC}}\ ] \begin{bmatrix} \hat{T} \\ \check{Z} \end{bmatrix},$$

where the $\hat{\Phi}_{\text{DC}}$ is the concatenation of Φ_{DC} with the first column of Δ_{DC}, and the $\check{\Delta}_{\text{DC}}$ denotes all the non-first columns of Δ_{DC}; the first column of Δ_{DC} is shifted to the last row of Φ_{DC}. Oppositely, the $\hat{T}$ denotes the concatenation of T with the first row of Z, and the $\check{Z}$ denotes all the non-first rows of the Z. By definition, the $\check{Z}$ is formed as $\check{Z} = [\ z_1 \quad \mathbf{0}\ ]$, and the transpose of z_1 appeared at the last row of $\hat{T}$. Let $\hat{t}_i$ denote the ith column of $\hat{T}$. As all the non-first columns of $\check{Z}$ are zero columns, the ith column, $i \neq 1$, of C_{DC}^{R} refers to

$$\hat{\Phi}_{\text{DC}}\hat{t}_i.$$

By the fourth condition of encoding matrix, the $\hat{\Phi}_{\text{DC}}$ is non-singular. Thus, the client can solve all the non-first columns of $\hat{T}$. In the decoding of the first column $\hat{t}_1$, the first column of C_{DC}^{R} refers to

$$\hat{\Phi}_{\text{DC}}\hat{t}_1 + \check{\Delta}_{\text{DC}} z_1.$$

The elements of z_1 is taken from the last row of the solved part of $\hat{T}$, so the term $\check{\Delta}_{\text{DC}} z_1$ is computable. Then the $\hat{t}_1$ can also be uniquely solved.

B. The decoding process on C_{DC}^{L}

Subsequently, for the decoding of S_1 and S_2, the (14) is reformulated as

$$C_{\text{DC}}^{\text{L}} - \Delta_{\text{DC}} T^t = \Lambda_{\text{DC}}\Phi_{\text{DC}} S_1 + \Phi_{\text{DC}} S_2,$$

where the left-hand side is a $k \times (k-1)$ matrix which can be directly computed. It is is noted that the case $d = 2k-2$ does not have the term $\Delta_{\text{DC}} T^t$, so the left-hand side is C_{DC}^{L} in this case. A decoding method is provided in [2], and the steps are restated below for the sake of completeness.

The $k \times (k-1)$ matrix $C_{\text{DC}}^{\text{L}} - \Delta_{\text{DC}} T^t$ is post-multiplied by the $(k-1) \times k$ matrix Φ_{DC}^t to obtain a $k \times k$ matrix

$$(C_{\text{DC}}^{\text{L}} - \Delta_{\text{DC}} T^t)\Phi_{\text{DC}}^t = \Lambda_{\text{DC}}\Phi_{\text{DC}} S_1 \Phi_{\text{DC}}^t + \Phi_{\text{DC}} S_2 \Phi_{\text{DC}}^t. \tag{16}$$

The notations in (16) are relabeled with the new notations given by

$$P = (C_{\text{DC}}^{\text{L}} - \Delta_{\text{DC}} T^t)\Phi_{\text{DC}}^t;$$

$$\tilde{S}_1 = \Phi_{\text{DC}} S_1 \Phi_{\text{DC}}^t; \tilde{S}_2 = \Phi_{\text{DC}} S_2 \Phi_{\text{DC}}^t. \tag{17}$$

It is noted that the matrices $\tilde{S}_1$ and $\tilde{S}_2$ are symmetric. Then the relabeled form of (16) is expressed as

$$P = \Lambda_{\text{DC}}\tilde{S}_1 + \tilde{S}_2.$$

TABLE I
COMPARISONS BETWEEN THE [2] AND THE PROPOSED APPROACH IN ENCODING

	Rashmi et al. [2]	Ours
Finite field size	$(n+d-2k+2)(d-k+1)$	n
Encoding matrix	$n \times (2d-2k+2)$	$n \times d$
Message matrix	$(2d-2k+2) \times (d-k+1)$	$d \times (d-k+1)$

For $1 \leq i, j \leq k$, the equations for the two elements at (i, j) and (j, i), of the matrices $\tilde{S}_1$ and $\tilde{S}_2$ are expressed as

$$P[i,j] = \lambda_i \tilde{S}_1[i,j] + \tilde{S}_2[i,j];$$
$$P[j,i] = \lambda_j \tilde{S}_1[j,i] + \tilde{S}_2[j,i].$$

By the second condition of the encoding matrix, the values of λ_i and λ_j are different for $i \neq j$. Thus, the entries of $\tilde{S}_1$ and $\tilde{S}_2$ can be uniquely solved for all $i \neq j$ through

$$\begin{aligned} \tilde{S}_1[i,j] = \tilde{S}_1[j,i] &= \frac{P[i,j]-P[j,i]}{\lambda_i - \lambda_j}; \\ \tilde{S}_2[i,j] = \tilde{S}_2[j,i] &= \frac{\lambda_j P[i,j] - \lambda_i P[j,i]}{\lambda_j - \lambda_i}. \end{aligned} \tag{18}$$

We consider the extraction process on $\tilde{S}_1$. Let the Φ_{DC} consist of k row vectors $\{\phi_1^t, \phi_2^t \dots \phi_k^t\}$. As all the non-diagonal elements of $\tilde{S}_1$ are known, the $k-1$ non-diagonal elements in the ith row are given by

$$\phi_i^t S_1 [\ \phi_1 \dots \phi_{i-1} \quad \phi_{i+1} \dots \phi_k\ ].$$

The size of right matrix is $(k-1) \times (k-1)$, and the matrix is invertible by the third condition of encoding matrix. Thus, the client can obtain

$$\left\{ \phi_i^t S_1 | 1 \leq i \leq k \right\}.$$

We can delete the last entity $\phi_i^t S_1$ to obtain

$$\begin{bmatrix} \phi_1^t \\ \vdots \\ \phi_{k-1}^t \end{bmatrix} S_1.$$

The left matrix is also invertible by the third condition of encoding matrix, so that the client can recover the S_1. By following similar techniques, the S_2 can be extracted from the $\tilde{S}_2$.

V. DISCUSSIONS

This section shows the comparisons between the Exact-MSR codes [2] and the proposed codes in encoding process, in terms of encoding complexities and the size of finite fields. By Corollary 8 of [2], the encoding process of [2] are the puncturing codes for $[n' = n+i; k' = k+i; d' = d+i]$ Exact-MSR codes with $d' = 2k' - 2$ and $i = d - 2k + 2$. By this definition, the size of encoding matrix is $n \times (2d-2k+2)$, and the size of message matrix is $(2d-2k+2) \times (d-k+1)$. Thus, the encoding process of [2] requires $O(n(2d-2k+2)(d-k+1))$ operations to calculate the matrix product. In contrast, the encoding of proposed code requires $O(nd(d-k+1))$ operations. Additionally, for $d > 2k-2$, the encoding process [2] involves the message-symbol remapping process. By the method addressed in [2], the message-symbol remapping process requires about five times of matrix multiplications, and each matrix multiplication takes $O(k'^3)$ operations. As the message-symbol remapping process is not necessary for the proposed Exact-MSR codes, the computational cost can be significantly reduced. Besides, the proposed codes still have lower computational overhead for date reconstruction and node regeneration. However, due to the length limitation of this paper, we do not include the comprehensive comparison here. For the size of finite field, the Section V of [2] indicates that the field size $n(d-k+1)$ suffices at $d = 2k-2$. As the [2] is the punctured version of $[n'; k'; d']$ Exact-MSR codes for $d > 2k-2$, a precise value of required field size is $(n+d-2k+2)(d-k+1)$ in this case. Table I summarizes the comparisons.

VI. CONCLUSIONS

This paper presents the $[n, k, d \geq 2k-2]$ Exact-MSR codes whose codewords are directly computed from the source symbols without having to use the shortening technique. We also present a new version of encoding matrix with shuffling the columns of Vandermande matrix, such that the size of finite field $GF(q)$ can be reduced up to $n \leq q$ for the q belonging to a power of two. The proposed techniques facilitate practical implementations.

REFERENCES

[1] I. S. Reed and G. Solomon , "Polynomial Codes over Certain Finite Fields", *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

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

[3] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran, "Network Coding for distributed storage systems," *in Proc. 26th IEEE International Conference on Computer Communications (INFOCOM)*, Anchorage, May 2007, pp. 2000–2008.

[4] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic Regenerating codes for Distributed Storage,*" in Proc. 45th Annual Allerton Conference on Control, Computing, and Communication*, Urbana-Champaign, Sep. 2007.

[5] N. B. Shah, K. V. Rashmi, P. V. Kumar and K. Ramchandran, "Distributed storage codes with repair-by-transfer and non-achievability of interior points on the storage-bandwidth tradeoff," *IEEE Transactions on Information Theory*, vol. 58, no. 3, pp. 1837-1852, 2012.

[6] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment,*" in Proc. 2009 IEEE international conference on Symposium on Information Theory (ISIT)*, vol. 4, pp. 2276-2280, 2009.

[7] D. Cullina, A. G. Dimakis, and T. Ho, "Searching for Minimum Storage Regenerating Codes," *in Proc. 47th Annual Allerton Conference on Communication, Control*, and Computing, Urbana-Champaign, 2009.

[8] 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 Transactions on Information Theory*, vol. 58, no. 4, pp. 2134-2158, 2012.