Update-Efficient Error-Correcting Product-Matrix Codes

Yunghsiang S. Han, *Fellow*, *IEEE*, Hung-Ta Pai, *Senior Member. IEEE*, Rong Zheng, *Senior Member. IEEE*, Pramod K. Varshney, *Fellow*, *IEEE*

***Abstract*—Regenerating codes provide an efficient way to recover data at failed nodes in distributed storage systems. It has been shown that regenerating codes can be designed to minimize the per-node storage (called MSR) or minimize the communication overhead for regeneration (called MBR). In this work, we propose new encoding schemes for error-correcting MSR and MBR codes that generalize our earlier results on error-correcting regenerating codes. General encoding schemes for product-matrix MSR and MBR codes are derived such that the encoder based on Reed-Solomon (RS) codes is no longer limited to the Vandermonde matrix proposed earlier. Furthermore, MSR codes and MBR codes with the least update complexity can be found. A decoding scheme is proposed that utilizes RS codes to perform data reconstruction for MSR codes. The proposed decoding scheme has better error correction capability and incurs least number of node accesses when errors are present. A new decoding scheme is also proposed for MBR codes that is more capable and can correct more error-patterns. Simulation results are presented that exhibit the superior performance of the proposed schemes.**

***Index Terms*—Distributed storage, Regenerating codes, Reed-Solomon codes, Decoding, Product-Matrix codes**

I. Introduction

Cloud storage is gaining popularity as an alternative to enterprise storage. In cloud storage, data is stored in virtualized pools of storage typically hosted by third-party data centers. Reliability is a key challenge in the design of distributed storage systems that provide cloud storage. Both crash-stop and Byzantine failures (as a result of software bugs and malicious attacks) are likely to be present during data retrieval. A crash-stop failure makes a storage node unresponsive to access requests. In contrast, a Byzantine failure responds to access requests with erroneous data. To achieve better reliability, one common approach is to replicate data files on multiple storage nodes in a network. There are two kinds of approaches: duplication (Google) [1] and erasure coding [2], [3]. In the duplication approaches, an exact copy of each data is stored at multiple storage nodes, thus requiring lots of storage space. The advantage of this type of approaches is that only one storage node needs to be accessed to obtain the original data. In contrast, in the second category of approaches, erasure coding is employed to encode the original data and then the encoded data is distributed to storage nodes. Typically, multiple storage nodes need to be accessed to recover the original data. One popular class of erasure codes is the maximum-distance-separable (MDS) codes. With $[n, k]$ MDS codes such as Reed-Solomon (RS) codes, k data items are encoded and then distributed to and stored at n storage nodes. A user or a data collector can retrieve the original data by accessing *any* k of the storage nodes, a process referred to as *data reconstruction*.

Any storage node can fail due to hardware or software faults. Data stored at the failed nodes need to be recovered (regenerated) to remain functional to perform data reconstruction. The process to recover the stored (encoded) data at a storage node is called *data regeneration*. A simple way for data regeneration is to first reconstruct the original data and then recover the data stored at the failed node. However, it is not efficient to retrieve the entire B symbols of the original file to recover a much smaller fraction of data stored at the failed node. *Regenerating codes*, first introduced in the pioneering works by Dimakis *et al.* in [4], [5], allow efficient data regeneration. To facilitate data regeneration, each storage node stores α symbols and any d surviving nodes can be accessed to retrieve $\beta \leq \alpha$ symbols from each node. A trade-off exists between the storage overhead and the regeneration (repair) bandwidth needed for data regeneration. Minimum Storage Regenerating (MSR) codes first minimize the amount of data stored per node, and then the repair bandwidth, while Minimum Bandwidth Regenerating (MBR) codes carry out the minimization in the reverse order. There have been many works that focus on the design of regenerating codes [6]–[15]. There are two categories of approaches to regenerate data at a failed node. If the replacement data is exactly the same as that previously stored at the failed node, we call it *exact regeneration*. Otherwise, if the replacement data only guarantees the correctness of data reconstruction and regeneration properties, it is called *functional regeneration*. In practice, exact regeneration is more desirable since there is no need to inform each node in the network regarding the replacement. Furthermore, it is easy to keep the codes systematic via exact regeneration, where partial data can be retrieved without accessing all k nodes. It has been proved that no linear code performing exact regeneration can achieve the MSR point for any $[n, k, d < 2k-3]$ when β is normalized to 1 [16]. However, when B approaches infinity, this is achievable for any $k \leq d \leq n-1$ [17]. In this work, we only consider

Part of this work was presented at the IEEE International Symposium on Information Theory (ISIT 2013).

Han is with the Dept. of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan (e-mail: yshan@mail.ntust.edu.tw), Pai is with the Dept. of Communication Engineering, National Taipei University, Taiwan, R.O.C., Zheng is with the Dept. of Computing and Software, McMaster University, Hamilton, ON, Canada, and Varshney is with the Dept. of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY USA.

exact regeneration.

There are several existing code constructions of regenerating codes for exact regeneration [9], [13], [17], [18]. In [9], Wu and Dimakis apply ideas from interference alignment [19], [20] to construct the codes for $n = 4$ and $k = 2$. The idea was extended to the more general case of $k < \max\{3, n/2\}$ in [18]. In [13], Rashmi *et al.* used product-matrix construction to design optimal $[n, k, d \geq 2k - 2]$ MSR codes and $[n, k, d]$ MBR codes for exact regeneration. These constructions of exact-regenerating codes are the first for which the code length n can be chosen independently of other parameters. However, only crash-stop failures of storage nodes are considered in [13]. Recently, a decentralized minimum-cost repair scheme for crash-stop failures was proposed in [21] by Gerami *et. al*. In [21], optimal-cost repair is formulated as a convex optimization problem for networks with convex transmission costs and is solved via dual decomposition.

The problem of the security of regenerating codes was considered in [11], [12], and [22]–[25]. In [11], the security problem in the presence of eavesdropping and adversarial attack during the data reconstruction and regeneration processes was considered. Upper bounds on the maximum amount of information that can be stored safely were derived. Pawar *et al.* [11] also gave an explicit code construction for $d = n-1$ in the bandwidth-limited regime. The problem of Byzantine fault tolerance for regenerating codes was considered in [12]. Oggier and Datta [12] investigated the resilience of regenerating codes when supporting multi-repairs. By collaboration among newcomers, they derived upper bounds on the resilience capability of regenerating codes. Kurihara and Kuwakado [24] analyzed the secrecy capacity of MBR codes, and proposed $[n, k, d, m]$ secure regenerating codes that prevent information leakage even when an eavesdropper is able to acquire data of m storage nodes or repaired data for m failed nodes. Our work deals with Byzantine failures for product-matrix regenerating codes and it does not need to have multiple newcomers to recover the data in the presence of failures.

Based on the code construction in [13], Han *et al.* extended the work of Rashmi *et al.* [13] to provide decoding algorithms that can handle Byzantine failures [22]. In [22], decoding algorithms for both MSR and MBR error-correcting product-matrix codes were provided. In particular, the decoding algorithm for an $[n, k, d]$ MBR code given in [22], [25] has the error correction capability of $\lfloor \frac{n-k+1}{2} \rfloor = \frac{n-k}{2}$ since $n - k$ is even. In [23], the code capability and resilience were discussed for error-correcting regenerating codes. Rashmi, *et al.* [23] proved that it is possible to decode an $[n, k, d]$ MBR code up to $\lfloor \frac{n-k}{2} \rfloor$ errors. The authors also claimed that any $[n, k, d \geq 2k - 2]$ MSR code can be decoded up to $\lfloor \frac{n-k}{2} \rfloor$ errors. However no explicit decoding (data reconstruction) procedure was provided due to which these codes cannot be used in practice. Thus, one contribution of this paper is to present a decoding algorithm for MSR codes.

In addition to bandwidth efficiency and error correction capability, another desirable feature for regenerating codes is *update complexity* [26], defined as the number of nonzero elements in the row of the encoding matrix with the maximum Hamming weight.[1] The smaller the number, the lower is the update complexity. Low update complexity is desirable in scenarios where updates are frequent.

One drawback of the decoding algorithms for MSR codes given in [22] is that, when one or more storage nodes have erroneous data, the decoder needs to access extra data from many storage nodes (at least k more nodes) for data reconstruction. Furthermore, when one symbol in the original data is updated, all storage nodes need to update their respective data. Thus, the MSR and MBR codes in [22] have the maximum possible update complexity. Both of these deficiencies are addressed in this paper. First, we propose a general encoding scheme for MSR codes. As a special case, least-update-complexity codes are designed. We also design least-update-complexity encoding matrix for the MBR codes by using the coefficients of generator polynomials of the $[n, k]$ and $[n, d]$ RS codes. The proposed codes not only have least update complexity but also have the smallest number of updated symbols when a single data symbol is modified. This is in contrast to the existing product-matrix codes. Second, a new decoding algorithm is presented for MSR codes. It not only exhibits better error correction capability but also incurs low communication overhead when errors occur in the accessed data. Third, we devise a decoding scheme for the MBR codes that can correct more error patterns compared to the one in [22].

The main contributions of this paper beyond the existing literature are as follows:

- General encoding schemes of product-matrix MSR and MBR codes are derived. The encoder based on RS codes is no longer limited to the Vandermonde matrix proposed in [13] and [22].
- MSR and MBR codes with systematic generator matrices of RS codes are provided. These codes have the least update complexity compared to that of existing codes such as systematic MSR and MBR codes proposed by Rashmi *et al.* [13].
- A detailed decoding algorithm for data construction of MSR codes is provided. It is non-trivial to extend the decoding procedure given in [13] to handle errors. The difficulty arises from the fact that an error in the received data will propagate to many places during the decoding process in [13].
- The decoding algorithm of MBR codes that can decode beyond the error-correction capability for some error patterns is also presented. This decoding algorithm can correct up to

$$\frac{n-k}{2} + \left\lfloor \frac{n-k+1-\lfloor \frac{n-k+1}{2} \rfloor}{2} \right\rfloor$$

  errors, though not all error patterns up to this number of errors can be corrected.

The rest of this paper is organized as follows. Section II gives an overview of error-correcting regenerating codes. Section III presents the least-update-complexity encoding and de-

[1]The update complexity defined in [26] is not equivalent to the maximum number of encoded symbols that must be updated while a single data symbol is modified.

coding schemes for error-correcting MSR regenerating codes. Section IV describes the least-update-complexity encoding of MBR codes and the corresponding decoding scheme. Section V details the evaluation results for the proposed decoding schemes. Section VI concludes the paper with a discussion on potential future work. Since only error-correcting regenerating codes are considered in this work, unless stated otherwise, we refer to error-correcting MSR and MBR codes as MSR and MBR codes in the rest of the paper.

II. Error-Correcting Product-Matrix Regenerating Codes

In this section, we give a brief overview of regenerating codes, and the MSR and MBR product-matrix code constructions presented in [13].

A. Regenerating Codes

Let α be the number of symbols stored at each storage node and $\beta \leq \alpha$ be the number of symbols downloaded from each storage node during regeneration. To repair the stored data at the failed node, a helper node accesses d surviving nodes. The design of regenerating codes ensures that the total regenerating bandwidth be much less than that of the original data, B. A regenerating code must be capable of reconstructing the original data symbols and regenerating coded data at a failed node. An $[n, k, d]$ regenerating code requires at least k nodes to ensure successful data reconstruction, and d surviving nodes to perform regeneration [13], where n is the number of storage nodes and $k \leq d \leq n-1$.

The cut-set bound given in [5], [6] provides a constraint on the repair bandwidth. Based on this bound, any regenerating code must satisfy the following inequality:

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

From (1), α or β can be minimized achieving either the minimum storage requirement or the minimum repair bandwidth requirement, but not both. The two extreme points in (1) are referred to as the minimum storage regeneration (MSR) and minimum bandwidth regeneration (MBR) points, respectively. The values of α and β for the MSR point can be obtained by first minimizing α and then minimizing β:

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

where we normalize β and set it equal to 1.[2] Reversing the order of minimization we have α for MBR as

$$\alpha = d, \ B = kd - k(k-1)/2 \ , \tag{3}$$

while $\beta = 1$.

[2]It has been proved that when designing $[n, k, d]$ MSR codes for $k/(n+1) \leq 1/2$, it suffices to consider those with $\beta = 1$ [13].

B. Product-Matrix MSR Codes With Error Correction Capability

Next, we describe the MSR code construction originally given in [13] and adopted later in [22]. Here, we assume $d = 2\alpha$.[3] The information sequence $\boldsymbol{m} = [m_0, m_1, \ldots, m_{B-1}]$ can be arranged into an information vector $U = [Z_1 Z_2]$ with size $\alpha \times d$ such that Z_1 and Z_2 are symmetric matrices with dimension $\alpha \times \alpha$. An $[n, d = 2\alpha]$ RS code is adopted to construct the MSR code [13]. Let a be a generator of $GF(2^m)$. In the encoding of the MSR code, we have

$$U \cdot G = C, \tag{4}$$

where

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{d-1} & (a^1)^{d-1} & \cdots & (a^{n-1})^{d-1} \end{bmatrix},$$

and C is the codeword vector with dimension $(\alpha \times n)$.

It is possible to rewrite generator matrix G of the RS code as,

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{\alpha-1} & (a^1)^{\alpha-1} & \cdots & (a^{n-1})^{\alpha-1} \\ (a^0)^{\alpha}1 & (a^1)^{\alpha}1 & \cdots & (a^{n-1})^{\alpha}1 \\ (a^0)^{\alpha}a^0 & (a^1)^{\alpha}a^1 & \cdots & (a^{n-1})^{\alpha}a^{n-1} \\ (a^0)^{\alpha}(a^0)^2 & (a^1)^{\alpha}(a^1)^2 & \cdots & (a^{n-1})^{\alpha}(a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{\alpha}(a^0)^{\alpha-1} & (a^1)^{\alpha}(a^1)^{\alpha-1} & \cdots & (a^{n-1})^{\alpha}(a^{n-1})^{\alpha-1} \end{bmatrix} \tag{5}$$

$$= \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix}, \tag{6}$$

where $\bar{G}$ contains the first α rows in G, and Δ is a diagonal matrix with $(a^0)^{\alpha}, \ (a^1)^{\alpha}, \ (a^2)^{\alpha}, \ldots, \ (a^{n-1})^{\alpha}$ as diagonal elements, namely,

$$\Delta = \begin{bmatrix} (a^0)^{\alpha} & 0 & 0 & \cdots & 0 & 0 \\ 0 & (a^1)^{\alpha} & 0 & \cdots & 0 & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & \cdots & 0 & (a^{n-1})^{\alpha} \end{bmatrix}. \tag{7}$$

Note that if the RS code is over $GF(2^m)$ for $m \geq \lceil \log_2 n\alpha \rceil$, then it can be shown that $(a^0)^{\alpha}, \ (a^1)^{\alpha}, \ (a^2)^{\alpha}, \ldots, \ (a^{n-1})^{\alpha}$ are all distinct. According to the encoding procedure, the α symbols stored at storage node i are given by,

$$U \cdot \begin{bmatrix} \boldsymbol{g}_i^T \\ (a^{i-1})^{\alpha}\boldsymbol{g}_i^T \end{bmatrix} = Z_1 \boldsymbol{g}_i^T + (a^{i-1})^{\alpha} Z_2 \boldsymbol{g}_i^T,$$

where $\boldsymbol{g}_i^T$ is the ith column in $\bar{G}$.

[3]An elegant method to extend the construction of $d > 2\alpha$ based on the construction of $d = 2\alpha$ has been given in [13]. Since the same technique can be applied to the code constructions proposed in this work, it is omitted here.

C. Product-Matrix MBR Codes With Error Correction Capability

In this section, we describe the MBR code constructed in [13] and reformulated later in [22]. Note that at the MBR point, $\alpha = d$. Let the information sequence $\boldsymbol{m} = [m_0, m_1, \ldots, m_{B-1}]$ be arranged into an information vector U with size $\alpha \times d$, where

$$U = \begin{bmatrix} A_1 & A_2^T \\ A_2 & \mathbf{0} \end{bmatrix}, \tag{8}$$

A_1 is a $k \times k$ symmetric matrix, A_2 a $(d-k) \times k$ matrix, $\mathbf{0}$ is the $(d-k) \times (d-k)$ zero matrix. Note that both A_1 and U are symmetric. It is clear that U has a dimension $d \times d$ (or $\alpha \times d$). An $[n, d]$ RS code is chosen to encode each row of U. The generator matrix of the RS code is given as

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{k-1} & (a^1)^{k-1} & \cdots & (a^{n-1})^{k-1} \\ (a^0)^k & (a^1)^k & \cdots & (a^{n-1})^k \\ & & \vdots & \\ (a^0)^{d-1} & (a^1)^{d-1} & \cdots & (a^{n-1})^{d-1} \end{bmatrix}, \tag{9}$$

where a is a generator of $GF(2^m)$. Let C be the codeword vector with dimension $(\alpha \times n)$. It can be obtained as

$$U \cdot G = C.$$

From (9), G can be divided into two sub-matrices as

$$G = \begin{bmatrix} G_k \\ S \end{bmatrix}, \tag{10}$$

where

$$G_k = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{k-1} & (a^1)^{k-1} & \cdots & (a^{n-1})^{k-1} \end{bmatrix} \tag{11}$$

and

$$S = \begin{bmatrix} (a^0)^k & (a^1)^k & \cdots & (a^{n-1})^k \\ & & \vdots & \\ (a^0)^{d-1} & (a^1)^{d-1} & \cdots & (a^{n-1})^{d-1} \end{bmatrix}.$$

It can be shown that G_k is a generator matrix of the $[n, k]$ RS code and it will be used in the decoding for data reconstruction.

III. Encoding and Decoding Schemes for Product-Matrix MSR Codes

In this section, we propose a new encoding scheme for $[n, d]$ error-correcting MSR codes. With a feasible matrix Δ, $\bar{G}$ in (6) can be any generator matrix of the $[n, \alpha]$ RS code. The code construction in [13], [22] is thus a special case of our proposed scheme. We can also select a suitable generator matrix such that the update complexity of the resulting code is minimized. A decoding scheme is then proposed that uses the subcode of the $[n, d]$ RS code, the $[n, \alpha = k-1]$ RS code generated by $\bar{G}$, to perform the data reconstruction.

A. Encoding Schemes for Error-Correcting MSR Codes

RS codes are known to have very fast decoding algorithms and exhibit good error correction capability. From (6) in Section II-B, a generator matrix G for product-matrix MSR codes needs to satisfy:

1) $G = \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix}$, where $\bar{G}$ contains the first α rows in G and Δ is a diagonal matrix with distinct elements in the diagonal.
2) $\bar{G}$ is a generator matrix of the $[n, \alpha]$ RS code and G is a generator matrix of the $[n, d = 2\alpha]$ RS code.

Next, we present a sufficient condition for $\bar{G}$ and Δ such that G is a generator matrix of an $[n, d]$ RS code. We first introduce some notations. Let $g_{0y}(x) = \prod_{i=0}^{n-y-1}(x - a^i)$ and the $[n, y]$ RS code generated by $g_{0y}(x)$ be C_{0y}. Similarly, let $g_{1y}(x) = \prod_{i=1}^{n-y}(x - a^i)$ and the $[n, y]$ RS code generated by $g_{1y}(x)$ be C_{1y}. Clearly, $a^0, a^1, a^2, \ldots, a^{n-y-1}$ are roots of $g_{0y}(x)$, and $a^1, a^2, \ldots, a^{n-y}$ are roots of $g_{1y}(x)$. It can be shown that, by the Mattson-Solomon polynomial [27], we can choose $\bar{G}_{0y} = \begin{bmatrix} \bar{\boldsymbol{g}}_0 & \bar{\boldsymbol{g}}_1 & \cdots & \bar{\boldsymbol{g}}_{n-1} \end{bmatrix}$, where $\bar{\boldsymbol{g}}_i = [(a^i)^1, (a^i)^2, \ldots, (a^i)^y]^T$, as the generator matrix of C_{0y}. Similarly, we can choose $\bar{G}_{1y} = \begin{bmatrix} \tilde{\boldsymbol{g}}_0 & \tilde{\boldsymbol{g}}_1 & \cdots & \tilde{\boldsymbol{g}}_{n-1} \end{bmatrix}$, where $\tilde{\boldsymbol{g}}_i = [(a^i)^0, (a^i)^1, \ldots, (a^i)^{y-1}]^T$, as the generator matrix of C_{1y}. Note that, C_{0y} and C_{1y} have the same weight distribution.

Theorem 1: Let $\bar{G}$ be a generator matrix of the $[n, \alpha]$ RS code $C_{0\alpha}$. Let the diagonal elements of Δ be $b_0, b_1, \ldots, b_{n-1}$ such that $b_i \neq b_j$ for all $i \neq j$, and $(b_0, b_1, \ldots, b_{n-1})$ is a codeword in $C_{1(\alpha+1)}$ but not $C_{1\alpha}$. In other words, $(b_0, b_1, \ldots, b_{n-1}) \in C_{1(\alpha+1)} \backslash C_{1\alpha}$. Then, $G = \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix}$ is a generator matrix of the $[n, d]$ RS code C_{0d}.

Proof: See Appendix A. ∎

When the RS code is over $GF(2^m)$, we have the following results. The detailed proof can be found in [28].

Corollary 1: Under the condition that the RS code is over $GF(2^m)$ for $m \geq \lceil \log_2 n \rceil$ and $\gcd(2^m - 1, \alpha) = 1$, the diagonal elements of Δ, $b_0, b_1, \ldots, b_{n-1}$, can be chosen as

$$\gamma(a^0)^\alpha, \gamma(a)^\alpha, \gamma(a^2)^\alpha, \ldots, \gamma(a^{n-1})^\alpha,$$

where $\gamma \in GF(2^m) \backslash \{\mathbf{0}\}$.

It is important to note that by setting $\gamma = 1$ in Corollary 1, we obtain the generator matrix G given in (6) first proposed in [13], [22] as a special case.[4]

One advantage of the proposed scheme is that it can now operate on a smaller finite field than that of the scheme in [13], [22]. Another advantage is that one can choose $\bar{G}$ (and Δ accordingly) freely as long as $\bar{G}$ is the generator matrix of an $[n, \alpha]$ RS code. In particular, as discussed in Section I, to minimize the update complexity, it is desirable to choose

[4]Though the roots in G given in (6) are different from those for the proposed generator matrix, they generate RS codes with the same weight distribution.

a generator matrix that has the least row-wise maximum Hamming weight. Next, we present a least-update-complexity generator matrix that satisfies (6).

Corollary 2: Suppose Δ is chosen according to Corollary 1. Let $\bar{G}$ be the generator matrix associated with a systematic $[n, \alpha]$ RS code. That is, $\bar{G} = [B\ I]$, where

$$B = \begin{bmatrix} b_{00} & b_{01} & b_{02} & \cdots & b_{0(n-\alpha-1)} \\ b_{10} & b_{11} & b_{12} & \cdots & b_{1(n-\alpha-1)} \\ b_{20} & b_{21} & b_{22} & \cdots & b_{2(n-\alpha-1)} \\ & \vdots & & & \vdots \\ b_{(\alpha-1)0} & b_{(\alpha-1)1} & b_{(\alpha-1)2} & \cdots & b_{(\alpha-1)(n-\alpha-1)} \end{bmatrix} \quad (12)$$

I is the identity matrix,

$$x^{n-\alpha+i} = u_i(x)g(x) + b_i(x) \text{ for } 0 \leq i \leq \alpha - 1 \ ,$$

where $u_i(x)$ is the quotient and

$$b_i(x) = b_{i0} + b_{i1}x + \cdots + b_{i(n-\alpha-1)}x^{n-\alpha-1}$$

is the remainder when $x^{n-\alpha+i}$ is divided by $g(x)$. Note that both $x^{n-\alpha+i}$ and $g(x)$ are known before the construction of B so that the division is feasible to obtain $b_i(x)$. Then, $G = \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix}$ is a least-update-complexity generator matrix.

Proof: The result holds since each row of $\bar{G}$ is a nonzero codeword with the minimum Hamming weight $n - \alpha + 1$. ■

Note that the above claim for least-update-complexity is true with product-matrix construction only when $\bar{G}$ and G are chosen as the generator matrices of MDS codes.

The update complexity defined in [26] differs from the maximum number of encoded symbols that must be updated when a single data symbol is modified. For instance, when the modified data symbol is located on the diagonal of Z_1 or Z_2, $(n - \alpha + 1)$ encoded symbols need to be updated. Otherwise, in general, when two symbols in U are modified, $2(n-\alpha+1)$ encoded symbols need to be updated. Hence, the least update-complexity codes found previously do not necessarily imply the least number of encoded symbols to be updated for a single modified data symbol. If pre-processing on the information vector is allowed or $d \neq 2k - 2$, the number of encoded symbols to be updated in this case might be further reduced.

B. Decoding Scheme for MSR Codes

Unlike the decoding scheme in [22] that uses the $[n, d]$ RS code, we propose to use the subcode of the $[n, d]$ RS code, i.e., the $[n, \alpha = k - 1]$ RS code generated by $\bar{G}$, to perform data reconstruction. The advantages of using the $[n, k-1]$ RS code are two-fold. First, its error correction capability is higher. Specifically, it can tolerate $\lfloor \frac{n-k}{2} \rfloor$ instead of $\lfloor \frac{n-d}{2} \rfloor$ errors. Second, it only requires the access of two additional storage nodes (as opposed to $d - k + 2 = k$ nodes) for each extra error.

Without loss of generality, we assume that the data collector retrieves encoded symbols from $k + 2v$ ($v \geq 0$) storage nodes, $j_0, j_1, \ldots, j_{k+2v-1}$. We also assume that there are v storage nodes whose received symbols are erroneous. The stored information on the $k + 2v$ storage nodes are collected as the $k + 2v$ columns in $Y_{\alpha\times(k+2v)}$. The $k + 2v$ columns of G corresponding to storage nodes $j_0, j_1, \ldots, j_{k+2v-1}$ are denoted as the columns of G_{k+2v}. First, we discuss data reconstruction when $v = 0$. The decoding procedure is similar to that in [13].

No Error: In this case, $v = 0$ and there is no error in Y. Then,

$$\begin{aligned} Y_{\alpha\times k} &= UG_k \\ &= [Z_1 Z_2] \begin{bmatrix} \bar{G}_k \\ \bar{G}_k\Delta_k \end{bmatrix} \\ &= [Z_1\bar{G}_k + Z_2\bar{G}_k\Delta_k] \ . \end{aligned} \quad (13)$$

Multiplying $\bar{G}_k^T$ to both sides of (13), we have [13]

$$\begin{aligned} \bar{G}_k^T Y_{\alpha\times k} &= \bar{G}_k^T U G_k \\ &= [\bar{G}_k^T Z_1 \bar{G}_k + \bar{G}_k^T Z_2 \bar{G}_k \Delta_k] \\ &= P + Q\Delta_k \ . \end{aligned} \quad (14)$$

Since Z_1 and Z_2 are symmetric, P and Q are symmetric as well. The (i, j)th element of $P + Q\Delta_k$, $1 \leq i, j \leq k$ and $i \neq j$, is

$$p_{ij} + q_{ij}a^{(j-1)\alpha} \ , \quad (15)$$

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

$$p_{ji} + q_{ji}a^{(i-1)\alpha} \ . \quad (16)$$

Since $a^{(j-1)\alpha} \neq a^{(i-1)\alpha}$ for all $i \neq j$, $p_{ij} = p_{ji}$, and $q_{ij} = q_{ji}$, combining (15) and (16), the values of p_{ij} and q_{ij} can be obtained. Note that we only obtain $k - 1$ values for each row of P and Q since no elements in the diagonal of P or Q are obtained.

To decode P, recall that $P = \bar{G}_k^T Z_1 \bar{G}_k$. P can be treated as a portion of the codeword vector, $\bar{G}_k^T Z_1 \bar{G}$. By the construction of $\bar{G}$, it is easy to see that $\bar{G}$ is a generator matrix of the $[n, k-1]$ RS code. Hence, each row in the matrix $\bar{G}_k^T Z_1 \bar{G}$ is a codeword. Since we know $k-1$ components in each row of P, it is possible to decode $\bar{G}_k^T Z_1 \bar{G}$ by the error-and-erasure decoder of the $[n, k-1]$ RS code.[5]

Since one cannot locate any erroneous position from the decoded rows of P, the decoded α codewords are accepted as $\bar{G}_k^T Z_1 \bar{G}$. By collecting the last α columns of $\bar{G}$ as $\bar{G}_\alpha$ to find its inverse (here it is an identity matrix), one can recover $\bar{G}_k^T Z_1$ from $\bar{G}_k^T Z_1 \bar{G}$. Since any α rows in $\bar{G}_k^T$ are independent and thus invertible, we can pick any α of them to recover Z_1. Z_2 can be obtained similarly by Q.

It is not trivial to extend the above decoding procedure to the case of errors. The difficulty arises from the fact that for any error in $Y_{\alpha\times n}$, this error will propagate into many places in P and Q, due to operations involved in (14), (15), and (16), such that many of their rows cannot be decoded successfully or correctly (Please refer to Lemma 1). In the following, we present the procedure to locate erroneous columns in Y based on RS decoder.

[5]The error-and-erasure decoder of an $[n, k-1]$ RS code can successfully decode a received vector if $s + 2v < n - k + 2$, where s is the number of erasure (no symbol) positions, v is the number of errors in the received portion of the received vector, and $n - k + 2$ is the minimum Hamming distance of the $[n, k-1]$ RS code.

Multiple Errors: Before presenting the proposed decoding algorithm, we first prove that a decoding procedure can always successfully decode Z_1 and Z_2 if $v \leq \lfloor \frac{n-k}{2} \rfloor$ and all storage nodes are accessed. Assume the storage nodes with errors correspond to the ℓ_0th, ℓ_1th, ..., ℓ_{v-1}th columns in the received matrix $Y_{\alpha \times n}$. Then,

$$\begin{aligned} & \bar{G}^T Y_{\alpha \times n} \\ = \; & \bar{G}^T U G + \bar{G}^T E \\ = \; & \bar{G}^T [Z_1 Z_2] \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix} + \bar{G}^T E \\ = \; & [\bar{G}^T Z_1 \bar{G} + \bar{G}^T Z_2 \bar{G} \Delta] + \bar{G}^T E \ , \end{aligned} \tag{17}$$

where

$$E = \left[\mathbf{0}_{\alpha \times (\ell_0 - 1)} | \boldsymbol{e}_{\ell_0}^T | \mathbf{0}_{\alpha \times (\ell_1 - \ell_0 - 1)} | \cdots | \boldsymbol{e}_{\ell_{v-1}}^T | \mathbf{0}_{\alpha \times (n - \ell_{v-1})} \right] \ .$$

Lemma 1: There are at least $n-k+2$ errors in each of the ℓ_0th, ℓ_1th, ..., ℓ_{v-1}th columns of $\bar{G}^T Y_{\alpha \times n}$.

Proof: From (17), we have

$$\bar{G}^T Y_{\alpha \times n} = P + Q\Delta + \bar{G}^T E.$$

The error vector in ℓ_jth column is then

$$\bar{G}^T \boldsymbol{e}_{\ell_j}^T = \left(\boldsymbol{e}_{\ell_j} \bar{G} \right)^T \ . \tag{18}$$

Since $\bar{G}$ is a generator matrix of the $[n, k-1]$ RS code, $\boldsymbol{e}_{\ell_j} \bar{G}$ in (18) is a nonzero codeword in the RS code. Hence, the number of nonzero symbols in $\boldsymbol{e}_{\ell_j} \bar{G}$ is at least $n-k+2$, the minimum Hamming distance of the RS code. ∎

Let $\bar{G}^T Y_{\alpha \times n} = \tilde{P} + \tilde{Q}\Delta$, where $\tilde{P}$ and $\tilde{Q}$ can be obtained via (15) and (16). Let $\tilde{P}_i$ be the ith row of $\tilde{P}$ for $1 \leq i \leq n$. After decoding row $\tilde{P}_i$, we have the corresponding decoded codeword $\hat{P}_i$.[6] Let $\hat{P}$ be the matrix containing $\hat{P}_i$ as its ith row, $1 \leq i \leq n$. We next state the main theorem to perform data reconstruction.

Theorem 2: Let $E_P = \hat{P} \oplus \tilde{P}$ be the error pattern vector. Assume that the data collector accesses all storage nodes and there are v, $1 \leq v \leq \lfloor \frac{n-k}{2} \rfloor$, of them with errors. Then, there are at least $n-k+2-v$ nonzero elements in ℓ_jth column of E_P, $0 \leq j \leq v-1$, and at most v nonzero elements in the rest of the columns of E_P.

The proof can be found in our conference version of the paper [28]. In Figure 1, we present an illustrative example for ease of understanding of the main idea of Theorem 2. In the decoding process, $\tilde{P}$ is found from the received matrix Y. Each row of $\tilde{P}$ is then decoded to obtain $\hat{P}$. Since $\hat{P}$ is not symmetric, Z_1 cannot be obtained from it. According to E_p, one can identify the first column of Y as erroneous. In Fig. 1, erroneous elements in Y, $\tilde{P}$, and $\hat{P}$ are boxed.

The above theorem allows us to design a decoding algorithm that can correct up to $\lfloor \frac{n-k}{2} \rfloor$ errors.[7] In particular, we need to examine the erroneous positions in $\bar{G}^T E$. Since $1 \leq v \leq \lfloor \frac{n-k}{2} \rfloor$, we have $n-k+2-v \geq \lfloor \frac{n-k}{2} \rfloor + 1 > v$. Thus,

[6] If the decoder cannot generate a valid codeword when decoding $\tilde{P}_i$, then we set $\hat{P}_i = \tilde{P}_i$.

[7] In constructing $\tilde{P}$ we only get $n-1$ values (excluding the diagonal). Since the minimum Hamming distance of an $[n, k-1]$ RS code is $n-k+2$, the error-and-erasure decoding scheme can only correct up to $\lfloor \frac{n-1-k+2-1}{2} \rfloor$ errors.

$GF(2^3), n = 7, k = 4, \gamma = 5, \alpha = 3, d = 6$

$$G = \begin{bmatrix} 5 & 7 & 7 & 4 & 1 & 0 & 0 \\ 2 & 4 & 6 & 1 & 0 & 1 & 0 \\ 5 & 5 & 3 & 2 & 0 & 0 & 0 \\ 7 & 1 & 3 & 3 & 6 & 0 & 0 \\ 1 & 6 & 4 & 2 & 0 & 1 & 0 \\ 7 & 2 & 2 & 4 & 0 & 0 & 3 \end{bmatrix}, U = \begin{bmatrix} 0 & 2 & 3 & 0 & 2 & 2 \\ 2 & 0 & 2 & 2 & 6 & 4 \\ 3 & 2 & 6 & 2 & 4 & 5 \end{bmatrix}, C = \begin{bmatrix} 7 & 4 & 5 & 3 & 0 & 0 & 5 \\ 2 & 7 & 3 & 0 & 5 & 6 & 5 \\ 4 & 4 & 5 & 5 & 4 & 6 & 2 \end{bmatrix}$$

$$Y = \begin{bmatrix} \boxed{0} & 4 & 5 & 3 & 0 & 0 & 5 \\ \boxed{6} & 7 & 3 & 0 & 5 & 6 & 5 \\ \boxed{0} & 4 & 5 & 5 & 4 & 6 & 2 \end{bmatrix}, \tilde{P} = \begin{bmatrix} \times & \boxed{5} & \boxed{3} & \boxed{7} & \boxed{5} & \boxed{1} & \boxed{1} \\ \boxed{5} & \times & 0 & 1 & 7 & 4 & 2 \\ \boxed{3} & 0 & \times & 3 & 2 & 3 & 4 \\ \boxed{7} & 1 & 3 & \times & 4 & 7 & 2 \\ \boxed{5} & 7 & 2 & 4 & \times & 2 & 3 \\ \boxed{1} & 4 & 3 & 7 & 2 & \times & 2 \\ \boxed{1} & 2 & 4 & 2 & 3 & 2 & \times \end{bmatrix}, \hat{P} = \begin{bmatrix} 0 & \boxed{7} & \boxed{3} & \boxed{1} & \boxed{5} & \boxed{1} & \boxed{1} \\ 4 & 4 & 0 & 1 & 7 & 4 & 2 \\ 5 & 0 & 3 & 3 & 2 & 3 & 4 \\ 6 & 1 & 3 & 5 & 4 & 7 & 2 \\ 0 & 7 & 2 & 4 & 0 & 2 & 3 \\ 0 & 4 & 3 & 7 & 2 & 0 & 2 \\ 3 & 2 & 4 & 2 & 3 & 2 & 6 \end{bmatrix}$$

$$Ep = \begin{bmatrix} \times & 2 & 0 & 6 & 0 & 0 & 0 \\ 1 & \times & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & \times & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & \times & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & \times & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & \times & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix}, P = \begin{bmatrix} \times & 4 & 5 & 6 & 0 & 0 & 3 \\ 4 & \times & 0 & 1 & 7 & 4 & 2 \\ 5 & 0 & \times & 3 & 2 & 3 & 4 \\ 6 & 1 & 3 & \times & 4 & 7 & 2 \\ 0 & 7 & 2 & 4 & \times & 2 & 3 \\ 0 & 4 & 3 & 7 & 2 & \times & 2 \\ 3 & 2 & 4 & 2 & 3 & 2 & \times \end{bmatrix}$$

Fig. 1. An example to illustrate the results of Theorem 2

the approach to locate all erroneous columns in $\tilde{P}$ is to find out all columns in E_P whose number of nonzero elements are greater than or equal to $\lfloor \frac{n-k}{2} \rfloor + 1$. After we locate all erroneous columns, we can follow a procedure similar to that given in the no error (or single error) case to recover Z_1 from $\hat{P}$.

The above decoding procedure is guaranteed to recover Z_1 (Z_2) when all n storage nodes are accessed. However, it is not very efficient in terms of bandwidth usage. Next, we present a progressive decoding version of the proposed algorithm that accesses extra nodes only when necessary. Before presenting it, we need the following corollary.

Corollary 3: Consider that one accesses $k+2v$ storage nodes, among which v nodes are erroneous and $1 \leq v \leq \lfloor \frac{n-k}{2} \rfloor$. There are at least $v+2$ nonzero elements in the ℓ_Jth column of E_P, $0 \leq j \leq v-1$, and at most v nonzero elements among the remaining columns of E_P.

Proof: This is a direct result from Theorem 2 when we delete $n-(k+2v)$ elements in each column of E_P according to the size of $Y_{\alpha \times (k+2v)}$ and $n-k+2-v-\{n-(k+2v)\} = v+2$. ∎

Based on Corollary 3, we can design a progressive decoding algorithm [29] that retrieves extra data from the remaining storage nodes when necessary. To handle Byzantine fault tolerance, it is necessary to perform integrity check after the original data is reconstructed. Two verification mechanisms have been suggested in [22]: cyclic redundancy check (CRC) and cryptographic hash function. Both mechanisms introduce redundancy to the original data before they are encoded and are suitable to be used in combination with the decoding algorithm.

The progressive decoding algorithm starts by accessing k storage nodes. Error-and-erasure decoding succeeds only when there is no error. If the integrity check passes, then the data collector recovers the original data. If the decoding procedure fails or the integrity check fails, then the data collector retrieves two more blocks of data from the remaining storage nodes. Since the data collector has $k+2$ blocks of data, error-and-erasure decoding can correctly recover the original data if there is only one erroneous storage node among the $k+1$ nodes accessed. If the integrity check passes, then the data collector recovers the original data. If the decoding procedure fails or

the integrity check fails, then the data collector retrieves two more blocks of data from the remaining storage nodes. The data collector repeats the same procedure until it recovers the original data or runs out of the storage nodes. The detailed decoding procedure is summarized in Algorithm 1 given in our conference version of the paper [28].

IV. Encoding and Decoding schemes for Product-Matrix MBR Codes

In this section, we will find a generator matrix of the form (10) such that the row with the maximum Hamming weight has the least number of nonzero elements. This generator matrix is thus a least-update-complexity matrix. A decoding scheme for MBR codes that can correct more error patterns is also provided.

A. Encoding Scheme for MBR Codes

Let $g(x) = \prod_{j=1}^{n-k}(x - a^j) = \sum_{i=0}^{n-k} g_i x^i$ be the generator polynomial of the $[n, k]$ RS code and $f(x) = \prod_{j=1}^{n-d}(x - a^j) = \sum_{i=0}^{n-d} f_i x^i$ the generator polynomial of the $[n, d]$ RS code, where a is a generator of $GF(2^m)$.[8] A matrix G can be constructed as

$$G = \begin{bmatrix} G_k \\ S \end{bmatrix}, \tag{19}$$

where

$$G_k = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & 0 & \cdots & 0 \\ & & & \vdots & & & & \\ 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} \end{bmatrix} \tag{20}$$

and

$$S = \begin{bmatrix} f_0 & f_1 & \cdots & f_{n-d} & 0 & 0 & \cdots & 0 & 0 \\ 0 & f_0 & \cdots & f_{n-d-1} & f_{n-d} & 0 & \cdots & 0 & 0 \\ & & & \vdots & & & & & \\ 0 & \cdots & 0 & f_0 & \cdots & f_{n-d} & 0 & \cdots & 0 \end{bmatrix} \tag{21}$$

The dimensions of G_k and S are $k \times n$ and $(d - k) \times n$, respectively. Next, we state the main theorem about the rank of G given in (19).

Theorem 3: The rank of G given in (19) is d. That is, it is a generator matrix of the MBR code.

The proof of Theorem 3 can be found in Appendix B.

Corollary 4: The G given in (19) is the least-update-complexity matrix.

Proof: See Appendix C. ∎

Since $\bar{C}$ is also a cyclic code, it can be arranged as a systematic code. G_k is then given by $G_k = [B_k \ I]$, where

$$B_k = \begin{bmatrix} b_{00} & b_{01} & b_{02} & \cdots & b_{0(n-k-1)} \\ b_{10} & b_{11} & b_{12} & & \\ b_{20} & b_{21} & b_{22} & \cdots & b_{1(n-k-1)} \\ & \vdots & & & \vdots \\ b_{(k-1)0} & b_{(k-1)1} & b_{(k-1)2} & \cdots & b_{(k-1)(n-k-1)} \end{bmatrix} \tag{22}$$

[8] We assume that $n - k$ and $n - d$ are even.

I is the identity matrix,

$$x^{n-k+i} = u_i(x)g(x) + b_i(x) \text{ for } 0 \le i \le k-1 \ ,$$

and $b_i(x) = b_{i0} + b_{i1}x + \cdots + b_{i(n-k-1)}x^{n-k-1}$. It is easy to see that G with G_k as a submatrix is still a least-update-complexity matrix. The advantage of a systematic code will become clear in the decoding procedure of the MBR code. Note that the above claim for least-update-complexity is true with product-matrix construction only when G_k and G are chosen as the generator matrices of MDS codes.

We now consider the number of encoded symbols that need to be updated when a single data symbol is modified. First, we assume that the modified data symbol is located in A_1. If the modified data symbol is located on the diagonal of A_1, $(n - k + 1)$ encoded symbols need to be updated; otherwise, two corresponding encoded symbols in A_1 are modified such that $2(n - k + 1)$ encoded symbols need to be updated. Next, we assume that the modified data symbol is located in A_2. Then $(n - k + 1) + (n - d + 1) = 2n - k - d + 2$ encoded symbols need to be updated.

B. Decoding Scheme for MBR Codes

The generator polynomial of the RS code encoded by (22) has $a^{n-k}, a^{n-k-1}, \ldots, a$ as roots. Hence, the progressive decoding scheme based on the $[n, k]$ RS code given in [22] can be applied to decode the MBR code. The decoding algorithm given in [22] is slightly modified as follows.

Assume that the data collector retrieves encoded symbols from ℓ storage nodes $j_0,\ j_1, \ldots,\ j_{\ell-1}$, $k \le \ell \le n$. The data collector receives d vectors where each vector has ℓ symbols. Denoting the first k vectors among the d vectors as $Y_{k\times\ell}$ and the remaining $d - k$ vectors as $Y_{(d-k)\times\ell}$. By the encoding of the MBR code, the codewords in the last $d - k$ rows of C can be viewed as encoded by G_k instead of G. Hence, the decoder of the $[n, k]$ RS code can be applied on $Y_{(d-k)\times\ell}$ to recover the codewords in the last $d - k$ rows of C.

Let $\tilde{C}_{(d-k)\times k}$ be the last k columns of the codewords recovered by the error-and-erasure decoder in the last $d - k$ rows of C. Since the code generated by (22) is a systematic code, A_2 in U can be reconstructed as

$$\tilde{A}_2 = \tilde{C}_{(d-k)\times k} \ . \tag{23}$$

We then calculate the j_0th, j_1th, …, $j_{\ell-1}$th columns of $\tilde{A}_2^T \cdot B$ as $E_{k\times\ell}$, and subtract $E_{k\times\ell}$ from $Y_{k\times\ell}$:

$$Y'_{k\times\ell} = Y_{k\times\ell} - E_{k\times\ell} \ . \tag{24}$$

Applying the error-and-erasure decoding algorithm of the $[n, k]$ RS code again on $Y'_{k\times\ell}$ we can reconstruct A_1 as

$$\tilde{A}_1 = \tilde{C}_{k\times k} \ . \tag{25}$$

The decoded information sequence is then verified by data integrity check. If the integrity check is passed, the data reconstruction is successful; otherwise the progressive decoding procedure is applied, where two more storage nodes need to be accessed from the remaining storage nodes in each round until no further errors are detected.

The decoding capability of the above decoding algorithm is $\frac{n-k}{2}$. Since each erroneous storage node sends $\alpha = d$ symbols to the data collector, in general, not all α symbols are wrong if failures in the storage nodes are caused by random faults. Hence, the decoding algorithm given in [22] can be modified as follows to extend error correction capability. After decoding $Y_{(d-k)\times\ell}$, one can locate the erroneous columns of $Y_{(d-k)\times\ell}$ by comparing the decoded result to it. Assume that there are v erroneous columns located. Delete the corresponding columns in $E_{k\times\ell}$ and $Y_{k\times\ell}$ and we have

$$Y'_{k\times(\ell-v)} = Y_{k\times(\ell-v)} - E_{k\times(\ell-v)} \ . \tag{26}$$

Applying the error-and-erasure decoding algorithm of the $[n,k]$ RS code again on $Y'_{k\times(\ell-v)}$ to reconstruct A_1 if $\ell - v \geq k$; otherwise progressive decoding is applied. The modified decoding algorithm is summarized in Algorithm 1. The advantage of the modified decoding algorithm is that it can correct up to

$$\frac{n-k}{2} + \left\lfloor \frac{n-k+1-\lfloor \frac{n-k+1}{2} \rfloor}{2} \right\rfloor$$

errors even though not all error patterns up to this number of errors can be corrected.

Algorithm 1: Decoding of MBR Codes for Data Reconstruction

begin
- The data collector randomly chooses k storage nodes and retrieves encoded data, $Y_{d\times k}$;
- $\ell \leftarrow k$;
- **repeat**
  - Perform progressive error-erasure decoding on last $d-k$ rows in $Y_{d\times\ell}$, $Y_{(d-k)\times\ell}$, to recover $\tilde{C}$ (error-erasure decoding performs $d-k$ times);
  - Locate the erroneous columns in $Y_{(d-k)\times\ell}$ (assume to have v columns);
  - Calculate $\tilde{A}_2$ via (23);
  - Calculate $\tilde{A}_2 \cdot B$ and obtain $Y'_{k\times(\ell-v)}$ via (26);
  - **if** *($\ell - v \geq k$)* **then**
    - Perform progressive error-erasure decoding on $Y'_{k\times(\ell-v)}$ to recover the first k rows in codeword vector (error-erasure decoding performs k times);
    - Calculate $\tilde{A}_1$ via (25);
    - Recover the information sequence $\tilde{\boldsymbol{m}}$ from $\tilde{A}_1$ and $\tilde{A}_2$;
    - **if** *integrity-check($\tilde{\boldsymbol{m}}$) = SUCCESS* **then**
      - **return** $\tilde{\boldsymbol{m}}$;
  - $\ell \leftarrow \ell + 2$;
  - Retrieve two more encoded data from remaining storage nodes and merge them into $Y_{d\times\ell}$;
- **until** $\ell \geq n-2$;
- **return** FAIL;

One important function of regenerating codes is to perform data regeneration with least repair bandwidth when one node has failed. Since the decoding schemes proposed in [22] can be applied directly without modification to the proposed MSR and MBR codes in this work, the decoding schemes of data regeneration for these codes are omitted in this work. The interested readers can refer to [22] for details on these decoding schemes.

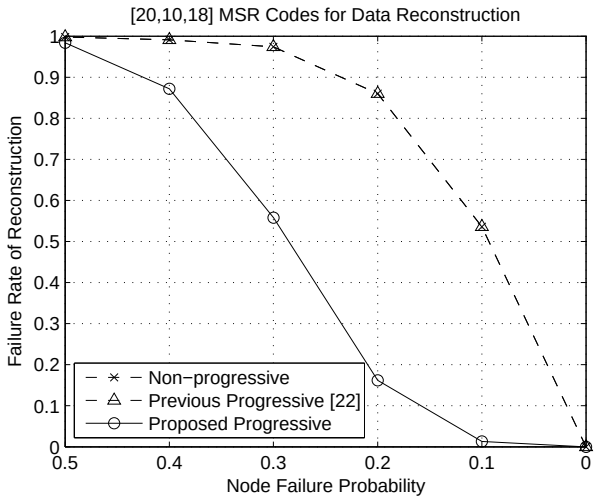

Fig. 2. Comparison of the failure rate between the algorithm in [22] and the proposed algorithm for $[20, 10, 18]$ MSR codes

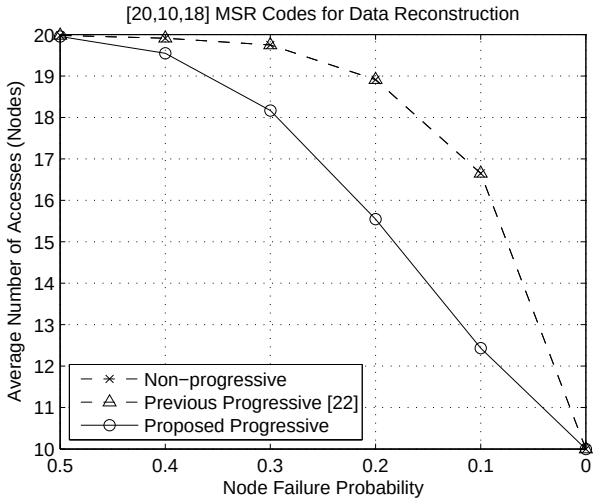

Fig. 3. Comparison of the number of node accesses between the algorithm in [22] and the proposed algorithm for $[20, 10, 18]$ MSR codes

V. Performance Evaluation

In this section, we first analyze the fault-tolerance capability of the proposed codes in the presence of crash-stop and Byzantine failures, and then carry out numerical simulations to evaluate the performance of the proposed schemes. The issue of update-efficiency is also discussed in this section.

The fault-tolerance capability of product-matrix MSR and MBR codes has been investigated fully in [22] where CRC or cryptographic hash function was adopted as the data integrity check. Their error-correction capability was also presented in [23]. Progressive decoding algorithms have been

implemented that incrementally retrieve additional stored data and perform data reconstruction when errors have been detected. Since cryptographic hash functions have better security strength than CRC on data integrity check, it is adopted to verify the integrity of stored data. In particular, for data reconstruction, the hash value is coded along with the original data and distributed among storage nodes.

We consider two types of failures, crash-stop failures and Byzantine failures. Nodes are assumed to fail independently. In both cases, the fault-tolerance capability is measured by the maximum number of failures that the system can handle to maintain functionality.

A crash-stop failure on a node can be viewed as an erasure in the codeword. Since k nodes need to be alive for data reconstruction, the maximum number of crash-stop failures that can be tolerated in data reconstruction is $n-k$. Note that since all accessed nodes contain correct data, the associated hash values are also correct. For an error-correcting code, two additional correct code fragments are needed to correct one erroneous code fragment. Thus, with the proposed MSR decoding algorithm, $\lfloor \frac{n-k}{2} \rfloor$ erroneous nodes can be tolerated in data reconstruction. The proposed MBR decoding algorithm can not only tolerate any $\frac{n-k}{2}$ erroneous nodes but it can also correct up to

$$\frac{n-k}{2} + \left\lfloor \frac{n-k+1-\lfloor \frac{n-k+1}{2} \rfloor}{2} \right\rfloor$$

errors even though not all error patterns up to this number of errors can be corrected.

The proposed data reconstruction algorithms for MSR and MBR codes have also been evaluated by Monte Carlo simulations. In the simulations, the codes based on shortened RS codes are employed for simulations. They are compared with the data reconstruction algorithms previously proposed in [22]. The performance of a traditional decoding scheme that is non-progressive is also provided for comparison purposes.[9] After k nodes are accessed, if the integrity check fails, the data collector will access all remaining $n-k$ nodes for data reconstruction in the non-progressive decoding scheme. Each data point is generated from 10^3 simulation runs. Storage nodes may fail arbitrarily with the Byzantine failure probability ranging from 0 to 0.5. In both schemes, $[n, k, d]$ and m are chosen to be $[20, 10, 18]$ and 5, respectively.

In the first set of simulations, we compare the proposed algorithm with the progressive algorithm in [22] and the non-progressive algorithm in terms of the failure rate of reconstruction and the average number of node accesses, which indicates the required bandwidth for data reconstruction. Failure rate is defined as the percentage of runs for which reconstruction fails (due to insufficient number of healthy storage nodes). Figure 2 shows that the proposed algorithm can successfully reconstruct the data with much higher probability than the previous progressive or non-progressive algorithm for the same node failure probability. For example, when the node failure probability is 0.1, only about 1% of the time, reconstruction fails using the proposed algorithm, in contrast to 50% with the old algorithm. The advantage of the proposed algorithm is also pronounced in the average number of accessed nodes for data reconstruction, as illustrated in Fig. 3. For example, on an average, only 2.5 extra nodes are needed by the proposed algorithm under the node failure probability of 0.1; while over 6.5 extra nodes are required by the old algorithm in [22]. It should be noted that the actual saving attained by the new algorithm depends on the values of n, k, d and the number of errors.

The previous and proposed decoding algorithms for MBR codes are compared in the second set of simulations. Figures 4 and 5 show that both of the progressive algorithms have identical failure rates of reconstruction and average number of accessed nodes. This result implies that the specific error patterns, which only the proposed algorithm is able

[9]Since no data integrity check is performed in the decoding algorithms given in [23], to reach error-correction capability of the MSR and MBR codes, n nodes need to be accessed. Hence, the number of accessed nodes in decoding algorithms in [23] is much larger than those of the non-progressive version presented here.

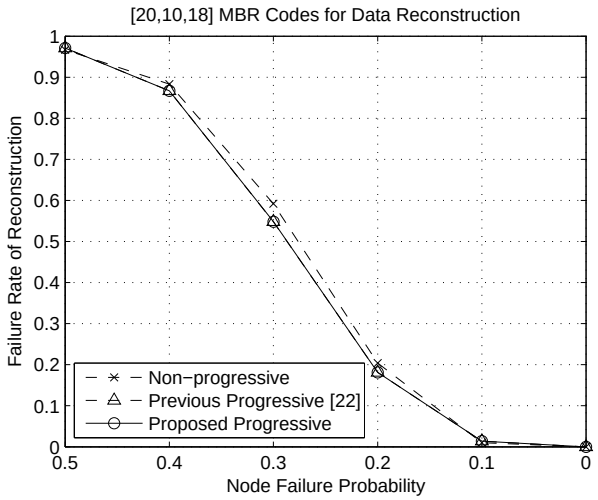

Fig. 4. Failure-rate comparison between the previous algorithm in [22] and the proposed algorithm for $[20, 10, 18]$ MBR codes

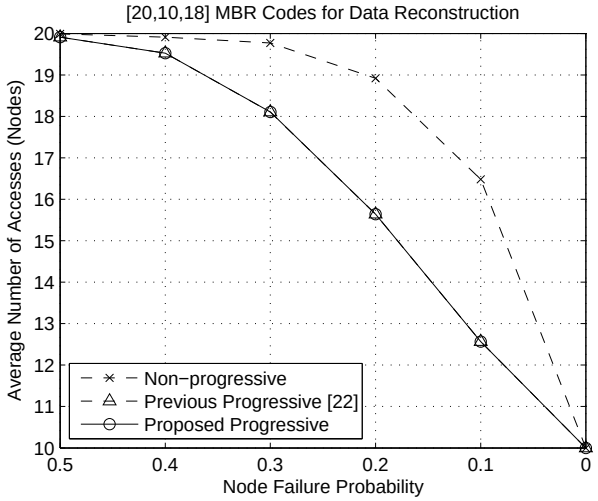

Fig. 5. Node-access comparison between the previous algorithm in [22] and the proposed algorithm for $[20, 10, 18]$ MBR codes

to handle for successful data reconstruction, do not occur very frequently. However, the computational complexity of the proposed algorithm for MBR encoding is much lower since no matrix inversion and multiplications are needed in (23) and (25). Moreover, both the progressive algorithms are better than the non-progressive algorithm in failure rates for reconstruction and average number of accessed nodes.

In the evaluation of the update complexity, two measures are considered: the metric given in [26] and the number of updated symbols when a single data symbol is modified. The first metric corresponds to the maximum number of nonzero elements in all rows of the generator matrix G. Denote by $\eta(R)$ the ratio of the update complexity of the proposed generator matrix to that of the generator matrix given in [13], where $R = k/n$. It can be seen that,

$$\eta_{MSR}(R) = \frac{n-\alpha+1}{n} \approx 1 - R$$

for MSR codes since the generator matrix of the MSR code proposed in [13] is a Vandermonde matrix. Two types of generator matrices of the MBR codes have been proposed in [13]: the Vandermonde matrix and a systematic matrix based on Cauchy matrix. With Vandermonde matrix,

$$\eta_{MBR}(R) = \frac{n-k+1}{n} \approx 1 - R \ .$$

The systematic matrix based on Cauchy matrix is given by [13]

$$\left[\begin{array}{cc} I_k & \phi^T \\ \mathbf{0} & \Delta^T \end{array} \right] ,$$

where I_k is the $k \times k$ identity matrix, $\mathbf{0}$ is the $(d-k) \times k$ all-zero matrix, and $[\phi \ \Delta]$ is a Cauchy matrix. Since all elements in the Cauchy matrix are nonzero,

$$\eta_{MBR}(R) = \frac{n-k+1}{n-k+1} = 1 \ .$$

The number of updated symbols that need to be modified when a single data symbol is changed in MSR and MBR codes are summarized in Table I. By the arguments given in previous sections, the average number of updated symbols when a single data symbol is modified for the proposed MSR and MBR codes are $2(n-\alpha+1)\frac{\alpha}{\alpha+1}$ and $\frac{kd(n-k+1)+k(d-k)(n-d+1)}{2kd-k(k-1)}$, respectively. These numbers for Vandermonde-matrix based MSR and MBR codes are $2n\frac{\alpha}{\alpha+1}$ and $\frac{n(2kd-k^2)}{2kd-k(k-1)}$, respectively. The number is $\frac{kd(n-k+1)+k(d-k)(n-k)}{2kd-k(k-1)}$ for the systematic MBR code based on Cauchy matrix. Note that, the numbers for systematic codes based on linear remapping are obtained from simulations. From Table I, one can observe that the proposed method has the best performance in terms of the number of updated symbols when a single data symbol is modified, and the systematic version based on linear remapping performs the worst among all schemes in the table. For example, for the $[20, 10, 18]$ MSR code, the average number of encoded symbols that need to be updated for a single data symbol modification is 88 in the systematic version based on linear remapping but only 22 with the proposed encoding matrix. This is a 4-fold improvement in complexity. In the case of the $[100, 40, 78]$ MSR code, the improvement is 19-fold. Hence, the proposed approach has much lower update complexity than the systematic approach. It can be seen that after linear remapping, the modified symbols occur in almost all check positions of the code vector. This is because even when only one data symbol is modified, due to the symmetry requirement on the information matrix, the modification propagates to check positions of all codewords (rows) in the code vector through linear remapping. One can also observe that even though the Cauchy-based MBR code results in the same maximum number of nonzero elements in all rows of the generator matrix as the proposed MBR code, it requires more symbol updates when a single data symbol is modified.

Next we compare the update complexity of the proposed schemes to other codes that are not based on product-matrix framework. In [26], the authors proved that there exists an MBR code with logarithmic complexity with respect to the number of storage nodes n when n is very large. The scheme takes advantage of randomization during the construction of update-efficient codes and is asymptotically optimal. In contrast, the proposed scheme is based on Reed-Solomon codes, and thus its best update complexity is linear with n. We also compare the proposed $[6, 3, 4]$ MSR code to the $[6, 3, 5]$ MSR code given in Fig. 9 of [18] designed using an interference alignment approach. The maximum number of nonzero elements in all rows of the generator matrix G for the $[6, 3, 5]$ code in [18] is 6 and 5 for the proposed $[6, 3, 4]$ code, respectively. The average number of updated symbols when a single data symbol is modified for the $[6, 3, 5]$ code in [18] is 6 and 6.7 for the proposed $[6, 3, 4]$ code, respectively.

VI. Conclusion

In this work, we proposed new encoding and decoding schemes for the $[n, d]$ error-correcting MSR and MBR codes that generalize the previously proposed codes in [22]. Through both theoretical analysis and numerical simulations, we have demonstrated the superior low update complexity, and low computation complexity of the new codes, as well as an improved error correction capability for MBR codes.

Clearly, there is a trade-off between the update complexity and error correction capability of regenerating codes. In this work, we first designed encoders of product-matrix regenerating codes and then optimized their update complexity. Possible future work includes the study of encoding schemes that first design regenerating codes with good update complexity and then optimize their error correction capability.

The least update-complexity codes in this work minimize the maximum number of nonzero elements in all rows of the generation matrix, but they do not minimize the number of symbol updates when a single data symbol is modified. For instance, due to symmetry requirement on the information vector, two symbols need to be updated in the information vector during the encoding process for a single modified symbol in some cases. Another possible future work is to seek codes with the least number of updated encoded symbols.

TABLE I
COMPARISON OF THE AVERAGE NUMBER OF UPDATED SYMBOLS WHEN A SINGLE DATA SYMBOL IS MODIFIED

	MSR code		MBR code	
	[20 10 18]	[100 40 78]	[20 10 18]	[100 40 78]
Proposed method	22	121	8	48
Vandermonde matrix	36	195	19	99
Systematic version based on linear remapping [13]*	88	2323	34	807
Systematic version based on Cauchy matrix [13]	-	-	10	60

* The numbers are obtained from simulation experiments

APPENDIX A
PROOF OF THEOREM 1

We need to prove that each row of $\bar{G}\Delta$ is a codeword of C_{0d} and all rows in G are linearly independent. Let $\hat{C}_{0\alpha}$ be the dual code of $C_{0\alpha}$. It is well-known that $\hat{C}_{0\alpha}$ is an $[n, n-\alpha]$ RS code [30], [31]. Similarly, let $\hat{C}_{0d}$ be the dual code of C_{0d} and its generator matrix be H_d. Note that H_d is a parity-check matrix of C_{0d}. Let $h_d(x) = (x^n-1)/g_{0d}(x)$ and $h_\alpha(x) = (x^n-1)/g_{0\alpha}(x)$. Then, the roots of $h_d(x)$ and $h_\alpha(x)$ are $a^{n-d}, a^{n-d+1}, \ldots, a^{n-1}$ and $a^{n-\alpha}, a^{n-\alpha+1}, \ldots, a^{n-1}$, respectively. Since an RS code is also a cyclic code, the generator polynomials of $\hat{C}_{0d}$ and $\hat{C}_{0\alpha}$ are $\hat{h}_d(x)$ and $\hat{h}_\alpha(x)$, respectively, where $\hat{h}_d(x) = x^{n-d}h_d(x^{-1})$ and $\hat{h}_\alpha(x) = x^{n-\alpha}h_\alpha(x^{-1})$. Clearly, the roots of $\hat{h}_d(x)$ are $a^{-(n-d)}, a^{-(n-d+1)}, \ldots, a^{-(n-1)}$ that are equivalent to $a^d, a^{d-1}, \ldots, a^1$. Similarly, the roots of $\hat{h}_\alpha(x)$ are $a^\alpha, a^{\alpha-1}, \ldots, a^1$. Since $\hat{h}_d(x)$ has roots of $a^d, a^{d-1}, \ldots, a^1$, we can choose

$$H_d = \left[\begin{array}{cccc} \boldsymbol{h}_0 & \boldsymbol{h}_1 & \cdots & \boldsymbol{h}_{n-1} \end{array} \right], \tag{27}$$

where $\boldsymbol{h}_i = [(a^i)^0, (a^i)^1, \ldots, (a^i)^{n-d-1}]^T$, as the generator matrix of $\hat{C}_{0d}$. To prove that each row of $\bar{G}\Delta$ is a codeword of the RS code C_{0d} generated by G, it is sufficient to show that $\bar{G}\Delta H_d^T = \mathbf{0}$. From the symmetry of Δ, we have

$$\bar{G}\Delta H_d^T = \bar{G}\left(H_d\Delta\right)^T .$$

Thus, we only need to prove that each row of $H_d\Delta$ is a codeword in $\hat{C}_{0\alpha}$. Let the diagonal elements of Δ be $b_0, b_1, \ldots, b_{n-1}$. The ith row of $H_d\Delta$ is thus $r_i(x) = \sum_{j=0}^{n-1} b_j (a^j)^{i-1} x^j$ in the polynomial representation. Let $(b_0, b_1, \ldots, b_{n-1})$ be a codeword in $C_{1(\alpha+1)}$. Then, we have

$$\sum_{j=0}^{n-1} b_j (a^{\ell'})^j = 0 \text{ for } 1 \le \ell' \le n-\alpha-1 . \tag{28}$$

Substituting $x = a^\ell$, for $1 \le \ell \le \alpha$, into $r_i(x)$, it becomes

$$r_i(a^\ell) = \sum_{j=0}^{n-1} b_j (a^j)^{i-1} (a^\ell)^j = \sum_{j=0}^{n-1} b_j (a^{i-1+\ell})^j . \tag{29}$$

Let $\ell' = i-1+\ell$. Since $1 \le i \le n-d$ and $1 \le \ell \le \alpha$, $1 \le \ell' \le n-\alpha-1$. By (28), $r_i(a^\ell) = 0$ for $1 \le i \le n-d$ and $1 \le \ell \le \alpha$. Hence, each row of $H_d\Delta$ is a codeword in $\hat{C}_{0\alpha}$.

The b_is need to make all rows in G linearly independent. Since all rows in $\bar{G}$ or those in $\bar{G}\Delta$ are linearly independent, it is sufficient to prove that $C_{0\alpha} \cap C_\Delta = \{\mathbf{0}\}$, where C_Δ is the code generated by $\bar{G}\Delta$. Let $\boldsymbol{c}'$ be a codeword in C_Δ. $\boldsymbol{c}' = \boldsymbol{c}\Delta$ for some $\boldsymbol{c} \in C_{0\alpha}$. We can choose $\bar{G} = \left[\begin{array}{cccc} \bar{\boldsymbol{g}}_0 & \bar{\boldsymbol{g}}_1 & \cdots & \bar{\boldsymbol{g}}_{n-1} \end{array} \right]$, where $\bar{\boldsymbol{g}}_i = [(a^i)^1, (a^i)^2, \ldots, (a^i)^\alpha]^T$, as the generator matrix of $C_{0\alpha}$. Then $\boldsymbol{c}' = \boldsymbol{u}\bar{G}\Delta$ for some $\boldsymbol{u} = [u_0, u_1, \ldots, u_\alpha]$. Evaluating $\boldsymbol{c}'(x)$ at $a^0, a^1, \ldots, a^{n-\alpha-1}$ and putting them into a matrix form, we have $\boldsymbol{u}\bar{G}\Delta\tilde{G} = \boldsymbol{z}$, where $\tilde{G} = \left[\begin{array}{cccc} \tilde{\boldsymbol{g}}_0 & \tilde{\boldsymbol{g}}_1 & \cdots & \tilde{\boldsymbol{g}}_{n-\alpha-1} \end{array} \right]$, $\tilde{\boldsymbol{g}}_i = [(a^i)^1, (a^i)^2, \ldots, (a^i)^{n-1}]^T$, and $\boldsymbol{z}$ is an $(n-\alpha)$-dimensional vector. If $\boldsymbol{z} = \mathbf{0}$, then $\boldsymbol{c}\Delta \in C_{0\alpha}$; otherwise, $\boldsymbol{c}\Delta \notin C_{0\alpha}$. Taking transpose on both sizes of (A), it becomes $\tilde{G}^T\Delta\bar{G}^T\boldsymbol{u}^T = \boldsymbol{z}^T$, where

$$\tilde{G}^T\Delta\bar{G}^T = \left[\begin{array}{cccc} \sum_{j=0}^{n-1} b_j a^j & \sum_{j=0}^{n-1} b_j (a^2)^j & \cdots & \sum_{j=0}^{n-1} b_j (a^\alpha)^j \\ \sum_{j=0}^{n-1} b_j (a^2)^j & \sum_{j=0}^{n-1} b_j (a^3)^j & \cdots & \sum_{j=0}^{n-1} b_j (a^{\alpha+1})^j \\ & & \vdots & \\ \sum_{j=0}^{n-1} b_j (a^{n-\alpha})^j & \sum_{j=0}^{n-1} b_j (a^{n-\alpha+1})^j & \cdots & \sum_{j=0}^{n-1} b_j (a^{n-1})^j \end{array} \right] \tag{30}$$

Since $(b_0, b_1, \ldots, b_{n-1}) \in C_{1(\alpha+1)}$,

$$\sum_{j=0}^{n-1} b_j (a^\ell)^j = 0 \text{ for } 1 \le \ell \le n-\alpha-1 . \tag{31}$$

Substituting (31) into (30) and taking out rows with all zeros, we have $M\boldsymbol{u}^T = \tilde{\boldsymbol{z}}$, where

$$M = \left[\begin{array}{cccc} 0 & \cdots & 0 & \sum_{j=0}^{n-1} b_j (a^{n-\alpha})^j \\ 0 & \cdots & \sum_{j=0}^{n-1} b_j (a^{n-\alpha})^j & \sum_{j=0}^{n-1} b_j (a^{n-\alpha+1})^j \\ & \vdots & & \\ \sum_{j=0}^{n-1} b_j (a^{n-\alpha})^j & \cdots & \sum_{j=0}^{n-2} b_j (a^{n-2})^j & \sum_{j=0}^{n-1} b_j (a^{n-1})^j \end{array} \right], \tag{32}$$

and

$$\tilde{\boldsymbol{z}} = \left[\begin{array}{c} z_{n-2\alpha} \\ z_{n-2\alpha+1} \\ \vdots \\ z_{n-\alpha-1} \end{array} \right] . \tag{33}$$

If $\sum_{j=0}^{n-1} b_j (a^{n-\alpha})^j = 0$, i.e., $a^{n-\alpha}$ is a root of $\sum_{j=0}^{n-1} b_j x^j$, then $\boldsymbol{c}' = [1, 0, \ldots, 0]\bar{G}\Delta \in C_{0\alpha}$ due to the fact that $\boldsymbol{u} = [1, 0, \ldots, 0]$ makes $M\boldsymbol{u}^T = \tilde{\boldsymbol{z}} = \mathbf{0}$. Thus, we need to exclude the codewords in $C_{1(\alpha+1)}$ that have $a^{n-\alpha}$ as a root. These codewords turn out to be in $C_{1\alpha}$. If $\sum_{j=0}^{n-1} b_j (a^{n-\alpha})^j \neq 0$,

then it is clear that the only $\boldsymbol{u}$ making $\tilde{\boldsymbol{z}} = \mathbf{0}$ in (33) is the all-zero vector. Hence, any $(b_0, b_1, \ldots, b_{n-1}) \in C_{1(\alpha+1)} \backslash C_{1\alpha}$ does not make $\tilde{\boldsymbol{z}}$ zero except $\boldsymbol{u} = \mathbf{0}$.

APPENDIX B
PROOF OF THEOREM 3

Let the codes generated by G_k and G be $\bar{C}$ and C, respectively. It can be seen that any row in G_k and S is a cyclic shift of the previous row. Hence, all rows in G_k and S are linearly independent. Now we only consider the linear combination of rows in G chosen from both G_k and S. Since $\bar{C}$ is a linear code, the portion of the linear combination that contains only rows from G_k results in a codeword, named $\boldsymbol{c}$, in $\bar{C}$. Assume that the rows chosen from S are the j_0th, j_1th, ..., and $j_{\ell-1}$th rows. Recall that S can be represented by a polynomial matrix as

$$B(x) = \left[\begin{array}{ccccc} f(x) & xf(x) & x^2 f(x) & \cdots & x^{d-k-1} f(x) \end{array} \right]^T .$$

Hence, in the polynomial form, the linear combination can be represented as

$$\boldsymbol{c}(x) + \sum_{i=0}^{\ell-1} b_i x^{j_i - 1} f(x) \ , \tag{34}$$

where $\boldsymbol{c}(x)$ is not the all-zero codeword and not all $b_i = 0$. Since $c(x)$ is the code polynomial of $\bar{C}$, it is divisible by $g(x)$ and can be represented as $u(x)g(x)$. Assume that (34) is zero. Then we have

$$u(x)g(x) = -f(x) \sum_{i=0}^{\ell-1} b_i x^{j_i - 1} \ . \tag{35}$$

Recall that $g(x) = \prod_{i=1}^{n-k}(x - a^i)$ and $f(x) = \prod_{i=1}^{n-d}(x - a^i)$. Hence,

$$g(x) = f(x) \prod_{i=n-d+1}^{n-k} (x - a^i) \ . \tag{36}$$

Substituting (36) into (35) we have

$$u(x) \prod_{i=n-d+1}^{n-k} (x - a^i) = -\sum_{i=0}^{\ell-1} b_i x^{j_i - 1} \ . \tag{37}$$

That is, $\sum_{i=0}^{\ell-1} b_i x^{j_i-1}$ is divisible by $\prod_{i=n-d+1}^{n-k}(x - a^i)$. However, the degree of $\prod_{i=n-d+1}^{n-k}(x - a^i)$ is $d - k$ and the degree of $\sum_{i=0}^{\ell-1} b_i x^{j_i-1}$ is at most $d-k-2$ when $\ell = d-k-1$, the largest possible value for ℓ. Thus, $\sum_{i=0}^{\ell-1} b_i x^{j_i-1}$ is not divisible by $\prod_{i=n-d+1}^{n-k}(x - a^i)$ since not all $b_i = 0$. This is a contradiction. Since all rows in G_k and S are codewords in C, G is then a generator matrix of the $[n, d]$ RS code C.

APPENDIX C
PROOF OF COROLLARY 4

Since G_k must be the generator matrix of the $[n, k]$ RS code $\bar{C}$, the Hamming weight of each row of G_k is greater than or equal to the minimum Hamming distance of $\bar{C}$, $n-k+1$. Since the degree of $g(x)$ is $n - k$ and itself is a codeword in $\bar{C}$, the nonzero coefficients of $g(x)$ is $n - k + 1$ and each row of G_k is with $n-k+1$ Hamming weight. A similar argument can be applied to each row of S such that the Hamming weight of it is $n-d+1$. Thus, the G given in (19) has the least number of nonzero elements. Further, Since G_k is the generator matrix of the $[n, k]$ code, the minimum Hamming of its row can have is $n - k + 1$, namely, the minimum Hamming distance of the code. Hence, the row with maximum Hamming weight in G is $n - k + 1$.

ACKNOWLEDGEMENT

This work was supported in part by National Science Council of Taiwan, under grants NSC 99-2221-E-011-158-MY3, NSC 101-2221-E-011-069-MY3, and in part by National Science and Engineering Council of Canada under Discovery Grant. Han's work was completed during his visit to Syracuse University from 2012 to 2013.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *the 19th ACM SIGOPS Symp. on Operating Systems Principles*, Bolton Landing, NY, October 2003.

[2] J. K. et al., "OceanStore: an architecture for global-scale persistent storage," in *the 9th International Conference on Architectural Support for programming Languages and Operating Systems*, Cambridge, MA, November 2000.

[3] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total recall: system support for automated availability management," in *the 1st Conf. on Networked Systems Design and Implementation*, San Francisco, CA, March 2004.

[4] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *IEEE IINFOCOM*, Anchorage, Alaska, May 2007, pp. 2000–2008.

[5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inform. Theory*, vol. 56, pp. 4539 – 4551, September 2010.

[6] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *the 45th Annual Allerton Conference on Control, Computing, and Communication*, Urbana-Champaign, Illinois, September 2007.

[7] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 277 – 288, February 2010.

[8] D. F. Cullina, "Searching for minimum storage regenerating codes," California Institute of Technology Senior Thesis, 2009.

[9] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *IEEE Int. Symp. on Information Theory*, Seoul, Korea, July 2009, pp. 2276–2280.

[10] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *the 47th Annual Allerton Conference on Control, Computing, and Communication*, Urbana-Champaign, Illinois, September 2009, pp. 1243–1249.

[11] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Trans. Inform. Theory*, pp. 6734–6753, 2011.

[12] F. Oggier and A. Datta, "Byzantine fault tolerance of regenerating codes," in *IEEE International Conference on Peer-to-Peer Computing*, 2011, pp. 112–121.

[13] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inform. Theory*, vol. 57, pp. 5227–5239, August 2011.

[14] M. Gerami, M. Xiao, and M. Skoglund, "Optimal-cost repair in multi-hop distributed storage system," in *IEEE Int. Symp. on Information Theory*, 2011, pp. 1437–1441.

[15] M. Gerami and M. Xiao, "Repair for distributed storage systems with erasure channels," in *IEEE International Conference on Communications*, 2013, pp. 4058–4062.

[16] N. Shah, K. V. Rashmi, P. Kumar, and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions," *IEEE Trans. Inform. Theory*, vol. 58, no. 4, pp. 2134–2158, 2012.

[17] H. M. V. R. Cadambe, S. A. Jafar, "Distributed data storage with minimum storage regenerating codes - exact and functional repair are asymptotically equally efficient," arXiv:1004.4299v1 [cs.IT] 24 Apr 2010.
[18] C. Suh and K. Ramchandran, "Exact-repair mds code construction using interference alignment," *IEEE Trans. Inform. Theory*, pp. 1425 – 1442, March 2011.
[19] V. R. Cadambe and C. Jafar, "Interference alignment and degrees of freedom of the k-user interference channel," *IEEE Trans. Inform. Theory*, pp. 3425 – 3441, August 2008.
[20] M. A. Maddah-Ali, A. S. Motahari, and A. K. Khandani, "Communication over MIMO X channels: Interference alignment, decomposition, and performance analysis," *IEEE Trans. Inform. Theory*, pp. 3457 – 3470, August 2008.
[21] M. Gerami, M. Xiao, C. Fischione, and M. Skoglund, "Decentralized minimum-cost repair for distributed storage systems," in *IEEE International Conference on Communications*, 2013, pp. 1910–1914.
[22] Y. S. Han, R. Zheng, and W. H. Mow, "Exact regenerating codes for byzantine fault tolerance in distributed storage," in *IEEE INFOCOM 2012*, Orlendo, FL, March 2012.
[23] K. Rashmi, N. Shah, K. Ramchandran, and P. Kumar, "Regenerating codes for errors and erasures in distributed storage," in *IEEE Int. Symp. on Information Theory*, Cambridge, MA, July 2012.
[24] M. Kurihara and H. Kuwakado, "Coding for errors and erasures in random network coding," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E79-A, no. 2, pp. 1298 – 1304, February.
[25] Y. Han, H.-T. Pai, R. Zheng, and W. H. Mow, "Efficient exact regenerating codes for byzantine fault tolerance in distributed networked storage," *IEEE Trans. Commun.*, vol. 62, no. 2, pp. 385–397, February 2014.
[26] A. S. Rawat, S. Vishwanath, A. Bhowmick, and E. Soljanin, "Update efficient codes for distributed storage," in *IEEE Int. Symp. on Information Theory*, Saint Petersburg, Russia, July 2011.
[27] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York, NY: Elsevier Science Publishing Company, Inc., 1977.
[28] Y. Han, H.-T. Pai, R. Zheng, and P. Varshney, "Update-efficient regenerating codes with minimum per-node storage," in *IEEE Int. Symp. on Information Theory*, July 2013, pp. 1436–1440.
[29] Y. S. Han, S. Omiwade, and R. Zheng, "Progressive data retrieval for distributed networked storage," *IEEE Trans. on Parallel and Distributed Systems*, vol. 23, pp. 2303–2314, December 2012.
[30] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, Inc., 2004.
[31] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, NJ: John Wiley & Sons, Inc., 2005.

Yunghsiang S. Han Yunghsiang S. Han (S'90-M'93-SM'08-F'11) was born in Taipei, Taiwan, 1962. He received B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and a Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, in 1993. He was from 1986 to 1988 a lecturer at Ming-Hsin Engineering College, Hsinchu, Taiwan. He was a teaching assistant from 1989 to 1992, and a research associate in the School of Computer and Information Science, Syracuse University from 1992 to 1993. He was, from 1993 to 1997, an Associate Professor in the Department of Electronic Engineering at Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering at National Chi Nan University, Nantou, Taiwan from 1997 to 2004. He was promoted to Professor in 1998. He was a visiting scholar in the Department of Electrical Engineering at University of Hawaii at Manoa, HI from June to October 2001, the SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and CASE center at Syracuse University, NY from September 2002 to January 2004 and July 2012 to June 2013, and the visiting scholar in the Department of Electrical and Computer Engineering at University of Texas at Austin, TX from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering at National Taipei University, Taipei, Taiwan from August 2004 to July 2010. From August 2010, he is with the Department of Electrical Engineering at National Taiwan University of Science and Technology as Chair Professor. He is also a Chair Professor at National Taipei University from February 2015. His research interests are in error-control coding, wireless networks, and security.

Dr. Han was a winner of the 1994 Syracuse University Doctoral Prize and a Fellow of IEEE. One of his papers won the prestigious 2013 ACM CCS Test-of-Time Award in cybersecurity.

Hung-Ta Pai (S'91–M'99–SM'11) was born in Taichung, Taiwan. He received a B.Sc. degree from National Tsing Hua University, Taiwan, in 1992, and the M.Sc. and Ph.D. degrees from the University of Texas at Austin, Texas, in 1996 and 1999, respectively, all in electrical engineering. Hung-Ta Pai served as an Army officer from 1992 to 1994. He worked as a senior design engineer at Silicon Integrated Systems Corp from 1999 to 2001. He was, from 2001 to 2002, an assistant professor at the Department of Electrical Engineering, Tatung University, Taiwan. He was an assistant professor at the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan from 2002 to 2004. He was a visiting scholar in the Department of Electrical and Computer Engineering at the University of Texas at Austin from August 2010 to July 2011. He has since August 2004 been with National Taipei University, Taiwan where he is currently a professor of the Department of Communication Engineering.

His research interests include communication systems and signal processing.

Rong Zheng (S'03-M'04-SM'10) received her Ph.D. degree from Dept. of Computer Science, University of Illinois at Urbana-Champaign and earned her M.E. and B.E. in Electrical Engineering from Tsinghua University, P.R. China. She is on the faculty of the Department of Computing and Software, McMaster University. She was with University of Houston between 2004 and 2012. Rong Zheng's research interests include network monitoring and diagnosis, cyber physical systems, and sequential learning and decision theory. She received the National Science Foundation CAREER Award in 2006. She serves on the technical program committees of leading networking conferences including INFOCOM, ICDCS, ICNP, etc. She served as a guest editor for EURASIP Journal on Advances in Signal Processing, Special issue on wireless location estimation and tracking, Elsevlers Computer Communications Special Issue on Cyber Physical Systems; and Program co-chair of WASA'12 and CPSCom'12.

Pramod K. Varshney Pramod K. Varshney was born in Allahabad, India, on July 1, 1952. He received the B.S. degree in electrical engineering and computer science (with highest honors), and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign in 1972, 1974, and 1976 respectively.

During 1972-76, he held teaching and research assistantships at the University of Illinois. Since 1976 he has been with Syracuse University, Syracuse, NY where he is currently a Distinguished Professor of Electrical Engineering and Computer Science and the Director of CASE: Center for Advanced Systems and Engineering. He served as the Associate Chair of the department during 1993-96. He is also an Adjunct Professor of Radiology at Upstate Medical University in Syracuse, NY. His current research interests are in distributed sensor networks and data fusion, detection and estimation theory, wireless communications, physical layer security and image processing. He has published extensively. He is the author of Distributed Detection and Data Fusion, published by Springer-Verlag in 1997. He has served as a consultant to several major companies.

While at the University of Illinois, Dr. Varshney was a James Scholar, a Bronze Tablet Senior, and a Fellow. He is a member of Tau Beta Pi and is the recipient of the 1981 ASEE Dow Outstanding Young Faculty Award. He was elected to the grade of Fellow of the IEEE in 1997 for his contributions in the area of distributed detection and data fusion. He was the guest editor of the special issue on data fusion of the Proceedings of the IEEE, January 1997. In 2000, he received the Third Millennium Medal from the IEEE and Chancellor's Citation for exceptional academic achievement at Syracuse University. He is the recipient of the IEEE 2012 Judith A. Resnik Award and Doctor of Engineering *honoris causa* from Drexel University in 2014. He is on the editorial board of Journal on Advances in Information Fusion and IEEE Signal processing Magazine. He was the President of International Society of Information Fusion during 2001.