Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction

K. V. Rashmi, Nihar B. Shah, and P. Vijay Kumar, *Fellow, IEEE*

***Abstract*—Regenerating codes are a class of distributed storage codes that allow for efficient repair of failed nodes, as compared to traditional erasure codes. An $[n, k, d]$ regenerating code permits the data to be recovered by connecting to any k of the n nodes in the network, while requiring that a failed node be repaired by connecting to any d nodes. The amount of data downloaded for repair is typically much smaller than the size of the source data. Previous constructions of exact-regenerating codes have been confined to the case $n = d+1$. In this paper, we present optimal, explicit constructions of (a) Minimum Bandwidth Regenerating (MBR) codes for all values of $[n, k, d]$ and (b) Minimum Storage Regenerating (MSR) codes for all $[n, k, d \geq 2k-2]$, using a new *product-matrix* framework. The product-matrix framework is also shown to significantly simplify system operation. To the best of our knowledge, these are the first constructions of exact-regenerating codes that allow the number n of nodes in the network, to be chosen independent of the other parameters. The paper also contains a simpler description, in the product-matrix framework, of a previously constructed MSR code with $[n = d+1, k, d \geq 2k-1]$.**

***Index Terms*—Distributed storage, interference alignment, network coding, node repair, partial data recovery, product-matrix framework, regenerating codes.**

I. Introduction

In a distributed storage system, information pertaining to a data file (the *message*) is dispersed across nodes in a network in such a manner that an end-user can retrieve the data stored by tapping into a subset of the nodes. A popular option that reduces network congestion and that leads to increased resiliency in the face of node failures is to employ erasure coding, for example, by calling upon maximum-distance-separable (MDS) codes such as Reed-Solomon (RS) codes. Let B be the total file size measured in terms of symbols over a finite field $\mathbb{F}_q$ of size q. With RS codes, data is stored across n nodes in the network in such a way that the entire message can be recovered by a data-collector by connecting to any k nodes, a process that we will refer to as *data-reconstruction*. Several distributed storage systems such as RAID-6 [1], OceanStore [2] and Total Recall [3] employ such an erasure-coding option.

A. Regenerating Codes

Upon failure of an individual node, a self-sustaining data-storage network must necessarily possess the ability to *regenerate* (i.e., repair) a failed node. An obvious means to accomplish this is to permit the replacement node to connect to any k nodes, download the entire message, and extract the data that was stored in the failed node. But downloading the entire B units of data in order to recover the data stored in a single node that stores only a fraction of the entire message is wasteful, and raises the question as to whether there is a better option. Such an option is indeed available and provided by the concept of a *regenerating code* introduced in the pioneering paper by Dimakis *et al.* [4].

Conventional RS codes treat each fragment stored in a node as a single symbol belonging to the finite field $\mathbb{F}_q$. It can be shown that when individual nodes are restricted to perform only linear operations over $\mathbb{F}_q$, the total amount of data download needed to repair a failed node can be no smaller than B, the size of the entire file. In contrast, regenerating codes are codes over a vector alphabet and hence treat each fragment as being comprised of α symbols over the finite field $\mathbb{F}_q$. Linear operations over $\mathbb{F}_q$ in this case, permit the transfer of a fraction of the data stored at a particular node. Apart from this new parameter α, two other parameters d and β are associated with regenerating codes. Under the definition of regenerating codes introduced in [4], a failed node is permitted to connect to an arbitrary set of d of the remaining nodes while downloading $\beta \leq \alpha$ symbols from each node. This process is termed as *regeneration* and the total amount $d\beta$ of data downloaded for repair purposes as the *repair bandwidth*. Further, the set of d nodes aiding in the repair are termed as *helper nodes*. Typically, with a regenerating code, the average repair bandwidth is small compared to the size of the file B.

It will be assumed throughout the paper, that whenever mention is made of an $[n, k, d]$ regenerating code, the code is such that k and d are the minimum values under which data-reconstruction and regeneration can always be guaranteed. This restricts the range of d to

$$k \leq d \leq n-1. \tag{1}$$

The first inequality arises because if the regeneration parameter d were less than the data-reconstruction parameter k then one

Manuscript received May 23, 2010; revised March 29, 2011; accepted March 29, 2011. Date of current version July 29, 2011. The results in this paper were presented in part at the Information Theory and Applications Workshop, San Diego, CA, Feb. 2011.

K. V. Rashmi and N. B. Shah are with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore-560012, India (e-mail: rashmikv@ece.iisc.ernet.in; nihar@ece.iisc.ernet.in).

P. V. Kumar is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore-560012, India. He is also with the Electrical Engineering Systems Department, University of Southern California, Los Angeles, CA 90089-2565 USA (e-mail: vijay@ece.iisc.ernet.in).

Communicated by N. Kashyap, Associate Editor for Coding Theory.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2011.2159049

could, in fact, reconstruct data by connecting to any $d < k$ nodes (treating the data stored in every other node as a function of that stored in these d nodes) thereby contradicting the minimality of k. Finally, while a regenerating code over $\mathbb{F}_q$ is associated with the collection of parameters

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

it will be found more convenient to regard parameters $\{n, k, d\}$ as primary and $\{\alpha, \beta, B\}$ as secondary and thus we will make frequent references in the sequel, to a code with these six parameters as an $[n, k, d]$ regenerating code having parameter set (α, β, B).

B. Cut-Set Bound and the Storage Versus Repair-Bandwidth Tradeoff

A major result in the field of regenerating codes is the proof in [5] that uses the cut-set bound of network coding to establish that the parameters of a regenerating code must necessarily satisfy

$$B \leq \sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta\}. \tag{2}$$

It is desirable to minimize both α as well as β since, minimizing α results in a minimum storage solution, while minimizing β (for fixed d) results in a storage solution that minimizes repair bandwidth. As can be deduced from (2), it is not possible to minimize both α and β simultaneously and thus there is a tradeoff between choices of the parameters α and β. The two extreme points in this tradeoff are termed the minimum storage regeneration (MSR) and minimum bandwidth regeneration (MBR) points respectively. The parameters α and β for the MSR point on the tradeoff can be obtained by first minimizing α and then minimizing β to obtain

$$\begin{aligned} \alpha &= \frac{B}{k} \\ \beta &= \frac{B}{k(d-k+1)}. \end{aligned} \tag{3}$$

Reversing the order leads to the MBR point which thus corresponds to

$$\begin{aligned} \beta &= \frac{2B}{k(2d-k+1)} \\ \alpha &= \frac{2dB}{k(2d-k+1)} = d\beta. \end{aligned} \tag{4}$$

We define an *optimal* $[n, k, d]$ regenerating code as a code with parameters (α, β, B) satisfying the twin requirements that:

1) the parameter set (α, β, B) achieves the cut-set bound with equality;
2) decreasing either α or β will result in a new parameter set that violates the cut set bound.

An MSR code is then defined as an $[n, k, d]$ regenerating code whose parameters (α, β, B) satisfy (3) and similarly, an MBR code as one with parameters (α, β, B) satisfying (4). Clearly, both MSR and MBR codes are optimal regenerating codes.

C. Striping of Data

The nature of the cut-set bound permits a divide-and-conquer approach to be used in the application of optimal regenerating codes to large file sizes, thereby simplifying system implementation. This is explained below.

Given an optimal $[n, k, d]$ regenerating code with parameter set (α, β, B), a second optimal regenerating code with parameter set $(\alpha' = \delta\alpha,\ \beta' = \delta\beta,\ B' = \delta B)$ for any positive integer δ can be constructed, by dividing the δB message symbols into δ groups of B symbols each, and applying the (α, β, B) code to each group independently. Secondly, a common feature of both MSR and MBR regenerating codes is that in either case, their parameter set (α, β, B) is such that both α and B are multiples of β and further that $\frac{\alpha}{\beta}$, $\frac{B}{\beta}$ are functions only of n, k and d. It follows that if one can construct an (optimal) $[n, k, d]$ MSR or MBR code with $\beta = 1$, then one can construct an (optimal) $[n, k, d]$ MSR or MBR code for any larger value of β. In addition, from a practical standpoint, a code constructed through concatenation of codes for a smaller β will in general, be of lesser complexity (see Section VI-C). For these reasons, in the present paper we design codes for the case of $\beta = 1$. Thus, throughout the remainder of the paper, we will assume that $\beta = 1$. In the terminology of distributed storage, such a process is called striping.

We document below the values of α and B of MSR and MBR codes respectively, when $\beta = 1$:

$$\alpha = d - k + 1 \tag{5}$$

$$B = k(d - k + 1) \tag{6}$$

for MSR codes and

$$\alpha = d \tag{7}$$

$$B = kd - \binom{k}{2} \tag{8}$$

in the case of MBR codes.

D. Additional Terminology

1) Exact Versus Functional Regeneration: In the context of a regenerating code, by functional regeneration of a failed node f, we will mean, replacement of the failed node by a new node f' in such a way that following replacement, the resulting network of n nodes continues to possess the data-reconstruction and regeneration properties. In contrast, by exact-regeneration, we mean replacement of a failed node f by a replacement node f' that stores exactly the same data as was stored in node f prior to failure. We will use the term *exact-regenerating code* to denote a regenerating code that has the capability of exactly regenerating each instance of a failed node. An exact-regenerating code is to be preferred over a functional-regenerating code wherever possible, due to the following reasons. In a system where the code coefficients are globally known, under functional-regeneration there is need for the network to inform all nodes of the replacement. Moreover, the repair and decoding algorithms also need to be re-tuned for the new set of coefficients. These additional overheads are clearly unnecessary under exact-regeneration. In addition, exact-regeneration permits the code to be systematic, as described below.

2) Systematic Regenerating Codes: A systematic regenerating code can be defined as a regenerating code designed in such a way that the B message symbols are explicitly present amongst the $k\alpha$ code symbols stored in a select set of k nodes, termed as the systematic nodes. Clearly, in the case of systematic regenerating codes, exact-regeneration of (the systematic portion of the data stored in) the systematic nodes is mandated.

3) Linear Regenerating Codes: A linear regenerating code is defined as a regenerating code in which:

a) the code symbols stored in each node are linear combinations over $\mathbb{F}_q$ of the B message symbols $\{u_i\}_{i=1}^{B}$;
b) the symbols passed by a helper node h to aid in the regeneration of a failed node f are linear over $\mathbb{F}_q$ in the α symbols stored in node h.

It follows as an easy consequence, that linear operations suffice for a data-collector to recover the data from the $k\alpha$ code symbols stored in the k nodes that it has connected to. Similarly, the replacement node for a failed node f, performs linear operations on the d symbols passed on to it by the d helper nodes $\{h_i\}_{i=1}^{d}$ aiding in the regeneration.

E. Results of the Present Paper

While prior work is described in greater detail in Section II, we begin by providing a context for the results presented here.

Background: To-date, explicit and general constructions for exact-regenerating codes at the MSR point have been found only for the case $[n = d + 1, k, d \geq 2k - 1]$. Similarly at the MBR point, the only explicit code to previously have been constructed is for the case $d = n - 1$. Thus, all existing code constructions limit the total number of nodes n in the system to $d + 1$. This is restrictive since in this case, the system can handle only a single node failure at a time. Also, such a system does not permit additional storage nodes to be brought into the system.

A second open problem in this area that has recently drawn attention is as to whether or not the storage-repair bandwidth tradeoff is achievable under the additional requirement of exact-regeneration. It has previously been shown that no linear code can achieve the MSR point for any $[n, k, d < 2k-3]$ with $\beta = 1$, but is achievable for all parameters $[n, k, d]$ when B (and hence β as well) is allowed to approach infinity.

Results Presented in Present Paper: In this paper, (optimal) explicit constructions of exact-regenerating MBR codes for all values of $[n, k, d]$ and exact-regenerating MSR codes for all $[n, k, d \geq 2k - 2]$ are presented. The constructions are of a product-matrix nature that is shown to significantly simplify operation of the distributed storage network. The constructions presented prove that the MBR point for exact-regeneration can be achieved for all values of the parameters and that the MSR point can be achieved for all parameters satisfying $d \geq 2k - 2$. In both constructions, the message size is as dictated by cut-set bound. The paper also contains a simpler description, in the product-matrix framework, of an MSR code for the parameters $[n = d + 1, k, d \geq 2k - 1]$ that was previously constructed in [6], [7].

A brief overview of prior work in this field is provided in Section II. The product-matrix framework underlying the code construction is described in Section III. An exact-regenerating MBR code for all feasible values of the parameters $[n, k, d]$ is presented in Section IV, and an exact-regenerating MSR code for all $[n, k, d \geq 2k - 2]$ is presented in Section V. Implementation advantages of the particular product-matrix nature of the code constructions provided here are described in Section VI. The final section, Section VII, draws conclusions. Appendix A contains a simpler description, in the product-matrix framework, of an MSR code with parameter d satisfying $[n = d + 1, k, d \geq 2k - 1]$, that was previously constructed in [6] and [7].

II. Prior Work

The concept of regenerating codes was introduced in [4], where it was shown that permitting the storage nodes to store more than B/k units of data helps in reducing the repair bandwidth. Several distributed systems were analyzed, and estimates of the mean node availability in such systems obtained. Using these values, it was shown through simulation, that regenerating codes can reduce repair bandwidth compared to other designs, while simplifying system architecture.

The problem of minimizing repair bandwidth for functional repair of a failed storage node is considered in [4], [5]. Here, the evolution of the storage network through a sequence of failures and regenerations is represented as a network, with all possible data-collectors represented as sinks. The data-reconstruction requirement is formulated as a multicast network coding problem, with the network having an infinite number of nodes. The cut-set analysis of this network leads to the relation between the parameters of a regenerating code given in (2). It can be seen that there is a tradeoff between the choice of the parameters α and β for a fixed B and this is termed as the storage-repair bandwidth tradeoff. It has been shown ([5], [8]) that this tradeoff is achievable under functional-regeneration. However, the coding schemes suggested are not explicit and require large field size. The journal version [9] also contains a handcrafted functional-regenerating code for the MSR point with $[n = 4, k = 2, d = 3]$.

A study of the computational complexity of regenerating codes is carried out in [10], in the context of random linear regenerating codes that achieve functional repair.

The problem of exact-regeneration was first considered independently in [11]–[13]. In [11], it is shown that the MSR point is achievable under exact-regeneration when $(k = 2, d = n - 1)$. The coding scheme proposed is based on the concept of interference alignment developed in the context of wireless communication. However, the construction is not explicit and has a large field size requirement. In [13], the authors carry out a computer search to find exact-regenerating codes at the MSR point, resulting in identification of codes with parameters $[n = 5, k = 3, d = 4]$.

The first, explicit construction of regenerating codes for a general set of parameters was provided for the MBR point in [12] with $d = n - 1$ and arbitrary k. These codes have low regeneration complexity as no computation is involved during the exact-regeneration of a failed node. The field size required is of the order of n^2. In addition, [12] (see also [14]) also contains the construction of an explicit MSR code for $d = k + 1$, that performs approximately-exact-regeneration of all failed nodes,

i.e., regeneration where a part of the code is exactly regenerated, and the remaining is functionally regenerated (it is shown subsequently in [6], [14] that exact-regeneration is not possible, when $k > 4$, for the set of parameters considered therein).

MSR codes performing a hybrid of exact and functional-regeneration are provided in [15], for the parameters $d = k + 1$ and $n > 2k$. The codes given even here are nonexplicit, and have high complexity and large field-size requirement.

A code structure that guarantees exact-regeneration of just the systematic nodes is provided in [6], for the MSR point with parameters $[n = d + 1, k, d \geq 2k - 1]$. This code makes use of interference alignment, and is termed as the 'MISER' code in journal-submission version [14] of [6]. Subsequently, it was shown in [7] that for this set of parameters, the code introduced in [6] for exact-regeneration of only the systematic nodes can also be used to repair the nonsystematic (parity) node failures exactly provided repair construction schemes are appropriately designed. Such an explicit repair scheme is indeed designed and presented in [7]. The paper [7] also contains an exact-regenerating MSR code for parameter set $[n = 5, k = 3, d = 4]$.

A proof of nonachievability of the cut-set bound on exact-regeneration at the MSR point with linear codes, for the parameters $[n, k, d < 2k - 3]$ when $\beta = 1$, is provided in [6], [14]. On the other hand, the MSR point is shown to be achievable in the limiting case of B approaching infinity (i.e., β approaching infinity) in [16], [17].

A flexible setup for regenerating codes is described in [18], where a data-collector (or a replacement node) can perform data-reconstruction (or regeneration) irrespective of the number of nodes to which it connects, provided the total data downloaded exceeds a certain threshold.

In [19], the authors establish that essentially all points on the interior of the tradeoff (i.e., points other than MSR and MBR) are not achievable under exact-regeneration.

III. Common Product-Matrix Framework

The constructions described in this paper follow a common product-matrix framework. Under this framework, each codeword in the distributed storage code can be represented by an $(n \times \alpha)$ *code matrix* C whose ith row $\underline{c}_i^t$ contains the α symbols stored by the ith node. Each code matrix is the product

$$C = \Psi M \tag{9}$$

of an $(n \times d)$ *encoding matrix* Ψ and an $(d \times \alpha)$ *message matrix* M. The entries of the matrix Ψ are fixed a priori and are independent of the message symbols. The message matrix M contains the B message symbols, with some symbols possibly repeated. We will refer to the ith row $\underline{\psi}_i^t$ of Ψ as the encoding vector of node i as it is this vector that is used to encode the message into the form in which it is stored within the ith node

$$\underline{c}_i^t = \underline{\psi}_i^t M \tag{10}$$

where the superscript 't' is used to denote the transpose of a matrix. Throughout this paper, we consider all symbols to belong to a finite field $\mathbb{F}_q$ of size q.

This common structure of the code matrices leads to common architectures for both data-reconstruction and exact-regeneration, as explained in greater detail below. It also endows the codes with implementation advantages that are discussed in Section VI.

Data-reconstruction amounts to recovering the message matrix M from the $k\alpha$ symbols obtained from an arbitrary set of k storage nodes. Let us denote the set of k nodes to which the data-collector connects as $\{i_1, \ldots, i_k\}$. The jth node in this set passes on the message vector $\underline{\psi}_{i_j}^t M$ to the data-collector. The data-collector thus obtains the product matrix

$$\Psi_{\mathrm{DC}} M$$

where Ψ_{DC} is the submatrix of Ψ consisting of the k rows $\left\{\underline{\psi}_{i_1}, \ldots, \underline{\psi}_{i_k}\right\}$. It then uses the properties of the matrices Ψ and M to recover the message. The precise procedure for recovering M is a function of the particular construction.

As noted above, each node in the network is associated to a distinct $(d \times 1)$ encoding vector $\underline{\psi}_i$. In the regeneration process, we will need to call upon a related vector $\underline{\mu}_i$ of length α, that contains a subset of the components of $\underline{\psi}_i$. To regenerate a failed node f, the node replacing the failed node connects to an arbitrary subset $\{h_1, \ldots, h_d\}$ of d storage nodes which we will refer to as the d *helper nodes*. Each helper node passes on the inner product of the α symbols stored in it with $\underline{\mu}_f$, to the replacement node: the helper node h_j passes

$$\underline{\psi}_{h_j}^t M \underline{\mu}_f.$$

The replacement node thus obtains the product matrix

$$\Psi_{\mathrm{repair}} M \underline{\mu}_f$$

where Ψ_{repair} is the submatrix of Ψ consisting of the d rows $\left\{\underline{\psi}_{h_1}, \ldots, \underline{\psi}_{h_d}\right\}$. From this it turns out, as will be shown subsequently, that one can recover the desired symbols. Here again, the precise procedure is dependent on the particular construction.

Remark 1: An important feature of the product-matrix construction presented here, is that each of the nodes h_j participating in the regeneration of node f, needs only have knowledge of the encoding vector of the failed node f and not the identity of the other nodes participating in the regeneration. This significantly simplifies the operation of the system.

Systematic Codes: The following theorem shows that any linear exact-regenerating code can be converted to a systematic form via a linear remapping of the symbols. The proof of the theorem may be found in Appendix B.

Theorem 1: Any linear exact-regenerating code can be converted to a systematic form via a linear remapping of the message symbols. Furthermore, the resulting code is also linear and possesses the data-reconstruction and exact-regeneration properties of the original code.

Thus, all codes provided in the present paper can be converted to a systematic form via a linear remapping of the message symbols. Specific details on the product-matrix MBR and

MSR codes in systematic form are provided in the respective sections, Sections IV and V.

IV. Product-Matrix MBR Code Construction

In this section, we identify the specific make-up of the encoding matrix Ψ and the message matrix M that results in an $[n,k,d]$ MBR code with $\beta=1$. A notable feature of the construction is that it is applicable to all feasible values of $[n,k,d]$, i.e., all n, k, d satisfying $k \leq d \leq n-1$. Since the code is required to be an MBR code with $\beta=1$, it must possess the data-reconstruction and exact-regeneration properties required of a regenerating code, and in addition, have parameters $\{\alpha, B\}$ that satisfy (7) and (8). Equation (8) can be rewritten in the form

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

Thus the parameter set of the desired $[n,k,d]$ MBR code is

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

Let S be a $(k \times k)$ matrix constructed so that the $\binom{k+1}{2}$ entries in the upper-triangular half of the matrix are filled up by $\binom{k+1}{2}$ distinct message symbols drawn from the set $\{u_i\}_{i=1}^{B}$. The $\binom{k}{2}$ entries in the strictly lower-triangular portion of the matrix are then chosen so as to make the matrix S a symmetric matrix. The remaining $k(d-k)$ message symbols are used to fill up a second $(k \times (d-k))$ matrix T. The message matrix M is then defined as the $(d \times d)$ symmetric matrix given by

$$M = \begin{bmatrix} S & T \\ T^t & 0 \end{bmatrix}. \tag{11}$$

The symmetry of the matrix will be found to be instrumental when enabling node repair. Next, define the encoding matrix Ψ to be any $(n \times d)$ matrix of the form

$$\Psi = [\,\Phi \quad \Delta\,]$$

where Φ and Δ are $(n \times k)$ and $(n \times (d-k))$ matrices respectively, chosen in such a way that:

1) any d rows of Ψ are linearly independent;
2) any k rows of Φ are linearly independent.

The above requirements can be met, for example, by choosing Ψ to be either a Cauchy [20] or else a Vandermonde matrix.[1] The only constraint on the field size comes from the above required properties of the encoding matrix Ψ. For instance, when Ψ is chosen as a Vandermonde matrix, any field of size n or higher suffices.

As per the product-matrix framework, the code matrix is then given by $C = \Psi M$. The two theorems below establish that the code presented is an $[n,k,d]$ MBR code by establishing respectively, the exact-regeneration and data-reconstruction properties of the code.

Theorem 2 (MBR Exact-Regeneration): In the code presented, exact-regeneration of any failed node can be achieved by downloading one symbol each from any d of the $(n-1)$ remaining nodes.

Proof: Let $\underline{\psi}_f^t$ be the row of Ψ corresponding to the failed node f. Thus the d symbols stored in the failed node correspond to the vector

$$\underline{\psi}_f^t M. \tag{12}$$

The replacement for the failed node f connects to an arbitrary set $\{h_j \mid j = 1,\ldots,d\}$ of d helper nodes. Upon being contacted by the replacement node, the helper node h_j computes the inner product

$$\underline{\psi}_{h_j}^t M \underline{\psi}_f$$

and passes on this value to the replacement node. Thus, in the present construction, the vector $\underline{\mu}_f$ equals $\underline{\psi}_f$ itself. The replacement node thus obtains the d symbols $\Psi_{\text{repair}} M \underline{\psi}_f$ from the d helper nodes, where

$$\Psi_{\text{repair}} = \begin{bmatrix} \underline{\psi}_{h_1}^t \\ \underline{\psi}_{h_2}^t \\ \vdots \\ \underline{\psi}_{h_d}^t \end{bmatrix}.$$

By construction, the $(d \times d)$ matrix Ψ_{repair} is invertible. Thus, the replacement node recovers $M\underline{\psi}_f$ through multiplication on the left by $\Psi_{\text{repair}}^{-1}$. Since M is symmetric

$$\left(M\underline{\psi}_f\right)^t = \underline{\psi}_f^t M, \tag{13}$$

and this is precisely the data previously stored in the failed node. ■

Theorem 3 (MBR Data-Reconstruction): In the code presented, all the B message symbols can be recovered by connecting to any k nodes, i.e., the message symbols can be recovered through linear operations on the entries of any k rows of the matrix C.

Proof: Let

$$\Psi_{\text{DC}} = [\,\Phi_{\text{DC}} \quad \Delta_{\text{DC}}\,] \tag{14}$$

be the $(k \times d)$ submatrix of Ψ, corresponding to the k rows of Ψ to which the data-collector connects. Thus, the data-collector has access to the symbols

$$\Psi_{\text{DC}} M = [\,\Phi_{\text{DC}} S + \Delta_{\text{DC}} T^t \quad \Phi_{\text{DC}} T\,]. \tag{15}$$

By construction, Φ_{DC} is a nonsingular matrix. Hence, by multiplying the matrix $\Psi_{\text{DC}} M$ on the left by Φ_{DC}^{-1}, one can recover first T and subsequently, S. ■

A. An Example for the Product-Matrix MBR Code

Let $n=6, k=3, d=4$. Then $\alpha = d = 4$ and $B = 9$. Let us choose $q=7$ so we are operating over $\mathbb{F}_7$. The matrices S and T are filled up by the 9 message symbols $\{u_i\}_{i=1}^{9}$ as follows:

$$S = \begin{bmatrix} u_1 & u_2 & u_3 \\ u_2 & u_4 & u_5 \\ u_3 & u_5 & u_6 \end{bmatrix}, \quad T = \begin{bmatrix} u_7 \\ u_8 \\ u_9 \end{bmatrix} \tag{16}$$

[1] Over a large finite field, a randomly chosen matrix Ψ will suffice with high probability. The present paper does not elaborate on the same, since the focus is on providing explicit, deterministic code constructions.

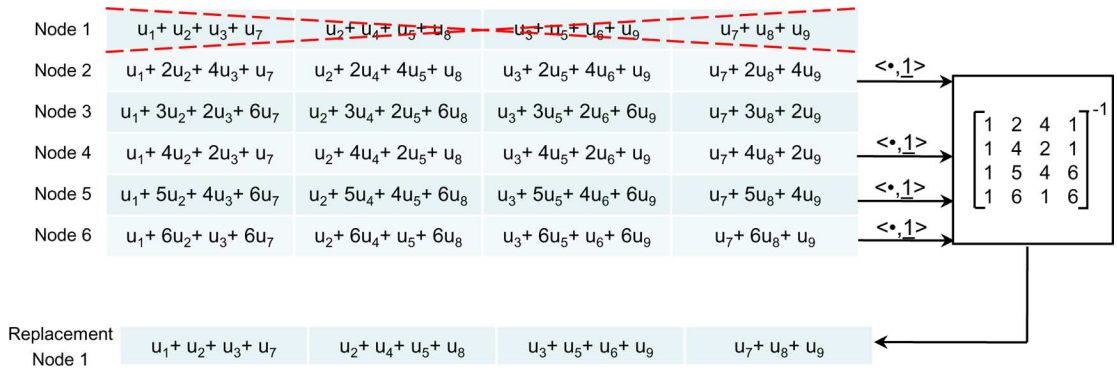

Fig. 1. Example for the MBR code construction: On failure of node 1, the replacement node downloads one symbol each from nodes 2, 4, 5 and 6, using which node 1 is exactly regenerated. The notation $\langle \cdot, \underline{1} \rangle$ indicates an inner product of the stored symbols with the vector $[1 \quad 1 \quad 1 \quad 1]^t$.

so that the message matrix M is given by

$$M = \begin{bmatrix} u_1 & u_2 & u_3 & u_7 \\ u_2 & u_4 & u_5 & u_8 \\ u_3 & u_5 & u_6 & u_9 \\ u_7 & u_8 & u_9 & 0 \end{bmatrix}. \tag{17}$$

We choose Ψ to be the (6×4) Vandermonde matrix over $\mathbb{F}_7$ given by

$$\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 \\ 1 & 3 & 2 & 6 \\ 1 & 4 & 2 & 1 \\ 1 & 5 & 4 & 6 \\ 1 & 6 & 1 & 6 \end{bmatrix}. \tag{18}$$

Fig. 1 shows at the top, the (6×4) code matrix $C = \Psi M$ with entries expressed as functions of the message symbols $\{u_i\}_{i=1}^{9}$. The rest of the figure explains how exact-regeneration of failed node 1 takes place. To regenerate node 1, the helper nodes (nodes 2, 4, 5, 6 in the example), pass on their respective inner products $\underline{\psi}_\ell^t M[1 \quad 1 \quad 1 \quad 1]^t$ for $\ell = 2, 4, 5, 6$. The replacement node then recovers the data stored in the failed node by multiplying by $\Psi_{\text{repair}}^{-1}$ where

$$\Psi_{\text{repair}} = \begin{bmatrix} 1 & 2 & 4 & 1 \\ 1 & 4 & 2 & 1 \\ 1 & 5 & 4 & 6 \\ 1 & 6 & 1 & 6 \end{bmatrix} \tag{19}$$

as explained in the proof of Theorem 2 above.

B. Systematic Version of the Code

As pointed out in Section III, any exact-regenerating code can be made systematic through a nonsingular transformation of the message symbols. In the present case, there is a simpler approach, in which the matrix Ψ can be chosen in such a way that the code is automatically systematic. We simply make the choice:

$$\Psi = \begin{bmatrix} I_k & 0 \\ \tilde{\Phi} & \tilde{\Delta} \end{bmatrix} \tag{20}$$

where I_k is the $(k \times k)$ identity matrix, 0 is a $(k \times (d-k))$ zero matrix, $\tilde{\Phi}$ and $\tilde{\Delta}$ are matrices of sizes $((n-k) \times k)$ and $((n-k) \times (d-k))$ respectively, such that $\begin{bmatrix} \tilde{\Phi} & \tilde{\Delta} \end{bmatrix}$ is a Cauchy matrix[2]. Clearly the code is systematic. It can be verified that the matrix Ψ has the properties listed just above Theorem 2.

V. The Product-Matrix MSR Code Construction

In this section, we identify the specific make-up of the encoding matrix Ψ and the message matrix M that results in an $[n, k, d]$ MSR code with $\beta = 1$. The construction applies to all parameters $[n, k, d \geq 2k-2]$.[3] Since the code is required to be an MSR code with $\beta = 1$, it must possess the data-reconstruction and exact-regeneration properties required of a regenerating code, and in addition, have parameters $\{\alpha, B\}$ that satisfy (5) and (6). We begin by constructing an MSR code in the product-matrix framework for $d = 2k-2$ and will show in Section V-C how this can be very naturally extended to yield codes with $d > 2k-2$.

At the MSR point with $d = 2k-2$ we have

$$\alpha = d - k + 1 = k - 1 \tag{21}$$

and hence

$$d = 2\alpha. \tag{22}$$

Also

$$B = k\alpha = \alpha(\alpha+1). \tag{23}$$

We define the $(d \times \alpha)$ message matrix M as

$$M = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \tag{24}$$

where S_1 and S_2 are $(\alpha \times \alpha)$ symmetric matrices constructed such that the $\binom{\alpha+1}{2}$ entries in the upper-triangular part of each of the two matrices are filled up by $\binom{\alpha+1}{2}$ distinct message symbols. Thus, all the $B = \alpha(\alpha+1)$ message symbols are contained in the two matrices S_1 and S_2. The entries in the strictly lower-triangular portion of the two matrices S_1 and S_2 are chosen so as to make the matrices S_1 and S_2 symmetric.

Next, we define the encoding matrix Ψ to be the $(n \times d)$ matrix given by

$$\Psi = \begin{bmatrix} \Phi & \Lambda\Phi \end{bmatrix} \tag{25}$$

[2]In general, any matrix, all of whose submatrices are of full rank, will suffice.

[3]As mentioned previously, it is impossible to construct linear MSR codes for the case of $d < 2k-3$ when $\beta = 1$ (see [6], [14]).

where Φ is an $(n \times \alpha)$ matrix and Λ is an $(n \times n)$ *diagonal* matrix. The elements of Ψ are chosen such that the following conditions are satisfied:

1) any d rows of Ψ are linearly independent;
2) any α rows of Φ are linearly independent;
3) the n diagonal elements of Λ are distinct.

The above requirements can be met, for example, by choosing Ψ to be a Vandermonde matrix with elements chosen carefully to satisfy the third condition. In this case, let the ith row of Ψ (for $i = 1, \ldots, n$) be $\underline{\psi}_i = [1 \;\; x_i \;\; \cdots \;\; x_i^{d-1}]$, which gives $\Lambda = \text{diag}\,\{x_1^\alpha, x_2^\alpha, \ldots, x_n^\alpha\}$. In order to satisfy the third property, one may choose $\mathbb{F}_q$ to be any field of size $n(d-k+1)$ or higher, with $x_i = g^{i-1}$, where g is the generator of the multiplicative group of the finite field $\mathbb{F}_q$. Note that as in the MBR code, the only constraint on the field size in this construction arises from the above required properties of the encoding matrix Ψ.

Then under our code-construction framework, the ith row of the $(n \times \alpha)$ product matrix $C = \Psi M$, contains the α code symbols stored by the ith node. The two theorems below establish that the code presented is an $[n, k, d]$ MSR code by establishing respectively, the exact-regeneration and data-reconstruction properties of the code.

Theorem 4 (MSR Exact-Regeneration): In the code presented, exact-regeneration of any failed node can be achieved by downloading one symbol each from any $d = 2k - 2$ of the remaining $(n - 1)$ nodes.

Proof: Let $\left[\underline{\phi}_f^t \quad \lambda_f \underline{\phi}_f^t\right]$ be the row of Ψ corresponding to the failed node. Thus the α symbols stored in the failed node were

$$\left[\underline{\phi}_f^t \quad \lambda_f \underline{\phi}_f^t\right] M = \underline{\phi}_f^t S_1 + \lambda_f \underline{\phi}_f^t S_2. \tag{26}$$

The replacement for the failed node f connects to an arbitrary set $\{h_j \mid j = 1, \ldots, d\}$ of d helper nodes. Upon being contacted by the replacement node, the helper node h_j computes the inner product $\underline{\psi}_{h_j}^t M \underline{\phi}_f$ and passes on this value to the replacement node. Thus, in the present construction, the vector $\underline{\mu}_f$ equals $\underline{\phi}_f$. The replacement node thus obtains the d symbols $\Psi_{\text{repair}} M \underline{\phi}_f$ from the d helper nodes, where

$$\Psi_{\text{repair}} = \begin{bmatrix} \underline{\psi}_{h_1}^t \\ \underline{\psi}_{h_2}^t \\ \vdots \\ \underline{\psi}_{h_d}^t \end{bmatrix}.$$

By construction, the $(d \times d)$ matrix Ψ_{repair} is invertible. Thus the replacement node now has access to

$$M \underline{\phi}_f = \begin{bmatrix} S_1 \underline{\phi}_f \\ S_2 \underline{\phi}_f \end{bmatrix}.$$

As S_1 and S_2 are symmetric matrices, the replacement node has thus acquired through transposition, both $\underline{\phi}_f^t S_1$ and $\underline{\phi}_f^t S_2$. Using this, it can obtain

$$\underline{\phi}_f^t S_1 + \lambda_f \underline{\phi}_f^t S_2, \tag{27}$$

which is precisely the data previously stored in the failed node. ■

Theorem 5 (MSR Data-Reconstruction): In the code presented, all the B message symbols can be recovered by connecting to any k nodes, i.e., the message symbols can be recovered through linear operations on the entries of any k rows of the code matrix C.

Proof: Let

$$\Psi_{\text{DC}} = \left[\, \Phi_{\text{DC}} \quad \Lambda_{\text{DC}} \Phi_{\text{DC}} \,\right] \tag{28}$$

be the $(k \times d)$ submatrix of Ψ, containing the k rows of Ψ which correspond to the k nodes to which the data-collector connects. Hence, the data-collector obtains the symbols

$$\begin{aligned} \Psi_{\text{DC}} M &= \left[\, \Phi_{\text{DC}} \quad \Lambda_{\text{DC}} \Phi_{\text{DC}} \,\right] \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \\ &= \left[\, \Phi_{\text{DC}} S_1 + \Lambda_{\text{DC}} \Phi_{\text{DC}} S_2 \,\right]. \end{aligned} \tag{29}$$

The data-collector can post-multiply this term with Φ_{DC}^T to obtain

$$\begin{aligned} &[\Phi_{\text{DC}} S_1 + \Lambda_{\text{DC}} \Phi_{\text{DC}} S_2] \Phi_{\text{DC}}^T \\ &= \Phi_{\text{DC}} S_1 \Phi_{\text{DC}}^T + \Lambda_{\text{DC}} \Phi_{\text{DC}} S_2 \Phi_{\text{DC}}^T. \end{aligned} \tag{30}$$

Next, let the matrices P and Q be defined as

$$P = \Phi_{\text{DC}} S_1 \Phi_{\text{DC}}^T \tag{31}$$

$$Q = \Phi_{\text{DC}} S_2 \Phi_{\text{DC}}^T. \tag{32}$$

As S_1 and S_2 are symmetric, the same is true of the matrices P and Q. In terms of P and Q, the data-collector has access to the symbols of the matrix

$$P + \Lambda_{\text{DC}} Q. \tag{33}$$

The $(i,\ j)$th, $1 \le i, j \le k$, element of this matrix is

$$P_{ij} + \lambda_i Q_{ij}, \tag{34}$$

while the (j, i)th element is given by

$$\begin{aligned} &P_{ji} + \lambda_j Q_{ji} \\ &= P_{ij} + \lambda j Q_{ij} \end{aligned} \tag{35}$$

where (35) follows from the symmetry of P and Q. By construction, all the λ_i are distinct and hence using (34) and (35), the data-collector can solve for the values of P_{ij}, Q_{ij} for all $i \neq j$.

Consider first the matrix P. Let Φ_{DC} be given by

$$\Phi_{\text{DC}} = \begin{bmatrix} \underline{\phi}_1^t \\ \vdots \\ \underline{\phi}_{\alpha+1}^t \end{bmatrix}. \tag{36}$$

All the nondiagonal elements of P are known. The elements in the ith row (excluding the diagonal element) are given by

$$\underline{\phi}_i^t S_1 \left[\underline{\phi}_1 \cdots \underline{\phi}_{i-1} \; \underline{\phi}_{i+1} \cdots \underline{\phi}_{\alpha+1}\right]. \tag{37}$$

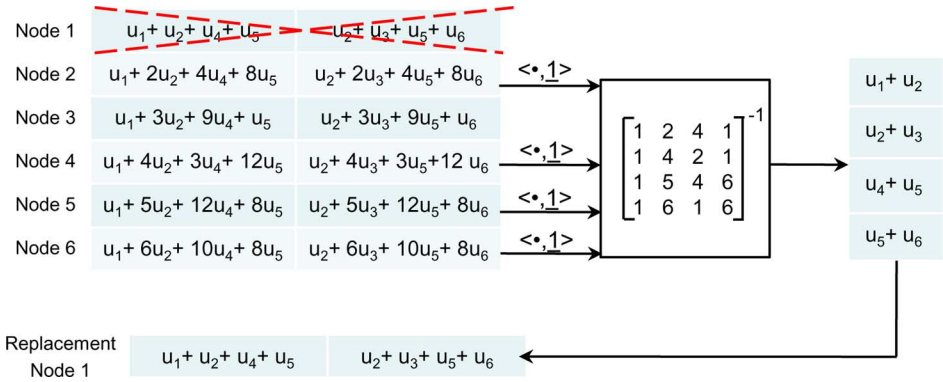

Fig. 2. Example for the MSR code construction: On failure of node 1, the replacement node downloads one symbol each from nodes 2, 4, 5, and 6, using which node 1 is exactly regenerated. The notation $< ., \underline{1} >$ indicates an inner product of the stored symbols with the vector $[\,1 \quad 1\,]^t$.

However, the matrix to the right is nonsingular by construction and hence the data-collector can obtain

$$\left\{\underline{\phi}_i^t S_1 \mid 1 \le i \le \alpha+1\right\}. \tag{38}$$

Selecting the first α of these, the data-collector has access to

$$\begin{bmatrix} \underline{\phi}_1^t \\ \vdots \\ \underline{\phi}_\alpha^t \end{bmatrix} S_1. \tag{39}$$

The matrix on the left is also nonsingular by construction and hence the data-collector can recover S_1. Similarly, using the values of the nondiagonal elements of Q, the data-collector can recover S_2. ∎

Remark 2: It is shown in [6], [14] that *interference alignment* is, in fact, a necessary ingredient of any minimum storage regenerating code. Interference alignment is also present in the product-matrix MSR code, and Appendix C brings out this connection.

A. An Example for the Product-Matrix MSR Code

Let $n = 6$, $k = 3$, $d = 4$. Then $\alpha = d - k + 1 = 2$ and $B = k\alpha = 6$. Let us choose $q = 13$, so we are operating over $\mathbb{F}_{13}$. The matrices S_1 and S_2 are filled up by the six message symbols $\{u_i\}_{i=1}^6$ as follows:

$$S_1 = \begin{bmatrix} u_1 & u_2 \\ u_2 & u_3 \end{bmatrix}, S_2 = \begin{bmatrix} u_4 & u_5 \\ u_5 & u_6 \end{bmatrix} \tag{40}$$

so that the message matrix M is given by

$$M = \begin{bmatrix} u_1 & u_2 \\ u_2 & u_3 \\ u_4 & u_5 \\ u_5 & u_6 \end{bmatrix}. \tag{41}$$

We choose Ψ to be the (6×4) Vandermonde matrix over $\mathbb{F}_{13}$ given by

$$\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 1 \\ 1 & 4 & 3 & 12 \\ 1 & 5 & 12 & 8 \\ 1 & 6 & 10 & 8 \end{bmatrix}. \tag{42}$$

Hence the (6×2) matrix Φ and the (6×6) diagonal matrix Λ are

$$\Phi = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} 1 & & & & & \\ & 4 & & & & \\ & & 9 & & & \\ & & & 3 & & \\ & & & & 12 & \\ & & & & & 10 \end{bmatrix}. \tag{43}$$

Fig. 2 shows at the top, the (6×2) code matrix $C = \Psi M$ with entries expressed as functions of the message symbols $\{u_i\}$. The rest of the figure explains how exact-regeneration of failed node 1 takes place. To regenerate node 1, the helper nodes (nodes 2, 4, 5, 6 in the example), pass on their respective inner products $\underline{\psi}_\ell^t M[1\ 1]^t$ for $\ell = 2, 4, 5, 6$. The replacement node multiplies the symbols it receives with $\Psi_{\text{repair}}^{-1}$, where

$$\Psi_{\text{repair}} = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 4 & 3 & 12 \\ 1 & 5 & 12 & 8 \\ 1 & 6 & 10 & 8 \end{bmatrix} \tag{44}$$

and decodes $S_1\underline{\psi}_1$ and $S_2\underline{\psi}_1$:

$$S_1\underline{\psi}_1 = \begin{bmatrix} u_1+u_2 \\ u_2+u_3 \end{bmatrix}, \ S_2\underline{\psi}_1 = \begin{bmatrix} u_4+u_5 \\ u_5+u_6 \end{bmatrix}. \tag{45}$$

Finally, it processes $S_1\underline{\psi}_1$ and $S_2\underline{\psi}_1$ to obtain the data stored in the failed node as explained in the proof of Theorem 4 above.

B. Systematic Version of the Code

It was pointed out in Section III, that every exact-regenerating code has a systematic version and further, that the code could be made systematic through a process of message-symbol remapping. In the following, we make this more explicit in the context of the product-matrix MSR code.

Let Ψ_k be the $(k \times d)$ submatrix of Ψ, containing the k rows of Ψ corresponding to the k nodes which are chosen to be made systematic. The set of $k\alpha$ symbols stored in these k nodes are given by the elements of the $(k \times \alpha)$ matrix $\Psi_k M$. Let U be a $(k \times \alpha)$ matrix containing the $B = k\alpha$ source symbols. We map

$$\Psi_k M = U \tag{46}$$

and solve for the entries of M in terms of the symbols in U. This is precisely the data-reconstruction process that takes place

when a data-collector connects to the k chosen nodes. Thus, the value of the entries in M can be obtained by following the procedure outlined in Theorem 5. Then, use this M to obtain the code $C = \Psi M$. Clearly, in this representation, the k chosen nodes store the source symbols U in uncoded form.

C. Explicit MSR Product-Matrix Codes for $d \geq 2k-2$

In this section, we show how an MSR code for $d = 2k-2$ can be used to obtain MSR codes for all $d \geq 2k-2$. Our starting point is the following theorem.

Theorem 6: An explicit $[n' = n+1, k' = k+1, d' = d+1]$ exact-regenerating code $\mathcal{C}'$ that achieves the cut-set bound at the MSR point can be used to construct an explicit $[n, k, d]$ exact-regenerating code $\mathcal{C}$ that also achieves the cut-set bound at the MSR point. Furthermore if $d' = ak' + b$ in code $\mathcal{C}'$, $d = ak + b + (a-1)$ in code $\mathcal{C}$. If $\mathcal{C}'$ is linear, so is $\mathcal{C}$.

Proof: If both codes operate at the MSR point, then the number of message symbols B', B in the two cases must satisfy

$$B' = k'(d' - k' + 1) \text{ and } B = k(d - k + 1)$$

respectively, so that

$$B' - B = d - k + 1 = \alpha.$$

We begin by constructing an MSR-point-optimal $[n', k', d']$ exact-regenerating code $\mathcal{C}'$ in systematic form with the first k' rows containing the B' message symbols. Let $\mathcal{C}''$ be the subcode of $\mathcal{C}'$ consisting of all code matrices in $\mathcal{C}'$ whose top row is the all-zero row (i.e., the first α of the B' message symbols are all zero). Clearly, the subcode $\mathcal{C}''$ is of size $q^{B'-\alpha} = q^B$. Note that $\mathcal{C}''$ also possesses the same exact-regeneration and data-reconstruction properties as does the parent code $\mathcal{C}'$.

Let the code $\mathcal{C}$ now be formed from subcode $\mathcal{C}''$ by puncturing (i.e., deleting) the first row in each code matrix of $\mathcal{C}''$. Clearly, code $\mathcal{C}$ is also of size q^B. We claim that $\mathcal{C}$ is an $[n, k, d]$ exact-regenerating code. The data-reconstruction requirement requires that the B underlying message symbols be recoverable from the contents of any k rows of a code matrix C in $\mathcal{C}$. But this follows since, by augmenting the matrices of code $\mathcal{C}$ by placing at the top an additional all-zero row, we obtain a code matrix in $\mathcal{C}''$ and code $\mathcal{C}''$ has the property that the data can be recovered from any $(k+1)$ rows of each code matrix in $\mathcal{C}''$. A similar argument shows that code $\mathcal{C}$ also possesses the exact-regeneration property. Clearly if $\mathcal{C}'$ is linear, so is code $\mathcal{C}$. Finally, we have

$$\begin{aligned} d' &= ak' + b \\ \Rightarrow d + 1 &= a(k+1) + b \\ \Rightarrow d &= ak + b + (a-1). \end{aligned}$$

■

By iterating the procedure in the proof of Theorem 6 above i times we obtain:

Corollary 7: An explicit $[n' = n+i, k' = k+i, d' = d+i]$ exact-regenerating code $\mathcal{C}'$ that achieves the cut-set bound at the MSR point can be used to construct an explicit $[n, k, d]$ exact-regenerating code $\mathcal{C}$ that also achieves the cut-set bound at the MSR point. Furthermore if $d' = ak' + b$ in code $\mathcal{C}'$, $d = ak + b + i(a-1)$ in code $\mathcal{C}$. If $\mathcal{C}'$ is linear, so is $\mathcal{C}$.

The corollary below follows from Corollary 7 above.

Corollary 8: An MSR-point optimal exact-regenerating code $\mathcal{C}$ with parameters $[n, k, d]$ for any $2k-2 \leq d \leq n-1$ can be constructed from an MSR-point optimal exact-regenerating $[n' = n+i, k' = k+i, d' = d+i]$ code $\mathcal{C}'$ with $d' = 2k' - 2$ and $i = d - 2k + 2$. If $\mathcal{C}'$ is linear, so is $\mathcal{C}$.

VI. Analysis and Advantages of the Codes

In this section, we detail the system-implementation advantages of the two code constructions presented in the paper.

A. Reduced Overhead

In the product-matrix based constructions provided, the data stored in the ith storage node in the system is completely determined by the single encoding vector $\underline{\psi}_i$ of length d. This is in contrast to a $(B \times \alpha)$ generator matrix in a general code, comprising of the α encoding vectors of length B as its columns, each associated to a different symbol stored in the node. The encoding vector suffices for the encoding, data-reconstruction, and regeneration purposes. The short length of the encoding vector reduces the overhead associated with the need for nodes to communicate their encoding vectors to the data-collector during data-reconstruction, and to the replacement node during regeneration of a failed node.

Also, in both MBR and MSR code constructions, during regeneration of a failed node, the information passed on to the replacement node by a helper node is only a function of the index of the failed node. Thus, it is independent of the identity of the $(d-1)$ other nodes that are participating in the regeneration. Once again, this reduces the communication overhead by requiring less information to be disseminated.

B. Applicability to Arbitrary n

In any real-world distributed storage application such as peer-to-peer storage, cloud storage, etc, it is natural that the number of nodes may go up or down: in due course of time, new nodes may be added to the system, or multiple nodes may fail or exit the system. For example, in peer-to-peer systems, individual nodes are free to come and go at will. The existing, explicit constructions of exact-regenerating codes [6], [7], [11]–[13] restrict the value of n to be $(d+1)$. On the other hand, the codes presented in this paper are applicable for all values of n, and independent of the values of the parameters k and d. This gives a practical appeal to the code constructions presented here.

C. Complexity

1) Linearity and Field Size: The codes are linear over a chosen finite field $\mathbb{F}_q$, i.e., the source symbols are from this finite field, and any stored symbol is a linear combination of these symbols over $\mathbb{F}_q$. As mentioned previously, to arrive at the product-matrix MBR code, any field of size n or higher suffices, and for the product-matrix MSR code, any field of size $n(d-k+1)$ or higher suffices. By cleverly choosing the matrix Ψ that meets the conditions governing the respective codes, it may often be possible to reduce the field size even further.

2) Striping: The codes presented here divide the entire message into *stripes* of sizes corresponding to $\beta = 1$. Since each stripe is of minimal size, the complexity of encoding, data-reconstruction and regeneration operations, are considerably lowered, and so are the buffer sizes required at data-collectors and replacement nodes. Furthermore, the operations that need to be performed on each stripe are identical and independent, and hence can be performed in parallel efficiently by a GPU/FPGA/multi-core processor.

3) Choice of the Encoding Matrix Ψ: The encoding matrix Ψ, for both the codes described, can be chosen as a Vandermonde matrix. Then each encoding vector can be described by just a scalar. Moreover with this choice, the encoding, data-reconstruction, and regeneration operations are, for the most part, identical to encoding or decoding of conventional Reed-Solomon codes.

VII. Conclusions

In this paper, an explicit MBR code for all values of the system parameters $[n, k, d]$, and an explicit MSR code for all parameters satisfying $[n, k, d \geq 2k-2]$ are presented. Both constructions are based on a common product-matrix framework introduced in this paper, and possess attributes that make them attractive from an implementation standpoint. To the best of our knowledge, these are the first explicit constructions of exact-regenerating codes that allow n to take any value independent of the other parameters; this results in a host of desirable properties such as the ability to optimally handle multiple simultaneous node failures as well as the ability of allowing the total number of storage nodes in the system to vary with time. Our results also prove that the MBR point on the storage-repair bandwidth tradeoff is achievable under the additional constraint of exact-regeneration for all values of the system parameters, and that the MSR point is achievable under exact-regeneration for all $d \geq 2k-2$.

Appendix A
Description of a Previously Constructed MSR Code in the Product-Matrix Framework

An explicit code that performs data-reconstruction, and exact-regeneration of the systematic nodes is provided in [6], for the MSR point with parameters $[n = d+1, k, d \geq 2k-1]$. Subsequently, it was shown in [7] that for this set of parameters, the code introduced in [6] for exact-regeneration of only the systematic nodes can also be used for exact-regeneration of the nonsystematic (parity) nodes, provided repair construction schemes are appropriately designed. Such an explicit repair scheme is indeed designed and presented in [7]. In this section, we provide a simpler description of this code in the product-matrix framework.

As in [6], [7], we begin with the case $d = 2k-1$, since the code as well as both data-reconstruction and exact-regeneration algorithms can be extended to larger values of d by making use of Corollary 8.

At the MSR point, with $d = n-1 = 2k-1$, we have from (5) and (6) that

$$\alpha = d - k + 1 = k \tag{47}$$

$$B = k\alpha = k^2. \tag{48}$$

Let S be a $(k \times k)$ matrix whose entries are precisely the B message symbols $\{u_i\}_{i=1}^{B}$ and let M be the $(2k \times k)$ message matrix[4] given by

$$M = \begin{bmatrix} S \\ S^t \end{bmatrix}. \tag{49}$$

Next, let Φ be a $(k \times k)$ Cauchy matrix over $\mathbb{F}_q$ and ρ a scalar chosen such that

$$\rho \neq 0, \quad \rho^2 \neq 1. \tag{50}$$

Let Ψ be the $(n \times 2k)$ encoding matrix given by

$$\Psi = \begin{bmatrix} I & 0 \\ \Phi & \rho\Phi \end{bmatrix}. \tag{51}$$

The code constructed in [6], [7] can be verified to have an alternate description as the collection of code matrices of the form

$$C = \Psi M = \begin{bmatrix} S \\ \Phi(S + \rho S^t) \end{bmatrix}. \tag{52}$$

Note that the first k nodes store the message symbols in uncoded form and hence correspond to the systematic nodes. A simple description of the exact-regeneration and data-reconstruction properties of the code is presented below.

Theorem 9 (Exact-Regeneration): In the code presented, exact-regeneration of any failed node can be achieved by downloading one symbol each from the remaining $n-1$ nodes.

Proof: In this construction, the vector $\underline{\mu}_f$ used in the exact-regeneration of a failed node f is composed of the first $k = \alpha$ symbols of $\underline{\psi}_f$.

1) Exact-Regeneration of Systematic Nodes: Consider regeneration of the ith systematic node. The k symbols thus desired by the replacement node are $\underline{e}_i^t S$. The replacement node obtains the following $n-1$ symbols from the remaining nodes:

$$\begin{bmatrix} \tilde{I} & 0 \\ \Phi & \rho\Phi \end{bmatrix} \begin{bmatrix} S \\ S^t \end{bmatrix} \underline{e}_i = \begin{bmatrix} \tilde{I} S \underline{e}_i \\ \Phi(S + \rho S^t)\underline{e}_i \end{bmatrix} \tag{53}$$

where $\tilde{I}$ is a $((k-1) \times k)$ matrix which is the identity matrix with ith row removed. Since Φ is full rank by construction, the replacement node has access to

$$\begin{aligned} &\left[(S + \rho S^t)\underline{e}_i\right] \\ &= \left[\rho \underline{e}_i^t S + \underline{e}_i^t S^t\right]^t. \end{aligned} \tag{54}$$

[4]Note that the constructions presented in Sections IV and V employ a $(d \times \alpha)$ matrix M as the message matrix, whereas the dimension of M in the present construction is $((d+1) \times \alpha)$.

From (53) and (54), we see that the replacement node has access to

$$\begin{bmatrix} \tilde{I} & 0 \\ \Phi & \rho\Phi \\ \rho\underline{e}_i^t & \underline{e}_i^t \end{bmatrix} \begin{bmatrix} S \\ S^t \end{bmatrix} \underline{e}_i. \tag{55}$$

Since $\rho \neq 1$, the $(2k \times 2k)$ matrix on the left is nonsingular. This allows the replacement node to recover the symbols $S^t\underline{e}_i$, which are precisely the set of symbols $\underline{e}_i^t S$ desired.

2) Exact-Regeneration of Non-Systematic Nodes: Let $\underline{\phi}_f^t$ be the row of Φ corresponding to the failed node. Then the k symbols stored in the failed node are $\underline{\phi}_f^t(S+\rho S^t)$. The replacement node requests and obtains the following $n-1$ symbols from the remaining nodes:

$$\begin{bmatrix} I & 0 \\ \Phi_{k-1} & \rho\Phi_{k-1} \end{bmatrix} \begin{bmatrix} S \\ S^t \end{bmatrix} \underline{\phi}_f \tag{56}$$

where Φ_{k-1} is the submatrix of Φ containing the $k-1$ rows corresponding to the remaining nonsystematic nodes. This gives the replacement node access to $S\underline{\phi}_f$ and therefore to

$$(S\underline{\phi}_f)^t\underline{\phi}_f = \underline{\phi}_f^t S^t \underline{\phi}_f. \tag{57}$$

Hence the replacement node has access to

$$\begin{bmatrix} I & 0 \\ \Phi_{k-1} & \rho\Phi_{k-1} \\ \underline{0}^t & \underline{\phi}_f^t \end{bmatrix} \begin{bmatrix} S \\ S^t \end{bmatrix} \underline{\phi}_f. \tag{58}$$

The matrix on the left is easily verified to be nonsingular and thus the replacement node acquires $S\underline{\phi}_f$ and $S^t\underline{\phi}_f$ individually from which it can derive the desired vector $(\underline{\phi}_f^t S + \rho\underline{\phi}_f^t S^t)$. ■

Theorem 10 (Data-Reconstruction): In the code presented, all the B message symbols can be recovered by connecting to any k nodes, i.e., the message symbols can be recovered through linear operations on the entries of any k rows of the matrix C.

Proof: We first introduce the following notation to denote submatrices of a matrix. If A is an $(m_1 \times m_2)$ matrix and P, Q are arbitrary subsets of $\{1,\ldots,m_1\}$ and $\{1,\ldots,m_2\}$ respectively, we will use $A_{(P,Q)}$ to denote the submatrix of A containing only the rows and columns, respectively, specified by the indices in P and Q. For the cases when either $P = \{1,\ldots,m_1\}$ or $Q = \{1,\ldots,m_2\}$, we will simply indicate this as "all".

Let $P = \{n_1,\ldots,n_i\}$ and $Q = \{m_1,\ldots,m_{(k-i)}\}$ be the systematic and nonsystematic nodes respectively to which the data-collector connects. Let $T = \{1,\ldots,k\}\backslash P$, i.e., the systematic nodes to which the data-collector does *not* connect. Then the data-collector is able to access the $k\alpha$ symbols

$$\begin{bmatrix} S_{(P,\text{all})} \\ \Phi_{(Q,\text{all})}(S+\rho S^t) \end{bmatrix}. \tag{59}$$

Thus, the data-collector has access to the i rows of S indexed by the entries of P and consequently, has access to the corresponding columns of S^t as well.

Consider the i columns of $\Phi_{(Q,\text{all})}(S+\rho S^t)$ indexed by P. Since the entries of these columns in S^t are known, the data-collector has access to $\Phi_{(Q,\text{all})}S_{(\text{all},P)}$. Now since the i rows of S indexed through P are also known, the data-collector has thus access to the product

$$\Phi_{(Q,T)}S_{(T,P)}. \tag{60}$$

Now as $\Phi_{(Q,T)}$ is nonsingular, being a $(k-|P|, k-|P|)$ submatrix of a Cauchy matrix, the data-collector can recover $S_{(T,P)}$. In this way, the data-collector has recovered all the entries in the rows of S indexed by P, as well as all the entries in the columns of S indexed by P. Clearly, the same statement holds when S is replaced by S^t. Thus the data-collector has access to the product:

$$\Phi_{(Q,T)}(S+\rho S^t)_{(T,T)}. \tag{61}$$

Again, $\Phi_{(Q,T)}$ is nonsingular, and this enables the data-collector to recover $(S+\rho S^t)_{(T,T)}$. It is easy to see that since $\rho^2 \neq 1$, from the diagonal elements of this matrix, all the diagonal elements of $S_{(T,T)}$ can be obtained. The nondiagonal elements are however of the form $S_{lj}+\rho S_{jl}$ and $S_{jl}+\rho S_{lj}$ for $l \in T$, $j \in T$, $l \neq j$. Again since $\rho^2 \neq 1$, all the nondiagonal elements of $S_{(T,T)}$ can also be decoded. In this way, the data-collector has recovered all the B entries of S. ■

Appendix B
Equivalent Codes and Conversion of Nonsystematic Codes to Systematic

In this section, we define the notion of "equivalent codes", and show that any exact-regenerating code is equivalent to a systematic exact-regenerating code.

Given any linear exact-regenerating code, one can express each of the $n\alpha$ symbols stored in the nodes as a linear combination of the B message symbols $\{u_i\}_{i=1}^B$. Let $\{c_{ij} | 1 \leq i \leq n, 1 \leq j \leq \alpha\}$ denote the jth symbol stored in the ith node. Thus, we have the relation:

$$\begin{aligned}&[u_1\, u_2 \cdots u_B][G_1\, G_2 \cdots G_n] \\ &\quad = [c_{11}\cdots c_{1\alpha}|c_{21}\cdots c_{2\alpha}|\cdots\cdots|c_{n1}\cdots c_{n\alpha}]\end{aligned} \tag{62}$$

where the $(B \times n\alpha)$ block generator matrix $G = [G_1\, G_2 \cdots G_n]$ is composed of the n component generator submatrices

$$G_i = [\,\underline{g}_{i1} \quad \underline{g}_{i2} \quad \cdots \quad \underline{g}_{i\alpha}\,]$$

each of size $(B \times \alpha)$, and associated to a distinct node.[5] Let W_i denote the column-space of G_i. A little thought will show that a distributed storage code is an exact-regenerating code iff:

1) for every subset of k nodes $\{i_j \mid 1 \leq j \leq k\}$,

$$\dim(W_{i_1} + W_{i_2} + \cdots + W_{i_k}) = B;$$

and

2) for every subset of $(d+1)$ nodes $\{i_j \mid 1 \leq j \leq (d+1)\}$, the subspaces $\{W_{i_j}\}_{j=1}^d$ contain a vector $\underline{w}_j$ such that

$$W_{i_{d+1}} \subseteq \text{span}\left(\underline{w}_{i_1}, \underline{w}_{i_2}, \ldots, \underline{w}_{i_d}\right).$$

[5]In the terminology of network coding, the $(B \times 1)$ column vector $\underline{g}_{ij}$ is termed the jth global kernel associated to the ith node.

We can thus define two exact-regenerating codes to be *equivalent* if the associated subspaces $\{W_i\}_{i=1}^n$ are identical. It is also clear that two codes are equivalent if one can be obtained from the other through a nonsingular transformation of the message symbols and the symbols stored within the nodes. With these two observations, it follows that two codes with generator matrices having the following relation are equivalent:

$$G, \text{ and } XG \begin{bmatrix} Y_1 & & & \\ & Y_2 & & \\ & & \ddots & \\ & & & Y_n \end{bmatrix}$$

where the $(B \times B)$ pre-multiplication matrix X, and the $(n\alpha \times n\alpha)$ post-multiplication block diagonal matrix comprising of the $(\alpha \times \alpha)$ matrices $\{Y_i\}_{i=1}^n$, are nonsingular. Clearly, equivalent codes have identical data-reconstruction and regeneration properties.

Systematic Version of Exact-Regenerating Codes: It also follows that any exact-regenerating code is equivalent to a systematic, exact-regenerating code. To see this, suppose the set of k nodes to be systematic are the first k nodes. Let

$$\{\underline{\tilde{g}}_{a_1}, \underline{\tilde{g}}_{a_2}, \ldots, \underline{\tilde{g}}_{a_B}\} \subseteq \{\underline{g}_{ij} \mid 1 \leq i \leq k,\ 1 \leq j \leq \alpha\}$$

denote a set of B linearly independent column vectors drawn from the generator matrices of the first k nodes $[G_1 \ \cdots \ G_k]$. That such a subset is guaranteed to exist follows from the data-reconstruction property of a regenerating code. Let $\tilde{G}$ be the $(B \times B)$ invertible matrix

$$\tilde{G} = \left[\, \underline{\tilde{g}}_{a_1} \quad \underline{\tilde{g}}_{a_2} \quad \cdots \quad \underline{\tilde{g}}_{a_B} \,\right].$$

Then we have the relation

$$[\, u_1 \quad u_2 \quad \cdots \quad u_B \,]\tilde{G} = [\tilde{c}_{a_1} \ \tilde{c}_{a_2} \ \cdots \ \tilde{c}_{a_B}] \tag{63}$$

where $\{\tilde{c}_{a_i}\}_{i=1}^B$ is the corresponding set of code symbols. It follows that if we wish to encode in such a way that the code is systematic with respect to code symbols $\{\tilde{c}_{a_i}\}_{i=1}^B$, the input to be "fed" to the generator matrix G is

$$[\, u_1 \quad u_2 \quad \cdots \quad u_B \,]\tilde{G}^{-1}.$$

Appendix C
Interference Alignment in the Product-Matrix MSR Code

The concept of *interference alignment* was introduced in [21] and [22] in the context of wireless communication. This concept was subsequently used to construct regenerating codes in [6], [7], [11], [14]. Furthermore, [6], [14] showed that interference alignment is in fact, a necessary ingredient of any linear MSR code. Since the product-matrix MSR construction provided in the present paper does not explicitly use the concept of interference alignment, a natural question that arises is how does interference alignment manifest itself in this code. We answer this question in the present section.

Consider repair of a failed node (say, node f) in a distributed storage system employing an MSR code, and let nodes $\{1, \ldots, d\}$ be the set of d helper nodes. Recall that [from (3)], at the MSR point we have $B = k\alpha$. Further, since all the B message symbols should be recoverable from any subset of k nodes, it must be that any subset of k nodes does not store any redundant information. Let $\underline{c}_i$, $1 \leq i \leq n$, be an α-length vector denoting the α symbols stored in node i. Then, from the above argument, it is clear that any symbol in the system can be written as a linear combination of the B symbols in $\{\underline{c}_f, \underline{c}_1, \ldots, \underline{c}_{(k-1)}\}$.

Let θ_ℓ, $k \leq \ell \leq d$, denote the symbol passed by node ℓ to assist in the repair of node f. Then we can write

$$\theta_\ell = \underline{c}_f^t \, \underline{v}_{\ell,f} + \sum_{i=1}^{k-1} \underline{c}_i^t \, \underline{v}_{\ell,i} \tag{64}$$

for some vectors $\underline{v}_{\ell,i}$ and $\underline{v}_{\ell,f}$ each of length α. The symbols in $\{\underline{c}_f, \underline{c}_1, \ldots, \underline{c}_{(k-1)}\}$ have no redundancy among themselves. Thus, the components comprising of $\{\underline{c}_1, \ldots, \underline{c}_{(k-1)}\}$ are undesired and hence are termed as *interference components*, and the component comprising of $\underline{c}_f$ is termed the *desired component*.

It is shown in [6], [14] that for any MSR code, it must be that for every $i \in \{1, \ldots, k-1\}$, the set of vectors

$$\left\{\underline{v}_{\ell,i} | k \leq \ell \leq d\right\} \tag{65}$$

are *aligned* (i.e., are scalar multiples of each other).

The following lemma considers the repair scenario discussed above to illustrate how interference alignment arises in the product-matrix MSR code presented in Section V.

Lemma 11: For every helper node ℓ, $k \leq \ell \leq d$, there exist scalars $\{a_{\ell,i} | 1 \leq i \leq k-1\}$ and an α-length vector $\underline{b}_\ell = [b_{\ell,1} \cdots b_{\ell,\alpha}]^t$ such that

$$\underline{\psi}_\ell^t M \underline{\phi}_f = \underline{\psi}_f^t M \underline{b}_\ell + \sum_{i=1}^{k-1} a_{\ell,i} \underline{\psi}_i^t M \underline{\phi}_f. \tag{66}$$

Proof: Rewriting the symbols passed by the helper node $j (1 \leq j \leq d)$

$$\underline{\psi}_j^t M \underline{\phi}_f = \left[\underline{\phi}_j^t S_1 + \lambda_j \underline{\phi}_j^t S_2\right] \underline{\phi}_f \tag{67}$$

$$= \left[\underline{\phi}_f^t S_1 + \lambda_j \underline{\phi}_f^t S_2\right] \underline{\phi}_j \tag{68}$$

$$= \left[\underline{\phi}_f^t S_1 + \lambda_f \underline{\phi}_f^t S_2\right] \underline{\phi}_j + (\lambda_j - \lambda_f) \underline{\phi}_f^t S_2 \underline{\phi}_j \tag{69}$$

$$= \underline{\psi}_f^t M \underline{\phi}_j + (\lambda_j - \lambda_f) \underline{\phi}_f^t S_2 \underline{\phi}_j \tag{70}$$

where (68) follows from the symmetry of matrices S_1 and S_2. By construction, the values of the scalars $\{\lambda_j \mid 1 \leq j \leq n\}$ are distinct, which allows us to write

$$\underline{\phi}_f^t S_2 \underline{\phi}_j = (\lambda_j - \lambda_f)^{-1} \left(\underline{\psi}_j^t M \underline{\phi}_f - \underline{\psi}_f^t M \underline{\phi}_j\right). \tag{71}$$

Also, since the $(k-1=)\alpha$-length vectors $\{\underline{\phi}_i \mid 1 \leq i \leq k-1\}$ are linearly independent by construction, for $k \leq \ell \leq d$, there exist scalars $\{\tilde{a}_{\ell,i} \mid 1 \leq i \leq k-1\}$ such that

$$\underline{\phi}_\ell = \sum_{i=1}^{k-1} \tilde{a}_{\ell,i} \underline{\phi}_i. \tag{72}$$

From (70), (71), and (72), for any $\ell \in \{k, \ldots, d\}$, we can write

$$\underline{\psi}_\ell^t M \underline{\phi}_f = \underline{\psi}_f^t M \underline{\phi}_\ell + (\lambda_\ell - \lambda_f)\underline{\phi}_f^t S_2 \underline{\phi}_\ell \tag{73}$$

$$= \underline{\psi}_f^t M \underline{\phi}_\ell + (\lambda_\ell - \lambda_f) \sum_{i=1}^{k-1} \tilde{a}_{\ell,i} \underline{\phi}_f^t S_2 \underline{\phi}_i \tag{74}$$

$$= \underline{\psi}_f^t M \left(\underline{\phi}_\ell - (\lambda_\ell - \lambda_f) \sum_{i=1}^{k-1} \tilde{a}_{\ell,i} (\lambda_i - \lambda_f)^{-1} \underline{\phi}_i \right) + \sum_{i=1}^{k-1} \left(\tilde{a}_{\ell,i} (\lambda_\ell - \lambda_f)(\lambda_i - \lambda_f)^{-1} \right) \left(\underline{\psi}_i^t M \underline{\phi}_f \right) \tag{75}$$

where (74) follows from (72), and (75) follows from (71). ■

REFERENCES

[1] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, Chicago, IL, Jun. 1988, pp. 109–116.

[2] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, "Pond: The OceanStore prototype," in *Proc. 2nd USENIX Conf. File and Storage Technologies (FAST)*, 2003, pp. 1–14.

[3] R. Bhagwan, K. Tati, Y. C. Cheng, S. Savage, and G. M. Voelker, "Total recall: System support for automated availability management," in *Proc. 1st Conf. Networked Systems Design and Implementation (NSDI)*, 2004.

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

[5] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. 45th Annu. Allerton Conf. Control, Computing, and Communication*, Urbana-Champaign, IL, Sep. 2007.

[6] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, Jan. 2010.

[7] C. Suh and K. Ramchandran, "Exact-repair MDS codes for distributed storage using interference alignment," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Austin, TX, Jun. 2010, pp. 161–165.

[8] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE J. Select. Areas Commun.*, vol. 28, no. 2, pp. 277–288, Feb. 2010.

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

[10] A. Duminuco and E. Biersack, "A practical study of regenerating codes for peer-to-peer backup systems," in *Proc. 29th IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, Jun. 2009, pp. 376–384.

[11] Y. Wu and A. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Seoul, South Korea, Jul. 2009, pp. 2276–2280.

[12] 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. Communication, Control, and Computing*, Urbana-Champaign, IL, Sep. 2009, pp. 1243–1249.

[13] D. Cullina, A. G. Dimakis, and T. Ho, "Searching for minimum storage regenerating codes," in *Proc. 47th Annu. Allerton Conf. Communication, Control, and Computing*, Urbana-Champaign, IL, Sep. 2009.

[14] 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*, submitted for publication.

[15] Y. Wu, "A construction of systematic MDS codes with minimum repair bandwidth," *IEEE Trans. Inf. Theory*, submitted for publication.

[16] V. R. Cadambe, S. A. Jafar, and H. Maleki, Distributed Data Storage with Minimum Storage Regenerating Codes—Exact and Functional Repair are Asymptotically Equally Efficient [Online]. Available: arXiv:1004.4299 [cs.IT]

[17] C. Suh and K. Ramchandran, On the Existence of Optimal Exact-Repair MDS Codes for Distributed Storage [Online]. Available: arXiv:1004.4663 [cs.IT]

[18] N. B. Shah, K. V. Rashmi, and P. V. Kumar, "A flexible class of regenerating codes for distributed storage," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Austin, TX, Jun. 2010, pp. 1943–1947.

[19] 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 Trans. Inf. Theory*, submitted for publication.

[20] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas With Application to Linear Systems Theory*. Princeton, NJ: Princeton University Press, 2005.

[21] M. Maddah-Ali, A. Motahari, and A. Khandani, "Communication over MIMO X channels: Interference alignment, decomposition, and performance analysis," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3457–3470, Aug. 2008.

[22] V. Cadambe and S. Jafar, "Interference alignment and spatial degrees of freedom for the k user interference channel," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3425–3441, Aug. 2008.

K. V. Rashmi received the M.E. degree from the Indian Institute of Science (IISc), Bangalore, in 2010.

Her research interests include coding theory, information theory, networks, communications and signal processing, with a current focus on coding for data storage networks and network coding.

Nihar B. Shah received the M.E. degree from the Indian Institute of Science (IISc), Bangalore, in 2010.

His research interests include coding and information theory, algorithms, and statistical inference.

Mr. Shah is a recipient of the Prof. S.V.C. Aiya Medal for the best master-of-engineering student in the ECE Department at IISc, 2010.

P. Vijay Kumar (S'80–M'82–SM'01–F'02) received the B.Tech. and M.Tech. degrees from the Indian Institutes of Technology (Kharagpur and Kanpur) ,and the Ph.D. degree from the University of Southern California (USC) in 1983, all in electrical engineering.

From 1983 to 2003, he was on the faculty of the EE-Systems Department at USC. Since 2003, he has been on the faculty of the Indian Institute of Science, Bangalore, and also holds the position of adjunct research professor at USC. His current research interests include codes for distributed storage, distributed function computation, sensor networks and space-time codes for MIMO and cooperative communication networks.

Dr. Kumar is a an ISI highly-cited author. He is co-recipient of the 1995 IEEE Information Theory Society prize paper award as well as of a best paper award at the DCOSS 2008 conference on sensor networks.