Optimal Construction of Regenerating Code Through Rate-Matching in Hostile Networks

Jian Li, Tongtong Li, *Senior Member, IEEE*, and Jian Ren, *Senior Member, IEEE*

***Abstract*— Regenerating code is a class of distributed storage codes that can optimally trade the bandwidth required to repair a failed node with the amount of data stored per node. There are two optimal points in the regeneration tradeoff curve: the minimum storage regeneration code and the minimum bandwidth regeneration code. However, in hostile networks where the storage nodes may be compromised, the storage capacity of the network can be significantly affected. In this paper, we propose two optimal regenerating code constructions through rate-matching to combat this kind of adversarial attacks in hostile networks. We first develop a two-layer rate-matched regenerating code construction. By matching the parameters of the full rate code and the partial rate code, we can optimize the overall storage efficiency while maintaining the corrupted node detection probability. Through comprehensive analysis, we show that the two-layer rate-matched regenerating code can achieve 70% higher storage efficiency than the universally resilient regenerating code. We then propose an optimal *m*-layer regenerating code construction. While the principle remains the same as the two-layer code, it is designed to optimize the total number of detectable corrupted nodes of *m* layers from which the errors can be corrected under the constraint of any given code efficiency. Compared with the universally resilient regenerating code with the same rate, our *m*-layer code can detect 50% more corrupted nodes.**

***Index Terms*— Optimal regenerating code, MDS code, error-correction, adversary attack.**

I. Introduction

DISTRIBUTED storage is a popular method to provide reliable data storage. Instead of storing a file and its replicas in multiple servers, we can break the file into components and store the components into multiple servers. In this way, while increasing data reliability, we can also achieve data confidentiality without data encryption and key management involved. A typical approach is to encode the file using an (n, k) Reed-Solomon (RS) code and distribute the encoded file into n servers. When we need to recover the file, we only need to collect the encoded parts from any k servers, which achieves a trade-off between reliability and efficiency. However, when repairing or regenerating the contents of a failed node, the whole file has to be recovered first, which is a waste of bandwidth.

The concept of regenerating code was introduced in [1], where a replacement node is allowed to connect to some individual nodes directly to regenerate a substitute of the failed node, instead of first recovering the original data then regenerating the failed component. Compared to the RS code, regenerating code achieves the optimal trade-off between bandwidth and storage within the minimum storage regeneration (MSR) and the minimum bandwidth regeneration (MBR) points.

However, when malicious behaviors exist in the network, both the regeneration of the failed node and the reconstruction of the original file could fail. The error resilience of the Reed-Solomon code based regenerating code in the network with errors and erasures was analyzed in [2]. In our previous work, a Hermitian code based regenerating code [3] was proposed to provide better error correction capability compared to the Reed-Solomon code based approach.

Inspired by the great performance improvement of the Hermitian code based regenerating codes, in this paper we move a step forward to construct optimal regenerating codes in distributed storage with structure similar to the Hermitian code.

The main contributions of this paper are:

- We propose an optimal construction of 2-layer rate-matched regenerating code. Both theoretical analysis and performance evaluation show that this code can achieve storage efficiency much higher than that of the universally resilient regenerating code proposed in [2].
- We propose an optimal construction of m-layer rate-matched regenerating code. The m-layer code can achieve optimal error correction capability, which is much higher than that of the code proposed in [2] and the Hermitian code based regenerating code proposed in [3]. Furthermore, the m-layered code is easier to understand and has more flexibility than the Hermitian code based regenerating code.

Here we will focus on malicious/compromised node locating from which the error can be corrected in distributed data regeneration and reconstruction. When no error occurs or no malicious node exists, the data regeneration and reconstruction should be processed the same as the existing works.

It is worth noting that although there are two types of regenerating codes: MSR and MBR codes on the MSR and MBR points, respectively, in this paper we will only focus

Manuscript received November 4, 2015; revised September 6, 2016 and December 6, 2016; accepted March 9, 2017. Date of publication April 14, 2017; date of current version June 14, 2017. This work was supported by the National Science Foundation under Grant CNS-1217206, Grant CNS-1232109, and Grant CNS-1524520. This paper was presented at the 2015 Resilience Week. *(Corresponding author: Jian Ren.)*

The authors are with the Department of ECE, Michigan State University, East Lansing, MI 48824-1226 USA (e-mail: lijian6@msu.edu; tongli@msu.edu; renjian@msu.edu).

Communicated by A. G. Dimakis, Associate Editor for Coding Techniques.

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.2017.2694441

on the optimization of the MSR code for the following two reasons:

1) The processes and results of the optimization for these two codes are similar. The optimization for the MSR code can be directly applied to the MBR code with similar optimization results.
2) The differences between the constructions of MSR and MBR codes have little impact on the optimization proposed in this paper.

The rest of this paper is organized as follows: in Section II, we introduce the related work. In Section III, the preliminary of this paper is presented. In Section IV, we propose two component codes for the rate-matched regenerating codes. We present and analyze the 2-layer rate-matched regenerating code in Section V. Then we propose and study the m-layer rate-matched regenerating code in Section VI. The paper is concluded in Section VII.

II. Related Work

When a storage node in the distributed storage network that employs the conventional (n, k) RS code (such as OceanStore [4] and Total Recall [5]) fails, the replacement node connects to k nodes and recovers the whole file to regenerate the symbols stored in the failed node. This approach is a waste of bandwidth because the whole file has to be downloaded to recover a fraction of it. To overcome this drawback, Dimakis *et al.* [1] introduced the conception of $\{n, k, d, \alpha, \beta, B\}$ regenerating code based on the network coding. In the context of regenerating code, the replacement node can regenerate the contents stored in a failed node by downloading γ help symbols from d helper nodes. The bandwidth consumption for the failed node regeneration could be far less than the whole file. A data collector (DC) can reconstruct the original file stored in the network by downloading α symbols from each of the k storage nodes. Dimakis *et al.* [1] proved that there is a trade-off between the bandwidth γ and per node storage α. They found two optimal points: minimum storage regeneration (MSR) and minimum bandwidth regeneration (MBR) points. The existing work has largely focused on the optimal regenerating codes design [6]–[18], and implementation of the regenerating code [19], [20].

The regenerating code can be divided into functional regeneration and exact regeneration. In the functional regeneration, the replacement node regenerates a new component that can functionally replace the failed component instead of being the same as the originally stored component. In [21], the data regeneration was formulated as a multicast network coding problem. The paper also constructed functional regenerating codes. A random linear regenerating code for distributed storage systems was implemented in [22]. It has been proved that by allowing data exchange among the replacement nodes, a better trade-off between repair bandwidth γ and per node storage α can be achieved [23]. Hou *et al.* [24] proposed a functional regenerating code with less computational complexity through binary operations. In the exact regeneration, the replacement node regenerates the exact symbols of a failed node. Wu and Dimakis [25] proposed to reduce the regeneration bandwidth through algebraic alignment. A code structure for exact regeneration using interference alignment technique was provided in [26]. In [27], an optimal exact constructions of MBR codes and MSR codes under product-matrix framework was presented. This is the first work that allows independent selection of the node number n in the network. In [28], repair performance of the Reed-Solomon codes was studied. A code construction that could achieve performance better than space-sharing between the minimum storage regenerating codes and the minimum bandwidth regenerating codes was proposed in [29].

However, none of these works considered code regeneration under node corruption or adversarial manipulation attacks in hostile networks. In fact, all these schemes will fail in both regeneration and reconstruction if some nodes in the storage cloud send out incorrect responses to the regeneration and reconstruction requests.

In [30], the Byzantine fault tolerance of regenerating codes was studied. The amount of information that can be safely stored against passive eavesdropping and active adversarial attacks based on the regeneration structure was discussed in [31]. To check data integrity of the regenerating code in hostile networks, CRC code was adopted in [32]. Unfortunately, the CRC checks can be easily manipulated by the malicious nodes, resulting in the failure of the regeneration and reconstruction. In [33], data integrity protection (DIP) was designed under a mobile Byzantine adversarial model to enable a client to verify the integrity of the outsourced data against general or malicious corruptions in distributed storage. Cachin and Tessaro [34] proposed to use erasure-coding and threshold cryptography to achieve storage efficiency and resilience. In [35], the verification cost for both the client read and write operations in workloads with idle periods was analyzed.

In [2], error resilience of the RS code based regenerating code in the network with both errors and erasures was evaluated. They provided the theoretical error correction capability. In [3], a Hermitian code based regenerating code was proposed. This code can provide better error correction capability under the same code efficiency. In [36], the universally secure regenerating code was developed to achieve information theoretic data confidentiality. However, the paper did not consider the extra computational cost and bandwidth for this code. Li *et al.* [37] proposed to apply linear feedback shift register (LFSR) to protect the data confidentiality. Tandon *et al.* [38] discussed the optimal trade-off between the storage space and the repair bandwidth in the presence of two types of wiretapper.

III. Preliminary and Assumptions

A. Regenerating Code

Regenerating code introduced in [1] is a linear code over finite field $\mathbb{F}_q$ with a set of parameters $\{n, k, d, \alpha, \beta, B\}$. A file of size B is stored in n storage nodes, each of which stores α symbols. A replacement node can regenerate the contents of a failed node by downloading β symbols from each of d randomly selected storage nodes. So the total bandwidth needed to regenerate a failed node is $\gamma = d\beta$. The data collector (DC) can reconstruct the whole file by downloading α symbols from each of $k \le d$ randomly selected storage nodes. In [1], the following theoretical bound was derived:

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

From equation (1), a trade-off between the regeneration bandwidth γ and the storage requirement α was derived. They cannot be decreased at the same time. There are two special cases: minimum storage regeneration (MSR) point in which the storage parameter α is minimized:

$$(\alpha_{MSR}, \gamma_{MSR}) = \left(\frac{B}{k}, \frac{Bd}{k(d-k+1)}\right), \tag{2}$$

and minimum bandwidth regeneration (MBR) point in which the bandwidth γ is minimized:

$$(\alpha_{MBR}, \gamma_{MBR}) = \left(\frac{2Bd}{2kd-k^2+k}, \frac{2Bd}{2kd-k^2+k}\right). \tag{3}$$

B. System Assumptions and Adversarial Model

In this paper, we assume there is a secure server that is responsible for encoding and distributing the data to storage nodes. The secure server will initialize the replacement nodes. The DC and the secure server can be implemented in the same computer. We assume that this server will never be compromised. We use the notation F/P to refer to either the full/partial rate MSR code or a codeword of the full/partial rate MSR code. The exact meaning can be clearly understood according to the context.

Our adversary model is the same as [2]. We assume that some network nodes may be corrupted due to hardware failure or communication errors, and/or compromised by malicious users which can take full control of up to $\tau \le n$ storage nodes and collude to perform attacks. Since our proposed codes work for all these cases, in the paper these nodes are referred to as corrupted nodes without distinguishing the specific error sources. As a result, upon request, these nodes may send out incorrect responses to disrupt the data regeneration and reconstruction.

The maximum number of corrupted nodes from which the errors can be corrected is referred to as the *error correction capability*.

IV. Component Codes of Rate-Matched Regenerating Code

In this section, we will introduce two different component codes for rate-matched MSR code on the MSR point with $d = 2k-2$. The code based on the MSR point with $d > 2k-2$ can be derived in the same way through truncating operations. In the rate-matched MSR code, there are two types of MSR codes with different code rates: full rate code and partial rate code.

A. Full Rate Code

1) Encoding: The full rate code $\{n, k, d, \alpha, \beta, B_{\mathsf{F}}\}$ is encoded based on the product-matrix code framework proposed in [27]. According to equation (2), we have $\alpha = d/2$, $\beta = 1$ for one block of data with the size $B_{\mathsf{F}} = k\alpha = (\alpha+1)\alpha$. The $d \times \alpha$ message matrix M_{F} is defined as

$$M_{\mathsf{F}} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix},$$

where S_1 and S_2 are $\alpha \times \alpha$ symmetric matrices, each of which will contain $B_{\mathsf{F}}/2$ data. We further define the $n \times d$ encoding matrix Ψ as $\Psi = [\Phi \;\, \Lambda\Phi]$, where

$$\Phi = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & g & g^2 & \cdots & g^{\alpha-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & g^{n-1} & (g^{n-1})^2 & \cdots & (g^{n-1})^{\alpha-1} \end{bmatrix} \tag{4}$$

is an $n \times \alpha$ Vandermonde matrix and $\Lambda = \mathrm{diag}[\lambda_1, \lambda_2, \cdots, \lambda_n]$ is an $n \times n$ diagonal matrix such that $\lambda_i \in \mathbb{F}_q$ and $\lambda_i \ne \lambda_j$ for $1 \le i, j \le n, i \ne j$, g is a primitive element in $\mathbb{F}_q$, and any d rows of Ψ are linearly independent.

The codeword F is defined as

$$\mathsf{F} = [\Phi \;\; \Lambda\Phi] \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \Psi M_{\mathsf{F}} = \begin{bmatrix} \mathsf{f}_1 \\ \vdots \\ \mathsf{f}_n \end{bmatrix}. \tag{5}$$

Each row $\mathsf{f}_i = \boldsymbol{\psi}_i M_{\mathsf{F}}$ $(1 \le i \le n)$ of the codeword matrix F will be stored in storage node i, where the encoding vector $\boldsymbol{\psi}_i$ is the i^{th} row of Ψ.

2) Regeneration: Suppose node z fails, the replacement node z' will send regeneration requests to the rest of the $n-1$ helper nodes. Upon receiving the regeneration request, helper node i will calculate and send out the help symbol $h_i = \mathsf{f}_i \boldsymbol{\phi}_z^T = \boldsymbol{\psi}_i M_{\mathsf{F}} \boldsymbol{\phi}_z^T$, where $\boldsymbol{\phi}_z$ is the z^{th} row of Φ and $\boldsymbol{\phi}_z^T$ is the transpose of $\boldsymbol{\phi}_z$. For $i \le j$, we define $\Psi_{i\to j} = \left[\boldsymbol{\psi}_i^T, \boldsymbol{\psi}_{i+1}^T \cdots, \boldsymbol{\psi}_j^T\right]^T$, and $\mathbf{x}^{(j)}$ to be the vector containing the first j symbols of $M_{\mathsf{F}}\boldsymbol{\phi}_z^T$ for convenience.

Suppose $h_i' = h_i + e_i$ is the response from helper node i. If $e_i \in \mathbb{F}_q \backslash \{0\}$, then node i is corrupted since the response h_i has been modified. We can successfully regenerate the symbols in node z when the total number of received help symbols h_i' being modified from the $n-1$ helper nodes is less than $\lfloor (n-d-1)/2 \rfloor$, where $\lfloor x \rfloor$ is the floor operation of x, which represents the greatest integer less than or equal to x. Without loss of generality, we assume $1 \le i \le n-1$. z' will perform Algorithm 1 to regenerate the contents of the failed node z.

Algorithm 1: z' regenerates symbols of the failed node z.

Step 1: Decode $\mathbf{h}'$ to $\mathbf{h}_{cw}$, where $\mathbf{h}' = [h_1', h_2', \cdots, h_{n-1}']^T$ can be viewed as an MDS code with parameters $(n-1, d, n-d)$ since $\Psi_{1\to(n-1)} \cdot \mathbf{x}^{(n-1)} = \mathbf{h}'$.

Step 2: Solve $\Psi_{1\to(n-1)} \cdot \mathbf{x}^{(n-1)} = \mathbf{h}_{cw}$ and compute $\mathsf{f}_z = \boldsymbol{\phi}_z S_1 + \lambda_z \boldsymbol{\phi}_z S_2$ as described in [27].

Proposition 1: For regeneration, the full rate code can correct errors from $\lfloor (n-d-1)/2 \rfloor$ corrupted nodes, where $\lfloor x \rfloor$ is the floor operation.

3) Reconstruction: When the DC needs to reconstruct the original file, it will send reconstruction requests to n storage nodes. Upon receiving the request, node i will send out the symbol vector $\mathbf{c}_i$ to the DC. Suppose $\mathbf{c}'_i = \mathbf{c}_i + \mathbf{e}_i$ is the response from storage node i. If $\mathbf{e}_i \in \mathbb{F}_q^{\alpha}\backslash\{\mathbf{0}\}$, then node i is corrupted since the response $\mathbf{c}_i$ has been modified.

The DC will reconstruct the file as follows: Let $R' = [\mathsf{f}'^T_1, \mathsf{f}'^T_2, \cdots, \mathsf{f}'^T_n]^T$, we have

$$\begin{aligned} R' &= \Psi \begin{bmatrix} S'_1 \\ S'_2 \end{bmatrix} = [\Phi \quad \Lambda\Phi] \begin{bmatrix} S'_1 \\ S'_2 \end{bmatrix}, \\ R'\Phi^T &= \Phi S'_1 \Phi^T + \Lambda \Phi S'_2 \Phi^T. \end{aligned} \tag{6}$$

Let $C = \Phi S'_1 \Phi^T$, $D = \Phi S'_2 \Phi^T$, and $\widehat{R}' = R'\Phi^T$, then

$$C + \Lambda D = \widehat{R}'. \tag{7}$$

Since C, D are both symmetric, we can solve the non-diagonal elements of C, D as follows:

$$\begin{cases} C_{i,j} + \lambda_i \cdot D_{i,j} = \widehat{R}'_{i,j} \\ C_{i,j} + \lambda_j \cdot D_{i,j} = \widehat{R}'_{j,i}. \end{cases} \tag{8}$$

Because matrices C and D have the same structure, here we only focus on C (corresponding to S'_1). It is straightforward to see that if node i is corrupted and there are errors in the i^{th} row of R', there will be errors in the i^{th} row of $\widehat{R}'$. Furthermore, there will be errors in the i^{th} row and i^{th} column of C.

Define $S'_1\Phi^T = \widehat{S}'_1$, we have $\Phi\widehat{S}'_1 = C$. We can view each column of C as an $(n-1, \alpha, n-\alpha)$ MDS code because Φ is a Vandermonde matrix. The length of the code is $n-1$ since the diagonal elements of C is unknown. Suppose node j is a legitimate node, we can decode the MDS code to recover the j^{th} column of C and locate the corrupted nodes. Eventually C can be recovered. So the DC can reconstruct S_1 using the method similar to [3] and [27]. For S_2, the recovering process is similar.

Proposition 2: For reconstruction, the full rate code can correct errors from $\lfloor (n-\alpha-1)/2 \rfloor$ corrupted nodes.

B. Partial Rate Code

1) Encoding: For the partial rate code, we also have $\alpha = d/2$, $\beta = 1$ for one block of data with the size

$$B_{\mathsf{P}} = \begin{cases} \frac{1}{2}xd(1+xd), & x \in (0, 0.5] \\ \frac{1}{2}\left(\alpha(\alpha+1)\right. \\ \left.+(x-0.5)d(1+(x-0.5)d)\right), & x \in (0.5, 1], \end{cases} \tag{9}$$

where x is the match factor of the rate-matched MSR code. It is easy to see that the partial rate code will become the full rate code when $x = 1$.

The data $\mathbf{m} = [m_1, m_2, \ldots, m_{B_{\mathsf{P}}}] \in \mathbb{F}_q^{B_{\mathsf{P}}}$ will be processed as follows:

- When $x \le 0.5$, the data will be arranged into a matrix S_1 of the size $\alpha \times xd$, where the first xd rows form a symmetric submatrix:

$$S_1 = \begin{bmatrix} m_1 & m_2 & \ldots & m_{xd} \\ m_2 & m_{xd+1} & \ldots & m_{2xd-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{xd} & m_{2xd-1} & \ldots & m_{B_{\mathsf{P}}} \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}. \tag{10}$$

The codeword P is defined as

$$\begin{aligned} \mathsf{P} &= \left[[\Phi \quad \Lambda\Phi] \begin{bmatrix} S_1 \\ \mathbf{0} \end{bmatrix} \middle\| \begin{bmatrix} r_{1,1} & r_{1,2} & \ldots & r_{1,\alpha-xd} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \ldots & r_{n,\alpha-xd} \end{bmatrix} \right] \\ &= [\,\Psi M_{\mathsf{P}} \parallel \mathbb{R}\,], \end{aligned} \tag{11}$$

where $\mathbf{0}$ is the $\alpha \times xd$ zero matrix, Φ, Λ, Ψ are the same as the full rate code, $r_{i,1}, r_{i,2}, \ldots, r_{i,\alpha-xd}$ $(1 \le i \le n)$ are random numbers generated by the secure server, $\mathbb{R}$ is the corresponding random number matrix, and $\|$ is the concatenation operator. Through the insertion of the random numbers, codeword of partial rate code with $x < 0.5$ will have the same appearance as the codeword of the full rate code. This can prevent the attackers from discriminating between the partial rate code and the full rate code. And the random numbers can be easily reproduced by the secure server for regeneration and reconstruction, making the additional overhead negligible. It is also interesting to point out that xd will be an integer in the optimal selection according to equation (20) and equation (21).

- When $x > 0.5$, the first $\alpha(\alpha+1)/2$ data will be arranged into an $\alpha \times \alpha$ symmetric matrix S_1. The rest of the data $m_{\alpha(\alpha+1)/2+1}, \ldots, m_{B_{\mathsf{P}}}$ will be arranged into another $\alpha \times \alpha$ symmetric matrix S_2:

$$S_2 = \begin{bmatrix} m_{\alpha(\alpha+1)/2+1} & \cdots & m_{\alpha(\alpha+1)/2+(x-0.5)d} & 0 & \ldots & 0 \\ m_{\alpha(\alpha+1)/2+2} & \cdots & m_{\alpha(\alpha+1)/2+2(x-0.5)d-1} & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ m_{\alpha(\alpha+1)/2+(x-0.5)d} & \cdots & m_{B_{\mathsf{P}}} & 0 & \ldots & 0 \\ 0 & \cdots & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \ldots & 0 \end{bmatrix}. \tag{12}$$

The codeword P is defined the same as equation (5) with the same parameters Φ, Λ and Ψ. Then each row p_i $(1 \le i \le n)$ of the codeword matrix P will be stored in storage node i respectively, in which the encoding vector $\boldsymbol{\psi}_i$ is the i^{th} row of Ψ.

Proposition 3: The partial rate code can achieve the MSR point in equation (2) since it is encoded under the product-matrix MSR code framework in [27].

2) Regeneration: The regeneration for the partial rate code is the same as the regeneration for the full rate code described in Section IV-A.2 with only a minor difference. If we define $\mathbf{x}^{(j)}$ as the vector containing the first j symbols of $M_{\mathsf{P}}\boldsymbol{\phi}_z^T$, there will be only xd nonzero elements in the vector. According to $\mathbf{\Psi}_{1\to n-1} \cdot \mathbf{x}^{(n-1)} = \mathbf{h}'$, the received symbol vector $\mathbf{h}'$ for the partial rate code in **Step 1** of Algorithm 1 can be viewed as an $(n-1, xd, n-xd)$ MDS code. Since $x < 1$, we can detect and correct more errors in data regeneration using the partial rate code than using the full rate code. For $x < 0.5$, the replacement node z' can eliminate the inserted random numbers for storage node i by subtracting $[\mathbf{0}, r_{i,1}, \dots, r_{i,\alpha-xd}]\boldsymbol{\phi}_z^T$ from the received help symbol, where $\mathbf{0}$ is the zero vector with length xd and $\boldsymbol{\phi}_z$ is the z^{th} row of Φ, before executing the regeneration algorithm.

Proposition 4: For regeneration, the partial rate code can correct errors from $\lfloor (n-xd-1)/2 \rfloor$ corrupted nodes.

3) Reconstruction: The reconstruction for the partial rate code is similar to that of the full rate code described in Section IV-A.3. Let $R' = [\mathsf{p}_1'^T, \mathsf{p}_2'^T, \cdots, \mathsf{p}_n'^T]^T$.

When the match factor $x > 0.5$, reconstruction for the partial rate code is the same as that of the full rate code.

When $x \le 0.5$, the inserted random numbers can be directly ignored. Equation (6) can be written as:

$$\Phi S_1' = R'. \tag{13}$$

So we can view each column of R' as an $(n, xd, n-xd+1)$ MDS code. After decoding R' to R_{cw}, we can recover the data matrix S_1 by solving the equation $\Phi S_1 = R_{cw}$. Meanwhile, if the i^{th} rows of R' and R_{cw} are different, we can mark node i as corrupted.

Proposition 5: For reconstruction, when the match factor $x > 0.5$, the partial rate code can correct errors from $\lfloor (n-\alpha-1)/2 \rfloor$ corrupted nodes. When the match factor $x \le 0.5$, the partial rate code can correct errors from $\lfloor (n-xd)/2 \rfloor$ corrupted nodes.

V. 2-Layer Rate-Matched Regenerating Code

In this section, we will show our first optimization of the rate-matched MSR code: 2-layer rate-matched MSR code. In the code design, we utilize two layers of the MSR code: the partial rate code for one layer and the full rate code for the other. The purpose of the partial rate code is to determine the optimized code efficiency while correcting the erroneous symbols sent by corrupted nodes and locating the corresponding corrupted nodes. Then we can treat the errors in the received symbols as erasures when regenerating with the full rate code. However, the rates of the two codes must match to achieve the optimal performance. Here we mainly focus on rate-matching for data regeneration. We can see in the later analysis that the performance of data reconstruction can also be improved with this design criterion.

We will first fix the error correction capabilities of the full rate code and the partial rate code. Then we will derive the rate matching criteria for optimal data storage efficiency under the fixed error correction capability.

A. Rate Matching

From the analysis above, we know that during data regeneration, the partial rate code can correct up to $\lfloor (n-xd-1)/2 \rfloor$ errors, which is more than the number of errors $\lfloor (n-d-1)/2 \rfloor$ that the full rate code can correct. In the 2-layer rate-matched MSR code design, our goal is to match the partial rate code with the full rate code. The main task for the partial rate code is to detect and correct errors, while the full rate code is to maintain the storage efficiency. So if the partial rate code can locate all the corrupted nodes, the full rate code can simply treat the symbols received from these corrupted nodes as erasures, which requires the minimum redundancy for the full rate code. The full rate code can correct up to $n-d-1$ erasures. Thus we have the following optimal rate-matching equation:

$$\lfloor (n-xd-1)/2 \rfloor = n-d-1, \tag{14}$$

from which we can derive the match factor x.

B. Encoding

To encode a file with size B using the 2-layer rate-matched MSR code, the file will first be divided into θ_{F} blocks of data with the size B_{F} and θ_{P} blocks of data with the size B_{P}, where the parameters should satisfy

$$B = \theta_{\mathsf{F}} B_{\mathsf{F}} + \theta_{\mathsf{P}} B_{\mathsf{P}}. \tag{15}$$

Then the θ_{F} blocks of data will be encoded into codeword matrices $\mathsf{F}_1, \dots, \mathsf{F}_{\theta_{\mathsf{F}}}$ using the full rate code and the θ_{P} blocks of data will be encoded into codeword matrices $\mathsf{P}_1, \dots, \mathsf{P}_{\theta_{\mathsf{P}}}$ using the partial rate code. To prevent the malicious nodes from corrupting the full rate code only, the secure server will randomly concatenate all the matrices together to form the final $n \times \alpha(\theta_{\mathsf{F}} + \theta_{\mathsf{P}})$ codeword matrix:

$$\mathcal{C} = [\text{Perm}(\mathsf{F}_1, \dots, \mathsf{F}_{\theta_{\mathsf{F}}}, \mathsf{P}_1, \dots, \mathsf{P}_{\theta_{\mathsf{P}}})], \tag{16}$$

where Perm denotes a random matrix permutation operation. The secure sever will also record the order of the permutation for future code regeneration and reconstruction. Then each row $\mathbf{c}_i = [\text{Perm}(\mathsf{f}_{1,i}, \dots, \mathsf{f}_{\theta_{\mathsf{F}},i}, \mathsf{p}_{1,i}, \dots, \mathsf{p}_{\theta_{\mathsf{P}},i})]$ $(1 \le i \le n)$ of the codeword matrix $\mathcal{C}$ will be stored in storage node i, where $\mathsf{f}_{j,i}$ is the i^{th} row of F_j $(1 \le j \le \theta_{\mathsf{F}})$, and $\mathsf{p}_{j,i}$ is the i^{th} row of P_j $(1 \le j \le \theta_{\mathsf{P}})$. The encoding vector $\boldsymbol{\psi}_i$ for storage node i is the i^{th} row of Ψ in equation (5). Therefore, we have the following Theorem.

Theorem 1: The encoding of 2-layer rate-matched MSR code can achieve the MSR point in equation (2) since both the full rate code and the partial code are MSR codes.

Remark 1: The permutation operation is designed to prevent the adversaries from identifying the full rate code. For the application scenarios where the errors are caused by hardware failures or communication errors, we can directly concatenate all the codeword matrices without the permutation operation.

C. Regeneration

Suppose node z fails, the security server will initialize a replacement node z' with the permutation information of the partial rate code and the full rate code in the 2-layer rate-matched MSR code. Then the replacement node z' will send regeneration requests to the rest of the $n-1$ helper nodes. Upon receiving the regeneration request, helper node i will calculate and send out the help symbols $\text{Perm}(\mathsf{f}_{1,i}\boldsymbol{\phi}_z^T, \dots, \mathsf{f}_{\theta_{\mathsf{F}},i}\boldsymbol{\phi}_z^T,$

$\mathsf{p}_{1,i}\boldsymbol{\phi}_z^T, \dots, \mathsf{p}_{\theta_{\mathsf{P}},i}\boldsymbol{\phi}_z^T)$. z' will perform Algorithm 2 to regenerate the contents of the failed node z. After the regeneration is finished, z' will erase the permutation information immediately. So even if z' was compromised later, the adversary would not get the permutation order of the partial rate code and the full rate code.

Algorithm 2: z' regenerates symbols of the failed node z for the 2-layer rate-matched MSR code.

Step 1: According to the permutation information, regenerate all the symbols related to the θ_{P} data blocks encoded by the partial rate code. If errors are detected in the symbols sent by node i, it will be marked as a corrupted node.

Step 2: Regenerate all the symbols related to the θ_{F} data blocks encoded by the full rate code. During the regeneration, all the symbols sent from nodes marked as corrupted nodes will be replaced by erasures $\otimes$.

It is easy to see that Algorithm 2 can correct errors and locate corrupted nodes using the partial rate code while achieving high storage efficiency using the full rate code. We summarize the result as the following Theorem.

Theorem 2: For regeneration, the 2-layer rate-matched MSR code can correct errors from $\lfloor (n - xd - 1)/2 \rfloor$ corrupted nodes.

An illustrative example of the 2-layer rate-matched MSR code with parameters $n = 7, d = 4, x = 1/2$ is shown in Fig. 1. In this example, there are two malicious nodes (Node 2 and Node 5) which will send manipulated responses for the data regeneration of Node 1. According to Proposition 4, during the regeneration of Node 1, the partial rate code can detect and correct the errors in the responses of Node 2 and Node 5. With the malicious nodes information provided by the partial rate code, the full rate code could be regenerated correctly thereafter. Through this design we could get an optimal trade-off between the error correction capability and the storage efficiency, while for the universally resilient MSR code, improving the error correction capability will cause a much lower storage efficiency.

D. Parameter Optimization

We have the following design requirements for a given distributed storage system applying the 2-layer rate-matched MSR code:

- The maximum number of corrupted nodes τ that the system can detect and locate using the partial rate code. We have
$$\lfloor (n - xd - 1)/2 \rfloor = \tau. \tag{17}$$
- We use $\mathcal{P}_{\text{det}}$ to represent the probability that the system can detect all the corrupted nodes. The detection will be successful if each corrupted node modifies at least one help symbol corresponding to the partial rate code and sends it to the replacement node. Suppose the probability with which each help symbol is modified by either errors or malicious manipulations is $\mathcal{P}$, then we have
$$\left(1 - (1 - \mathcal{P})^{\theta_{\mathsf{P}}}\right)^{\tau} \geq \mathcal{P}_{\text{det}}. \tag{18}$$

Define the storage efficiency δ_S as the ratio between the actual size of data to be stored and the total storage space needed by the encoded data. Then we have:
$$\delta_S = \frac{\theta_{\mathsf{F}} B_{\mathsf{F}} + \theta_{\mathsf{P}} B_{\mathsf{P}}}{(\theta_{\mathsf{F}} + \theta_{\mathsf{P}})n\alpha} = \frac{B}{(\theta_{\mathsf{F}} + \theta_{\mathsf{P}})n\alpha}. \tag{19}$$

There is a trade-off between θ_{P} the number of data blocks encoded by the partial rate code and θ_{F} the number of data blocks encoded by the full rate code. If we encode using too much full rate code, we may not meet the detection probability $\mathcal{P}_{\text{det}}$ requirement. If we employ too much partial rate code, the redundancy of the code may be too high. We can calculate the optimized parameters $x, d, \theta_{\mathsf{F}}, \theta_{\mathsf{P}}$ by maximizing equation (19) under the constraints defined by equations (14), (15), (17), (18). That is:

$$\begin{aligned}
&\text{maximize equation (19)}: \ \delta_S = \frac{B}{(\theta_{\mathsf{F}} + \theta_{\mathsf{P}})n\alpha},\\
&\text{subject to equation (14)}: \ \lfloor (n - xd - 1)/2 \rfloor = n - d - 1,\\
&\quad \text{equation (15)}: \ B = \theta_{\mathsf{F}} B_{\mathsf{F}} + \theta_{\mathsf{P}} B_{\mathsf{P}},\\
&\quad \text{equation (17)}: \ \lfloor (n - xd - 1)/2 \rfloor = \tau,\\
&\quad \text{equation (18)}: \ \left(1 - (1 - \mathcal{P})^{\theta_{\mathsf{P}}}\right)^{\tau} \geq \mathcal{P}_{\text{det}}.
\end{aligned}$$

For this optimization, d and x can be determined by equation (14) and (17):
$$d = n - \tau - 1, \tag{20}$$
$$x = (n - 2\tau - 1)/(n - \tau - 1). \tag{21}$$

Since B is constant, to maximize δ_S is equivalent to minimize $\theta_{\mathsf{F}} + \theta_{\mathsf{P}}$. So we can rewrite the optimization problem as follows:

$$\begin{aligned}
&\text{minimize } \theta_{\mathsf{F}} + \theta_{\mathsf{P}},\\
&\text{subject to equation (15): } B = \theta_{\mathsf{F}} B_{\mathsf{F}} + \theta_{\mathsf{P}} B_{\mathsf{P}},\\
&\text{equation (18): } \left(1 - (1 - \mathcal{P})^{\theta_{\mathsf{P}}}\right)^{\tau} \geq \mathcal{P}_{\text{det}}.
\end{aligned} \tag{22}$$

This is a simple linear programming problem. It is straightforward to derive the optimization results directly:
$$\theta_{\mathsf{P}} = \log_{(1-\mathcal{P})}\left(1 - \mathcal{P}_{\text{det}}^{1/\tau}\right), \tag{23}$$
$$\theta_{\mathsf{F}} = (B - \theta_{\mathsf{P}} B_{\mathsf{P}})/B_{\mathsf{F}}. \tag{24}$$

In this paper we assume that we are storing large files, which means $B > \theta_{\mathsf{P}} B_{\mathsf{P}}$. So an optimal solution for the 2-layer rate-matched MSR code can always be found. We have the following theorem:

Theorem 3: When the number of blocks of the partial rate code θ_{P} equals to $\log_{(1-\mathcal{P})}\left(1 - \mathcal{P}_{\text{det}}^{1/\tau}\right)$ and the number of blocks of the full rate code θ_{F} equals to $(B - \theta_{\mathsf{P}} B_{\mathsf{P}})/B_{\mathsf{F}}$, the 2-layer rate-matched MSR code can achieve the optimal storage efficiency.

E. Reconstruction

When DC needs to reconstruct the original file, it will send reconstruction requests to n storage nodes. Upon receiving the request, node i will send out the symbol vector $\mathbf{c}_i$. Suppose $\mathbf{c}'_i = \mathbf{c}_i + \mathbf{e}_i$ is the response from the i^{th} storage node. If $\mathbf{e}_i \in \mathbb{F}_q^{\alpha(\theta_{\mathsf{P}}+\theta_{\mathsf{F}})} \backslash \{\mathbf{0}\}$, it means the response $\mathbf{c}_i$ has been modified, therefore, the node i is corrupted. Since DC has

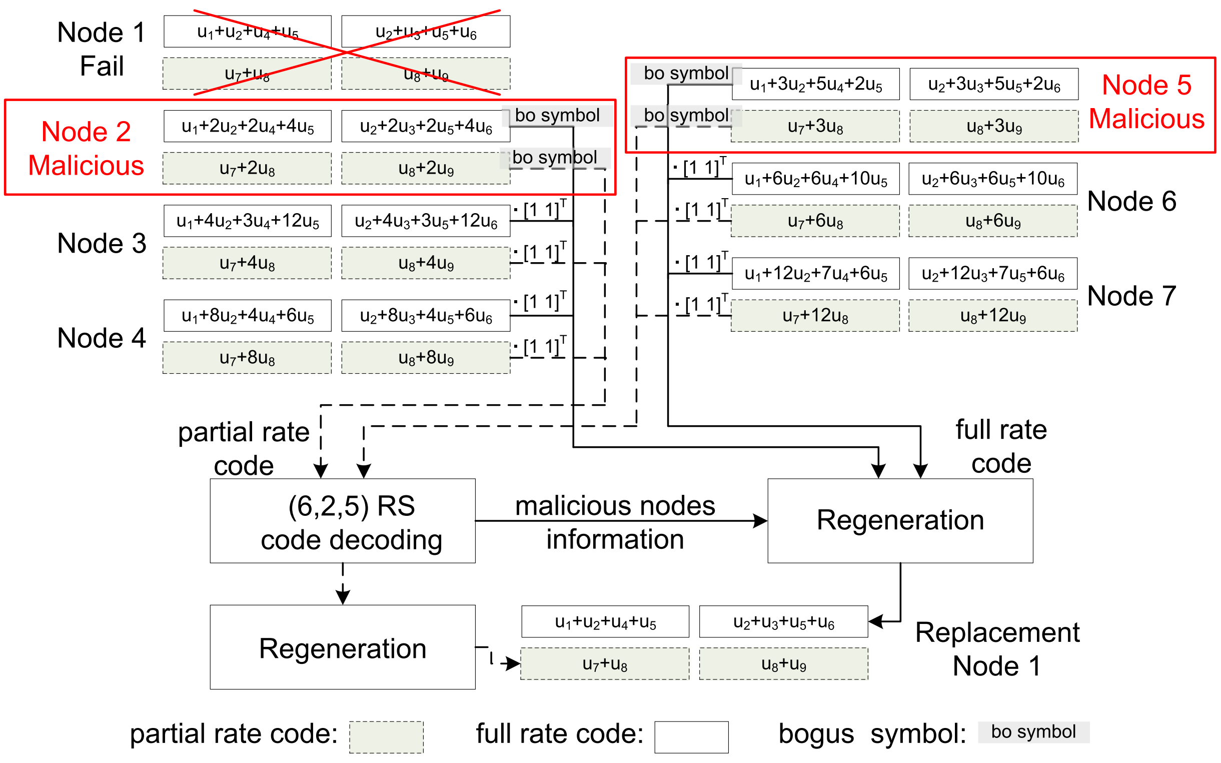

Fig. 1. An illustrative example of the 2-layer rate-matched MSR code

the permutation information of the partial rate code and the full rate code, similar to the regeneration of the 2-layer rate-matched MSR code, DC will perform reconstruction using Algorithm 3.

Algorithm 3: DC reconstructs the original file for the 2-layer rate-matched MSR code.

Step 1: According to the permutation information, reconstruct each of the θ_P data blocks encoded by the partial rate code and locate the corrupted nodes.

Step 2: Reconstruct each of the data blocks encoded by the full rate code. During the reconstruction, all the symbols sent from corrupted nodes will be replaced by erasures $\otimes$.

In Section V-D, we optimize the parameters for the data regeneration, considering the trade-off between the successful corrupted node detection probability and the storage efficiency. For data reconstruction, we have the following theorem:

Theorem 4 (Optimal Parameters): When the number of blocks of the partial rate code θ_P equals to $\log_{(1-\mathcal{P})}\left(1-\mathcal{P}_{\text{det}}^{1/\tau}\right)$ and the number of blocks of the full rate code θ_F equals to $(B-\theta_P B_P)/B_F$, the 2-layer rate-matched MSR code can guarantee that the same constraints for data regeneration (equation (17) and equation (18)) be satisfied for the data reconstruction.

Proof: The maximum number of corrupted nodes that can be detected for data reconstruction is calculated as follows:

- If $x > 0.5$, the number is $\lfloor (n-\alpha-1)/2 \rfloor$. We have $\lfloor (n-\alpha-1)/2 \rfloor \geq \lfloor (n-xd-1)/2 \rfloor = \tau$.
- If $x \leq 0.5$, the number is $\lfloor (n-xd)/2 \rfloor$. We have $\lfloor (n-xd)/2 \rfloor \geq \lfloor (n-xd-1)/2 \rfloor = \tau$.

Therefore, in both cases, we can detect the maximum number of corrupted nodes τ.

The probability for corrupted node to be detected successfully in data reconstruction can be calculated as:

$$\left(1-(1-\mathcal{P})^{\alpha\theta_P}\right)^{\tau} > \left(1-(1-\mathcal{P})^{\theta_P}\right)^{\tau} \geq \mathcal{P}_{\text{det}}.$$

□

Although the rate-matching equation (14) does not apply to the data reconstruction, the reconstruction strategy in Algorithm 3 can still benefit from the different rates of the two codes. When $x \leq 0.5$, the partial rate code can detect and correct $\lfloor (n-xd)/2 \rfloor$ corrupted nodes, which are more than $\lfloor (n-d/2-1)/2 \rfloor$ corrupted nodes that the full rate code can detect. When $x > 0.5$, the full rate code and the partial rate code can detect and correct the same number of corrupted nodes: $\lfloor (n-\alpha-1)/2 \rfloor$.

From the analysis above we can see that the optimized parameters obtained for the data regeneration can also achieve the optimized trade-off between the corrupted node detection and storage efficiency for data reconstruction.

F. Performance Evaluation

From the analysis above, we know that for a distributed storage system with n storage nodes, out of which at most τ nodes are corrupted, the 2-layer rate-matched MSR code can guarantee detection and correction of these nodes during data regeneration and reconstruction with the probability at least $\mathcal{P}_{\text{det}}$.

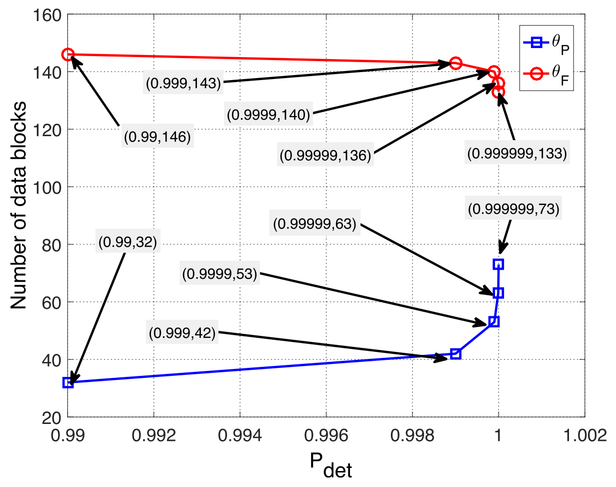

Fig. 2. The number of partial/full rate code blocks for different $\mathcal{P}_{\text{det}}$

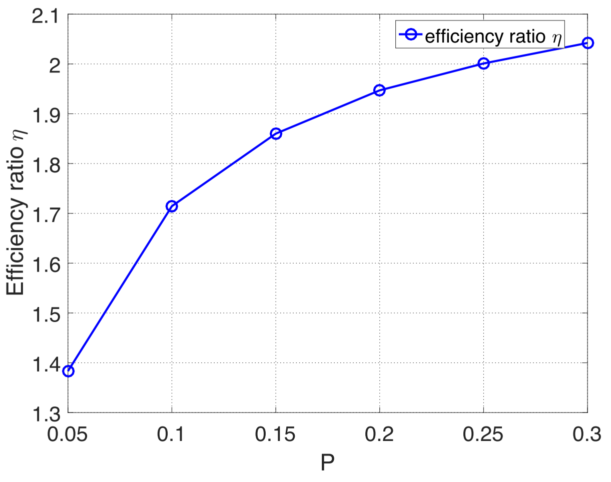

Fig. 4. Efficiency ratios between the 2-layer rate-matched MSR code and the universally resilient MSR code for different $\mathcal{P}$

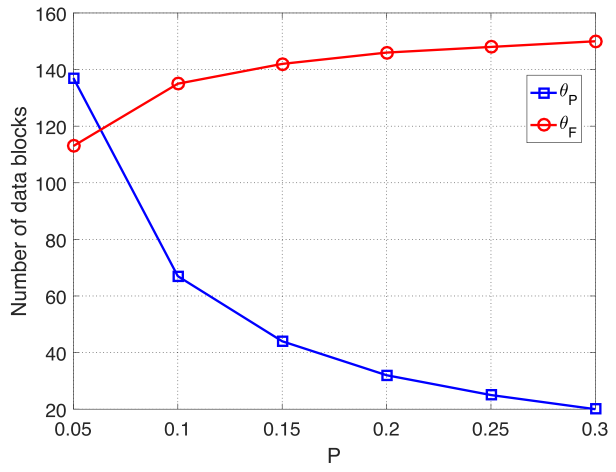

Fig. 3. The number of partial/full rate code blocks for different $\mathcal{P}$

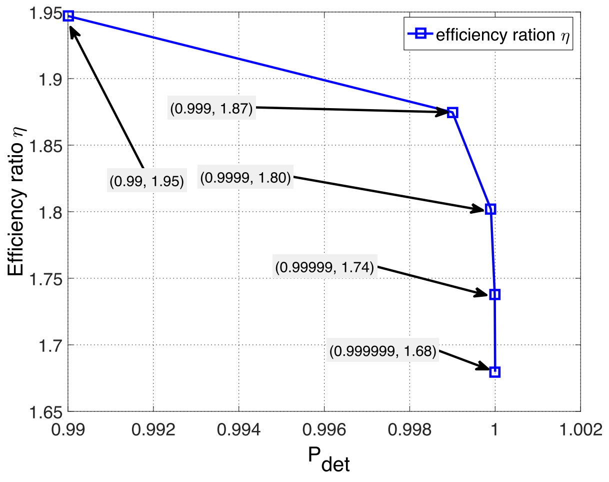

Fig. 5. Efficiency ratios between the 2-layer rate-matched MSR code and the universally resilient MSR code for different P_{det}

As an example, for a distributed storage system with $n = 30$, $\tau = 11$ and $\mathcal{P} = 0.2$, suppose we have a file with the size $B = 14000$ symbols to be stored in the system. According to the parameter optimization discussed above, we have the match factor $x = 7/18$, partial rate code block size $B_{\mathsf{P}} = 28$ and full rate code block size $B_{\mathsf{F}} = 90$. The number of the partial rate code blocks θ_{P} and the number of the full rate code blocks θ_{F} for different detection probabilities $\mathcal{P}_{\text{det}}$ are shown in Fig. 2. From the figure we can see that the number of partial rate code blocks will increase when the detection probability becomes larger. Accordingly, the number of full rate code blocks will decrease.

In Fig. 3, the number of the partial rate code blocks θ_{P} and the number of the full rate code blocks θ_{F} for different symbol corruption probabilities $\mathcal{P}$ and fixed detection probability $\mathcal{P}_{\text{det}} = 0.99$ are shown. The number of partial rate code blocks will decrease when the symbol corruption probability becomes larger.

To compare the performance of the 2-layer rate-matched MSR code with the universally resilient MSR code constructed in [2], the storage efficiency of the universally resilient MSR code with the same regeneration performance (error correction capability during regeneration) as the 2-layer rate-match MSR code can be calculated as

$$\delta'_S = \frac{\alpha'(\alpha'+1)}{\alpha' n} = \frac{\alpha'+1}{n} = \frac{xd/2+1}{n}, \tag{25}$$

where α' is the regeneration parameter α of the universally resilient MSR code. In Fig. 4 we show the efficiency ratios $\eta = \delta_S/\delta'_S$ between the 2-layer rate-matched MSR code and the universally resilient MSR code under different corruption probabilities $\mathcal{P}$. From the figure we can see that the 2-layer rate-matched MSR code has a higher efficiency than the universally resilient MSR code. In fact, even when the corruption probability is $\mathcal{P} = 0.05$, the efficiency of the 2-layer rate-matched MSR code is about 40% higher than the universally resilient MSR code. In Fig. 5 we also show the efficiency ratios under different detection probabilities $\mathcal{P}_{\text{det}}$. When the successfully corrupted nodes detection probability is 0.999999, the efficiency of the 2-layer rate-matched MSR code is about 70% higher than the universally resilient MSR code.

VI. m-Layer Rate-Matched Regenerating Code

In this section, we will show our second optimization of the rate-matched MSR code: m-layer rate-matched MSR code. In the code design, we extend the design concept of the 2-layer rate-matched MSR code to any m layers. Instead of encoding the data using two MSR codes with different match factors, we utilize m layers of the full rate MSR codes with different parameters d's, written as d_i for layer L_i, $1 \le i \le m$, which satisfy

$$d_i \le d_j, \quad \forall 1 \le i \le j \le m. \tag{26}$$

The data will be divided into m parts and each part will be encoded by a distinct full rate MSR code. According to the analysis above, the code with a lower code rate has better error correction capability. The codewords will be decoded layer by layer in the order from layer L_1 to layer L_m. That is, the codewords encoded by the full rate MSR code with a lower d will be decoded prior to those encoded by the full rate MSR code with a higher d for both regeneration and reconstruction. If errors were found by the full rate MSR code with a lower d, the corresponding nodes would be marked as corrupted. The symbols sent from these nodes would be treated as erasures in the subsequent decoding of the full rate MSR codes with higher d's. The purpose of this arrangement is to locate as many corrupted nodes as possible using full rate MSR codes with lower rates and correct the corresponding erroneous symbols using the full rate MSR codes with higher rates. However, the rates of the m-layer MSR codes must match to achieve an optimal performance. Here we mainly focus on rate-matching for data regeneration. We can see in the later analysis that the performance of data reconstruction can also be improved with this design criterion.

In summary, the main idea of this optimization is to optimize the overall error correction capability by matching the code rates of different full rate MSR codes.

A. Rate Matching and Parameter Optimization

According to Section IV-A.2, the full rate MSR code F_i for layer L_i can be viewed as an $(n-1, d_i, n-d_i)$ MDS code for $1 \le i \le m$ during regeneration. In the optimization, we set the summation of the d's of all the layers to a constant d_0:

$$\sum_{i=1}^{m} d_i = d_0. \tag{27}$$

Here we will show the optimization through an illustrative example first. Then we will present the general result.

1) Optimization for $m = 3$: There are three layers of full rate MSR codes for F_1, F_2 and F_3.

The first layer code F_1 can correct t_1 errors:

$$t_1 = \lfloor (n-d_1-1)/2 \rfloor = (n-d_1-1-\varepsilon_1)/2, \tag{28}$$

where $\varepsilon_1 = 0$ or 1 depending on whether $(n-d_1-1)/2$ is even or odd.

By treating the symbols from the t_1 nodes where errors are found by F_1 as erasures, the second layer code F_2 can correct t_2 errors [39]:

$$\begin{aligned} t_2 &= \lfloor (n-d_2-1-t_1)/2 \rfloor + t_1 \\ &= (n-d_2-1-t_1-\varepsilon_2)/2 + t_1 \\ &= (2(n-d_2)+n-d_1-2\varepsilon_2-\varepsilon_1-3)/4, \end{aligned} \tag{29}$$

where $\varepsilon_2 = 0$ or 1, with the restriction that $n-d_2-1 \ge t_1$, which can be written as:

$$-d_1 + 2d_2 \le n + \varepsilon_1 - 1. \tag{30}$$

The third layer code F_3 also treat the symbols from the t_2 nodes as erasures. F_3 can correct t_3 errors:

$$\begin{aligned} t_3 &= \lfloor (n-d_3-1-t_2)/2 \rfloor + t_2 \\ &= (n-d_3-1-t_2-\varepsilon_3)/2 + t_2 \\ &= (4(n-d_3)+2(n-d_2)+n-d_1-4\varepsilon_3-2\varepsilon_2-\varepsilon_1-7)/8, \end{aligned} \tag{31}$$

where $\varepsilon_3 = 0$ or 1, with the restriction that $n-d_3-1 \ge t_2$, which can be written as:

$$-d_1 - 2d_2 + 4d_3 \le n + \varepsilon_1 + 2\varepsilon_2 - 1. \tag{32}$$

According to the analysis above, the d's of the three layers satisfy:

$$d_1 - d_2 \le 0, \tag{33}$$

$$d_2 - d_3 \le 0. \tag{34}$$

To maximize the error correction capability of the m-layer rate-matched MSR code for $m = 3$, we have to maximize t_3, the number of errors that the third layer code F_3 can correct, since t_3 has included all the corrupted nodes from which errors are found by the codes of the first two layers. With all the constraints listed above, the optimization problem can be written as:

$$\begin{aligned} \text{maximize}\quad & \text{equation (31)}: t_3 = (4(n-d_3)+2(n-d_2)+n \\ & \qquad -d_1-4\varepsilon_3-2\varepsilon_2-\varepsilon_1-7)/8, \\ \text{subject to}\quad & \text{equation (27)}: d_1+d_2+d_3 = d_0, \\ & \text{equation (33)}: d_1-d_2 \le 0, \\ & \text{equation (34)}: d_2-d_3 \le 0, \\ & \text{equation (30)}: -d_1+2d_2 \le n+\varepsilon_1-1, \\ & \text{equation (32)}: -d_1-2d_2+4d_3 \le n+\varepsilon_1 \\ & \qquad +2\varepsilon_2-1. \end{aligned} \tag{35}$$

We can define slack variables $s_1, s_2, \ldots, s_7$ and establish the following linear equations:

$$\begin{aligned} d_3 &= d_0 - d_1 - d_2, \\ s_1 &= d_2 - d_1, \\ s_2 &= d_3 - d_2 = d_0 - 2d_2 - d_1, \\ s_3 &= n - 1 - 2d_2 + d_1 + \varepsilon_1, \\ s_4 &= n - 1 - 4d_0 + 6d_2 + 5d_1 + 2\varepsilon_2 + \varepsilon_1, \\ s_5 &= n - 1 - d_1 - \varepsilon_1, \\ s_6 &= (n - 1 - 2d_2 + d_1 - 2\varepsilon_2 + \varepsilon_1)/2, \\ s_7 &= (n - 1 - 4d_0 + 6d_2 + 5d_1 - 4\varepsilon_3 + 2\varepsilon_2 + \varepsilon_1)/4, \\ t_3 &= (7n - 7 - 4d_0 + 2d_2 + 3d_1 - 4\varepsilon_3 - 2\varepsilon_2 - \varepsilon_1)/8. \end{aligned} \tag{36}$$

Apparently this linear programming is feasible with a basic feasible solution (BFS). We solve it using the SIMPLEX algorithm [40]. To maximize t_3, we can increase d_1 in equation (36), which is a Gaussian elimination of d_1 in t_3. It is easy to know that as long as d_1 does not exceed d_2, then after the substitution is complete, we will have a new dictionary and a new improved BFS. The updated linear system can be expressed as follows:

$$\begin{aligned}
d_3 &= d_0 - d_1 - d_2,\\
d_1 &= d_2 - s_1,\\
s_2 &= d_0 - 3d_2 + s_1,\\
s_3 &= n - 1 - d_2 + \varepsilon_1 - s_1,\\
s_4 &= n - 1 - 4d_0 + 11d_2 + 2\varepsilon_2 + \varepsilon_1 - 5s_1,\\
s_5 &= n - 1 - d_2 - \varepsilon_1 + s_1,\\
s_6 &= (n - 1 - d_2 - 2\varepsilon_2 + \varepsilon_1 - s_1)/2,\\
s_7 &= (n - 1 - 4d_0 + 11d_2 - 4\varepsilon_3 + 2\varepsilon_2 + \varepsilon_1 - 5s_1)/4,\\
t_3 &= (7n - 7 - 4d_0 + 5d_2 - 4\varepsilon_3 - 2\varepsilon_2 - \varepsilon_1 + 3s_1)/8.
\end{aligned} \tag{37}$$

Repeat this process for d_2, we get

$$\begin{aligned}
d_3 &= d_0 - d_1 - d_2,\\
d_1 &= d_2 - s_1,\\
d_2 &= (d_0 - s_2 + s_1)/3,\\
s_3 &= (3n - 3 - d_0 + 3\varepsilon_1 + s_2 - 4s_1)/3,\\
s_4 &= (3n - 3 - d_0 + 6\varepsilon_2 + 3\varepsilon_1 - 11s_2 - 4s_1)/3,\\
s_5 &= (3n - 3 - d_0 - 3\varepsilon_1 + s_2 + 2s_1)/3,\\
s_6 &= (3n - 3 - d_0 - 6\varepsilon_2 + 3\varepsilon_1 + s_2 - 4s_1)/6,\\
s_7 &= (3n - 3 - d_0 - 12\varepsilon_3 + 6\varepsilon_2 + 3\varepsilon_1 - 11s_2 - 4s_1)/12,\\
t_3 &= (21n - 21 - 7d_0 - 12\varepsilon_3 - 6\varepsilon_2 - 3\varepsilon_1 - 5s_2 - 4s_1)/24.
\end{aligned} \tag{38}$$

Since all the coefficients of t_3 are negative, the value of t_3 cannot be further increased. Therefore, this is the optimal value of t_3. The corresponding BFS is $s_1 = s_2 = 0$, $d_1 = d_2 = d_3 = \text{round}(d_0/3) = \widetilde{d}$, and the m-layer rate-matched MSR code can correct errors from at most

$$\begin{aligned}
\widetilde{t_3} &= (7n - 7\widetilde{d} - 4\varepsilon_3 - 2\varepsilon_2 - \varepsilon_1 - 7)/8\\
&\geq (7n - 7\widetilde{d} - 14)/8 \text{ (worst case)}
\end{aligned} \tag{39}$$

corrupted nodes, where round is the rounding operation.

2) Evaluation of the Optimization for $m = 3$: The universally resilient MSR code with the same code rate can be viewed as an $(n-1, \widetilde{d}, n-\widetilde{d})$ MDS code which can correct errors from at most $(n - \widetilde{d} - 1)/2$ corrupted nodes (best case) during regeneration. The comparison of the error correction capability between m-layer rate-matched MSR code for $m = 3$ and universally resilient MSR code is shown in Fig. 6. In this comparison, we set the number of storage nodes in the network as $n = 30$. From the figure we can see that the m-layer rate-matched MSR code for $m = 3$ improves the error correction capability more than 50%.

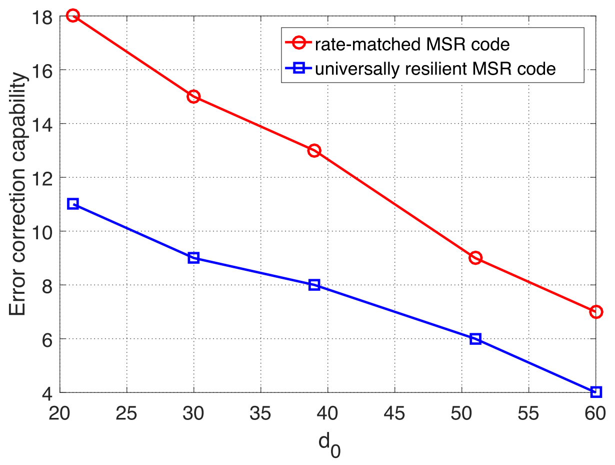

Fig. 6. Comparison of the error correction capability between m-layer rate-matched MSR code for $m = 3$ and universally resilient MSR code

3) General Optimal Result: For the general m-layer rate-matched MSR code, the optimization process is similar.

For the m-layer rate-matched MSR code, we have:

$$\sum_{i=1}^{m} d_i = d_0, \tag{40}$$

and

$$d_{i-1} - d_i \leq 0, \quad \text{for } 2 \leq i \leq m. \tag{41}$$

The first layer code F_1 can correct t_1 errors as in equation (28). For $2 \leq i \leq m$, by treating the symbols from the t_{i-1} nodes where errors are found by F_{i-1} as erasures, the i^{th} layer code can correct t_i errors:

$$\begin{aligned}
t_i &= \lfloor (n - d_i - 1 - t_{i-1})/2 \rfloor + t_{i-1}\\
&= (n - d_i - 1 - t_{i-1} - \varepsilon_i)/2 + t_{i-1}\\
&= \left(\sum_{j=1}^{i} 2^{j-1}(n - d_j) - \sum_{j=1}^{i} 2^{j-1}\varepsilon_j - 2^i + 1 \right) / 2^i,
\end{aligned} \tag{42}$$

where $\varepsilon_i = 0$ or 1, with the restriction that $n - d_i - 1 \geq t_{i-1}$, which can be written as:

$$-\sum_{j=1}^{i-1} 2^{j-1} d_j + 2^{i-1} d_i \leq n + \sum_{j=1}^{i-1} 2^{j-1}\varepsilon_j - 1. \tag{43}$$

We can maximize the error correction capability of the m-layer rate-matched MSR code by maximizing t_m. With all the constrains listed above, the optimization problem can be written as:

$$\begin{aligned}
\text{maximize} \quad & \text{equation (42)}: t_m = \left(\sum_{i=1}^{m} 2^{i-1}(n - d_i) \right.\\
& \qquad \left. - \sum_{i=1}^{m} 2^{i-1}\varepsilon_i - 2^m + 1 \right) / 2^m,\\
\text{subject to} \quad & \text{equation (40)}: \sum_{i=1}^{m} d_i = d_0,\\
& \text{equation (41)}: d_{i-1} - d_i \leq 0, \; 2 \leq i \leq m,
\end{aligned}$$

$$\text{equation (43)}: -\sum_{j=1}^{i-1} 2^{j-1}d_j + 2^{i-1}d_i \le n + \sum_{j=1}^{i-1} 2^{j-1}\varepsilon_j - 1, \quad 2 \le i \le m. \tag{44}$$

For this linear programming problem, the optimization result can be summarized as follows:

Theorem 5: For the regeneration of m-layer rate-matched MSR code, when

$$d_i = \text{round}(d_0/m) = \widetilde{d} \text{ for } 1 \le i \le m, \tag{45}$$

it can correct errors from at most

$$\begin{aligned}\widetilde{t}_m &= \left((2^m-1)(n-\widetilde{d}-1) - \sum_{i=1}^{m} 2^{i-1}\varepsilon_i\right)/2^m \\ &\ge \left((2^m-1)(n-\widetilde{d}-2)\right)/2^m \text{ (worst case)}\end{aligned} \tag{46}$$

corrupted nodes.

Proof: The proof of this theorem is very similar to $m = 3$. In fact, in the SIMPLEX algorithm process described from equation (36) to equation (38), only the constraints $d_0 = \sum_{i=1}^{m} d_i$ and $d_i \le d_j$ for $i \le j$ have been directly used. Therefore, we can define the following linear equation systems:

$$\begin{aligned} d_m &= d_0 - \sum_{i=1}^{m-1} d_i, \\ s_1 &= d_2 - d_1, \\ s_2 &= d_3 - d_2, \\ &\vdots \\ s_{m-1} &= d_m - d_{m-1} = d_0 - md_{m-1} + \sum_{i=1}^{m-2} is_i, \\ t_m &= \Bigg((2^m-1)(n-1) - 2^{m-1}d_0 \\ &\quad + \sum_{i=1}^{m-1}(2^{m-1}-2^{i-1})d_i - \sum_{i=1}^{m} 2^{i-1}\varepsilon_i\Bigg)/2^m. \end{aligned} \tag{47}$$

Since the coefficient of d_1 is the largest, using SIMPLEX algorithm, we will eliminate d_1 from d_m using Gaussian elimination based on $d_1 = d_2 - s_1$. We have the following updated linear equation systems:

$$\begin{aligned} d_m &= d_0 - \sum_{i=1}^{m-1} d_i, \\ d_1 &= d_2 - s_1, \\ s_2 &= d_3 - d_2, \\ &\vdots \\ s_{m-1} &= d_m - d_{m-1} = d_0 - md_{m-1} + \sum_{i=1}^{m-2} is_i, \\ t_m &= \Bigg((2^m-1)(n-1) - 2^{m-1}d_0 + \Big(2\cdot 2^{m-1} \\ &\quad - \sum_{i=1}^{2} 2^{i-1}\Big)d_2 + \sum_{i=3}^{m-1}(2^{m-1}-2^{i-1})d_i \\ &\quad - \sum_{i=1}^{m} 2^{i-1}\varepsilon_i - (2^m-1)s_1\Bigg)/2^m. \end{aligned} \tag{48}$$

Repeat from d_2 to d_{m-1}, we have

$$\begin{aligned} d_m &= d_0 - \sum_{i=1}^{m-1} d_i, \\ d_1 &= d_2 - s_1, \\ d_2 &= d_3 - s_2, \\ &\vdots \\ d_{m-2} &= d_{m-1} - s_{m-2}, \\ s_{m-1} &= d_m - d_{m-1} = d_0 - md_{m-1} + \sum_{i=1}^{m-2} is_i, \\ t_m &= \Bigg((2^m-1)(n-1) - 2^{m-1}d_0 \\ &\quad - \Big((m-2)\cdot 2^{m-1}+1\Big)d_{m-1} + - \sum_{i=1}^{m} 2^{i-1}\varepsilon_i \\ &\quad - \sum_{i=1}^{m-2}\Big(i\cdot 2^{m-1} - 2^{m-2}+1\Big)s_i\Bigg)/2^m. \end{aligned} \tag{49}$$

Finally, from d_{m-1} to s_{m-1}. We have

$$\begin{aligned} d_m &= d_0 - \sum_{i=1}^{m-1} d_i, \\ d_1 &= d_2 - s_1, \\ &\vdots \\ d_{m-2} &= d_{m-1} - s_{m-2}, \\ d_{m-1} &= \left(d_0 + \sum_{i=1}^{m-2} is_i - s_{m-1}\right)/m, \\ t_m &= \Bigg((2^m-1)(n-1) - \Big(2^{m-1} + \frac{1}{m}\big((m-2)2^{m-1} \\ &\quad +1\big)\Big)d_0 - \sum_{i=1}^{m} 2^{i-1}\varepsilon_i - \sum_{i=1}^{m-2}\Big(2^{m-1} \\ &\quad - \frac{1}{m}\big((2^m-1)+2^{i-1}\big)i - 2^{m-2}+1\Big)s_i \\ &\quad - \frac{1}{m}\big((m-2)2^{m-1}+1\big)s_{m-1}\Bigg)/2^m. \end{aligned} \tag{50}$$

From equation (50) we can see that the SIMPLEX algorithm should stop. The optimal solution of t_m can be achieved when

$$d_i = \text{round}(d_0/m) = \widetilde{d} \quad \text{for} \quad 1 \le i \le m.$$

Moreover, we have

$$\begin{aligned}\widetilde{t}_m &= \left((2^m-1)(n-\widetilde{d}-1) - \sum_{i=1}^{m} 2^{i-1}\varepsilon_i\right)/2^m \\ &\ge \left((2^m-1)(n-\widetilde{d}-2)\right)/2^m.\end{aligned}$$

The worst case is achieved when $\varepsilon_i = 1$ for $i = 1, \dots, m$. □

It is easy to derive the following corollary.

Corollary 1: The optimal error correction capability of the m-layer rate-matched MSR code increases with the number of layers m.

Proof: From equation (46), the error correction capability can be further written as:

$$\left(1-\frac{1}{2^m}\right)\left(n-\text{round}\left(\frac{d_0}{m}\right)-2\right)\le \widetilde{t}_m \le \left(1-\frac{1}{2^m}\right)\left(n-\text{round}\left(\frac{d_0}{m}\right)-1\right), \tag{51}$$

where both sides increase with m, and the difference between the two sides is at most 1. Based on this observation, it is easy to prove that $\widetilde{t}_m$ also increases with m. □

Remark 2: Although the m-layer rate-matched MSR code shares the same principle with the 2-layer code, it is not a direct extension of the 2-layer code for three reasons: First, the application scenario and optimization goals for the m-layer rate-matched MSR code are different from the 2-layer rate-matched MSR code. The 2-layer code is designed to optimize the storage efficiency under the constraint of any predetermined error correction capability, while the m-layer code is designed to optimize the overall error correction capability under the constraint of any giving code efficiency. Second, under the same comparable optimization constrains, the error correction capability of the 2-layer code is much worse than the m-layer rate-matched MSR code. Third, the m-layer rate-matched MSR code is more secure under malicious attacks than the direct generalization from the 2-layer code due to the more diversified structure.

4) Optimal Code Rate - Dual of Optimal Error Correction: During the optimization, we set the code rate of the rate-matched MSR code to a constant value and maximize the error correction capability. To optimize the rate-matched MSR code, we can also set the error correction capability t_i for $i=m$ in equation (42) to a constant value

$$t_m=\left(\sum_{i=1}^{m}2^{i-1}(n-d_i)-\sum_{i=1}^{m}2^{i-1}\varepsilon_i-2^m+1\right)/2^m=t_0, \tag{52}$$

and maximize the code rate during regeneration. The problem can be written as:

$$\begin{aligned}&\text{maximize} \quad \sum_{i=1}^{m}d_i,\\ &\text{subject to}\quad \text{equation (43)}: -\sum_{j=1}^{i-1}2^{j-1}d_j+2^{i-1}d_i\le n\\ &\qquad\qquad +\sum_{j=1}^{i-1}2^{j-1}\varepsilon_j-1,\quad 2\le i\le m,\\ &\qquad\qquad \text{equation (41)}: d_{i-1}-d_i\le 0,\quad 2\le i\le m,\\ &\qquad\qquad \text{equation (52)}: t_m=t_0.\end{aligned} \tag{53}$$

The optimization result is the same as that of equation (44). That is when all the $d_i's$ for $1\le i\le m$ are the same, the code rate is maximized. More specifically, we have:

$$\begin{aligned}d_i&=n-1-\left(2^m t_0+\sum_{i=1}^{m}2^{i-1}\varepsilon_i\right)\Big/(2^m-1)\\ &\ge n-2-\frac{2^m t_0}{2^m-1}\ \text{(worst case)}.\end{aligned} \tag{54}$$

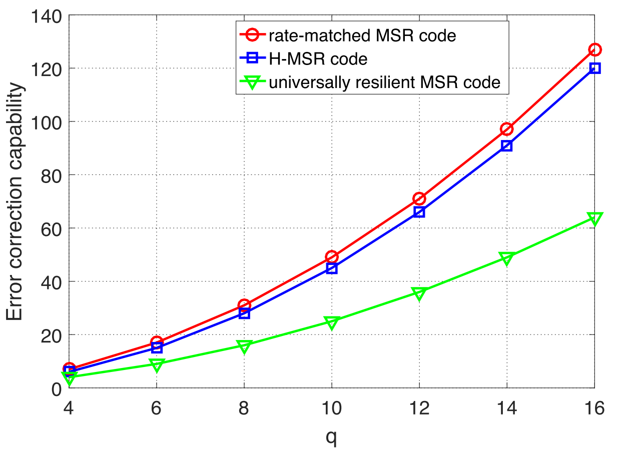

Fig. 7. Comparison of error correction capability between the m-layer rate matched MSR code and the H-MSR code

B. Evaluation of the Optimization

1) Comparison with the Hermitian Code Based MSR Code in [3]: The Hermitian code based MSR code (H-MSR code) in [3] has better error correction capability than the universally resilient MSR code. However, because the structure of the underlying Hermitian code is predetermined, the error correction capability may not be optimal. Fig. 7 shows the maximum number of corrupted nodes from which the errors can be corrected by the H-MSR code. Here we set the parameter q of the Hermitian code [41] from 4 to 16 with a step of 2. In the figure, we also plot the performance of the m-layer rate-matched MSR code with the same code rates as the H-MSR code. The comparative result demonstrates that the rate-matched MSR code has a better error correction capability than the H-MSR code. Moreover, the rate-matched code is easier to understand and has more flexibility than the H-MSR code. Even if there is an adversary that corrupts everything in the storage nodes, which neutralizes the gain of the correlated layered decoding, the performance of the rate-matched MSR code will be at least the same with the H-MSR code and the universally resilient MSR code.

2) Number of Layers and Error Correction Capability: Since we have seen the advantages of the rate-matched MSR code over the universally resilient MSR code in Section VI-A.2, here we will mainly discuss how the number of layers can affect the error correction capability. The error correction capability of the m-layer rate-matched MSR code is shown is Fig. 8, where we set $n=30$ and $d_0=50$. We also plot the error correction capability of the universally resilient MSR code with the same code rates for comparison. From the figure we can see that when n and d_0 are fixed, the optimal error correction capability will increase with the number of layers m as we have proved in Corollary 1.

3) Optimal Storage Capacity: The optimal condition in equation (45) also leads to maximum storage capacity besides the optimal error correction capability. We have the following theorem:

Theorem 6: The m-layer rate-matched MSR code can achieve the maximum storage capacity if the parameter d_i's

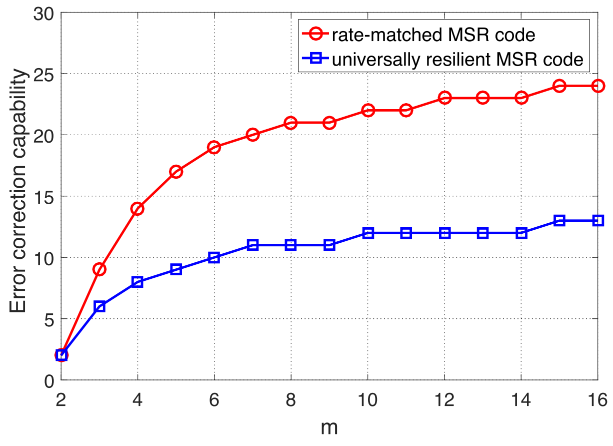

Fig. 8. The optimal error correction capability of the m-layer rate-matched MSR code under different m for $2 \le m \le 16$

of all the layers are the same, under the constraint in equation (27).

Proof: The code of the i^{th} layer can store one block of data with the size $B_i = \alpha_i(\alpha_i + 1) = (d_i/2)(d_i/2 + 1)$. So the m-layer code can store data with the size $B = \sum_{i=1}^{m}(d_i/2)(d_i/2 + 1)$. Our goal here is to maximize B under the constraint in equation (27).

We can use Lagrange multipliers to find the point of maximum B. Let

$$\Lambda_L(d_1, \ldots, d_m, \lambda) = \sum_{i=1}^{m}(d_i/2)(d_i/2 + 1) + \lambda\left(\sum_{i=1}^{m} d_i - d_0\right). \tag{55}$$

We can find the maximum value of B by setting the partial derivatives of this equation to zero:

$$\frac{\partial \Lambda_L}{\partial d_i} = \frac{d_i + 1}{2} + \lambda = 0, \quad \forall 1 \le i \le m. \tag{56}$$

Here we can see that when all the parameter d's of all the layers are the same, we can get the maximum storage capacity B. This maximization condition coincides with the optimal condition in achieving the goal of this section: optimize the overall error correction capability of the rate-matched MSR code. □

C. Practical Consideration of the Optimization

So far, we implicitly presume that there is only one data block of the size $B_i = \alpha_i(\alpha_i + 1)$ for each layer i. In practical distributed storage, it is the parameter d_i that is fixed instead of d_0, the summation of d_i. However, as long as we use m layers of MSR codes with the same parameter $d = \widetilde{d}$, we will still get the optimal solution for $d_0 = m\widetilde{d}$. In fact, the m-layer rate-matched MSR code here becomes a single full rate MSR code with parameter $d = \widetilde{d}$ and m data blocks. And based on the dependent decoding idea we describe at the beginning of Section VI, we can achieve the optimal performance.

So when the file size B is larger than one data block size $\widetilde{B}$ of the single full rate MSR code with parameter $d = \widetilde{d}$,

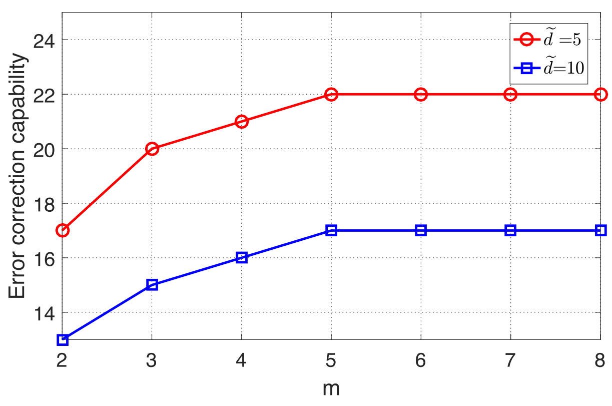

Fig. 9. The optimal error correction capability for $2 \le m \le 8$

we will divide the file into $\lceil B/\widetilde{B}\rceil$ data blocks and encode them separately. If we decode these data blocks dependently, we can get the optimal error correction capability.

1) Evaluation of the Optimal Error Correction Capability: In practical case, $\widetilde{d}$ could be fixed. So here we will study the relationship between the number of dependently decoding data blocks m and the error correction capability, which is shown in Fig. 9. In the figure, we set $n = 30$ and $\widetilde{d} = 5, 10$, respectively. From the figure we can see that although the error correction capability will become higher with the increasing of dependently decoding data blocks m, the amount of improvement will be negligible for $m \ge 5$. Actually when $m = 5$ the capability has already achieved the upper bound.

On the other hand, there exist parallel algorithms for fast MDS code decoding [42]. We can decode blocks of MDS codewords in parallel a pipeline fashion to accelerate the overall decoding speed. The more blocks of codewords we decode in parallel, the faster we will finish the whole decoding process. For large files that could be divided into a large amount of data blocks (θ blocks), we can get a trade-off between the optimal error correction capability and the decoding speed by setting the number of dependent decoding data blocks m and the number of parallel decoding data blocks ρ under the constraint $\theta = m\rho$.

D. Encoding

From the analysis above we know that to encode a file with size B using the optimal m-layer rate-matched MSR code is to encode the file using a full rate MSR code with predetermined parameter $d = 2\alpha = \widetilde{d}$. First the file will be divided into θ blocks of data with size $\widetilde{B}$, where $\theta = \lceil B/\widetilde{B}\rceil$. Then the θ blocks of data will be encoded into codeword matrices $\mathsf{F}_1, \ldots, \mathsf{F}_\theta$ and form the final $n \times \alpha\theta$ codeword matrix: $\mathcal{C} = [\mathsf{F}_1, \ldots, \mathsf{F}_\theta]$. Each row $\mathbf{c}_i = [\mathsf{f}_{1,i}, \ldots, \mathsf{f}_{\theta,i}]$, $1 \le i \le n$, of the codeword matrix $\mathcal{C}$ will be stored in storage node i, where $\mathsf{f}_{j,i}$ is the i^{th} row of F_j, $1 \le j \le \theta$. The encoding vector $\boldsymbol{\psi}_i$ for storage node i is the i^{th} row of Ψ in equation (5).

Theorem 7: The encoding of m-layer rate-matched MSR code can achieve the MSR point in equation (2) since each layer of the code is an MSR code.

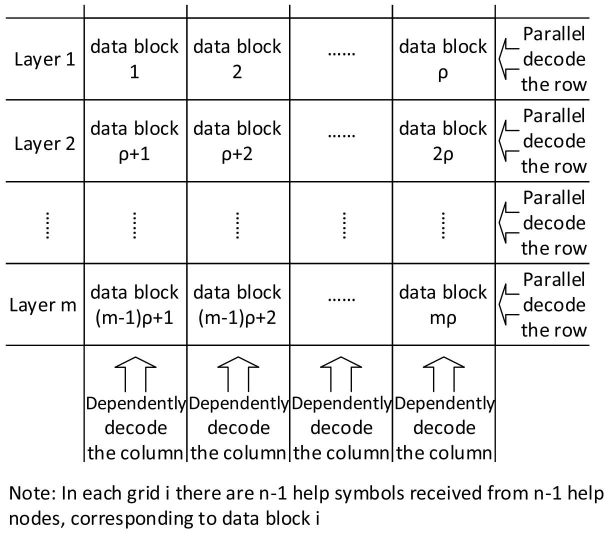

Fig. 10. Lattice of received help symbols for regeneration

E. Regeneration

Suppose node z fails, the replacement node z' will send regeneration requests to the rest of $n-1$ helper nodes. Upon receiving the regeneration request, helper node i will calculate and send out the help symbols $\mathsf{f}_{1,i}\boldsymbol{\phi}_z^T, \dots, \mathsf{f}_{\theta,i}\boldsymbol{\phi}_z^T$.

As we have discussed before, combining both dependent decoding and parallel decoding can achieve the trade-off between optimal error correction capability and decoding speed. Although all θ blocks of data are encoded with the same MSR code, z' will place the received help symbols into a 2-dimension lattice with size $m \times \rho$ as shown in Fig. 10. In each grid of the lattice there are $n-1$ help symbols corresponding to one data block, received from $n-1$ helper nodes. We can view each row of the lattice as related to a layer of an m-layer rate-matched MSR code with ρ blocks of data, which will be decoded in parallel. We also view each column of the lattice as related to m layers of an m-layer rate-matched MSR code with one block of data each layer, which will be decoded dependently. z' will perform Algorithm 4 to regenerate the contents of the failed node z.

Arrange the received help symbols according to Fig. 10. Repeat the following steps from Layer 1 to Layer m:

Algorithm 4: z' regenerates symbols of the failed node z for the m-layer rate-matched MSR code.

Step 1: For a grid, if errors are detected in the symbols sent by node i in previous layers of the same column, replace the symbol sent from node i by an erasure $\otimes$.

Step 2: Regenerate all the symbols related to ρ data blocks in parallel using the algorithm similar to Algorithm 1 with only one difference: Decode in parallel all the ρ MDS codes in **Step 1** of Algorithm 1.

The error correction capability of the regeneration is described in Theorem 5.

F. Reconstruction

When DC needs to reconstruct the original file, it will send reconstruction requests to n storage nodes. Upon receiving the request, node i will send out the symbol vector $\mathbf{c}_i$. Suppose $\mathbf{c}'_i = \mathbf{c}_i + \mathbf{e}_i$ is the response from the i^{th} storage node. If $\mathbf{e}_i \in \mathbb{F}_q^{\alpha\theta}\backslash\{\mathbf{0}\}$, then node i is corrupted since $\mathbf{c}_i$ has been modified. The strategy of combining dependent decoding and parallel decoding for reconstruction is similar to that of regeneration. DC will place the received symbols into a 2-dimension lattice with size $m \times \rho$. The only difference is that in a grid of the lattice, there are n symbol vectors $\mathsf{f}'_{j,1}, \dots, \mathsf{f}'_{j,n}$ corresponding to data block j, received from n storage nodes. DC will perform reconstruction using Algorithm 5.

Arrange the received symbols similar to Fig. 10. Here we place the received codeword matrix F'_j into grid j instead of help symbols received from $n-1$ help nodes. Repeat the following steps from Layer 1 to Layer m:

Algorithm 5: DC reconstructs the original file for the m-layer rate-matched MSR code.

Step 1: For a grid, if errors are detected in the symbols sent by node i in previous layers of the same column, replace symbols sent from node i by erasures $\otimes$.

Step 2: Regenerate all the symbols of the ρ data blocks in parallel using the algorithm similar to Section IV-A.3 with only one difference: Decode in parallel all the MDS codes in Section IV-A.3.

For data reconstruction, we have the following theorem:

Theorem 8 (Optimal Parameters): For reconstruction of the m-layer rate-matched MSR code, when

$$d_i = \text{round}(d_0/m) = \widetilde{d} \text{ for } 1 \le i \le m,$$

the number of corrupted nodes from which the errors can be corrected is maximized.

Proof: From Section VI-A we know that for regeneration of an optimal m-layer rate-matched MSR code, the parameter d's of all the layers are the same, which implies the parameter α's of all layers are also the same. Since optimization of regeneration is derived based on the decoding of $(n-1, d, n-d)$ MDS codes and in reconstruction we have to decode $(n-1, \alpha, n-\alpha)$ MDS codes, if the parameter α's of all the layers are the same, we can achieve the same optimization results for reconstruction. □

VII. Conclusion

In this paper, we develop two rate-matched regenerating codes for corrupted nodes detection and correction in hostile networks: 2-layer rate-matched regenerating code and m-layer rate-matched regenerating code. We propose encoding, regeneration and reconstruction algorithms for both codes. For the 2-layer rate-matched code, we develop the optimal parameter for data regeneration, considering the trade-off between the corrupted nodes detection probability and the storage efficiency. Theoretical analysis shows that the code can successfully detect and correct corrupted nodes using the optimized parameters. Our analysis also shows that the code has higher storage efficiency compared to the universally resilient regenerating code (70% higher for the detection probability 0.999999). Then based on the same motivation of "previous errors could be viewed as current erasures", we propose the m-layer code and develop parameters to optimize the overall error correction capability by matching the code rate of each layer's regenerating code. Theoretical analysis shows that the optimized parameter could also achieve the maximum storage capacity under the same constraint, which is consistent with the optimization goal of the 2-layer code. Furthermore, analysis shows that compared to the universally resilient regenerating code, our m-layer code can improve the error correction capability more than 50%.

Acknowledge

The authors would like to thank the anonymous referees for their valuable comments, which greatly improved the quality of the presentation.

References

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

[2] K. V. Rashmi, N. B. Shah, K. Ramchandran, and P. V. Kumar, "Regenerating codes for errors and erasures in distributed storage," in *Proc. Int. Symp. Inf. Theory (ISIT)*, 2012, pp. 1202–1206.

[3] J. Li, T. Li, and J. Ren, "Beyond the MDS bound in distributed cloud storage," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 307–315.

[4] S. Rhea *et al.*, "Maintenance-free global data storage," *IEEE Internet Comput.*, vol. 5, no. 5, pp. 40–49, Sep. 2001.

[5] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total recall: System support for automated availability management," in *Proc. Symp. Netw. Syst. Design Implement.*, 2004, pp. 337–350.

[6] D. Cullina, A. G. Dimakis, and T. Ho. (2009). "Searching for minimum storage regenerating codes." [Online]. Available: https://arxiv.org/abs/0910.2245

[7] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Jan. 2010, pp. 1–5.

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

[9] Y. Wu, "A construction of systematic MDS codes with minimum repair bandwidth," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3738–3741, Jun. 2011.

[10] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: Network coding for cloud storage," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2801–2805.

[11] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. 48th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2010, pp. 1510–1517.

[12] I. Tamo, Z. Wang, and J. Bruck, "MDS array codes with optimal rebuilding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul./Aug. 2011, pp. 1240–1244.

[13] V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li. (2011). "Optimal repair of MDS codes in distributed storage via subspace interference alignment." [Online]. Available: https://arxiv.org/abs/1106.1250

[14] D. Papailiopoulos, A. G. Dimakis, and V. Cadambe, "Repair optimal erasure codes through Hadamard designs," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3021–3037, May 2013.

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

[16] K. W. Shum and Y. Hu, "Existence of minimum-repair-bandwidth cooperative regenerating codes," in *Proc. Int. Symp. Netw. Coding (NetCod)*, 2011, pp. 1–6.

[17] A. Wang and Z. Zhang, "Exact cooperative regenerating codes with minimum-repair-bandwidth for distributed storage," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 400–404.

[18] C. Tian and T. Liu, "Multilevel diversity coding with regeneration," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 4833–4847, Sep. 2016.

[19] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC regenerating code: Binary addition and shift for exact repair," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1621–1625.

[20] Y.-L. Chen, G.-M. Li, C.-T. Tsai, S.-M. Yuan, and H.-T. Chiao, "Regenerating code based P2P storage scheme with caching," in *Proc. 4th Int. Conf. Comput. Sci. Converg. Inf. Technol. (ICCIT)*, 2009, pp. 927–932.

[21] Y. Wu, A. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. 45th Annu. Allerton Conf. Control, Comput., Commun.*, 2007, pp. 1–8.

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

[23] K. W. Shum, "Cooperative regenerating codes for distributed storage systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.

[24] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC codes: Low-complexity regenerating codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3053–3069, Jun. 2016.

[25] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun./Jul. 2009, pp. 2276–2280.

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

[27] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[28] V. Guruswami and M. Wootters. (2016). "Repairing Reed–Solomon codes." [Online]. Available: https://arxiv.org/abs/1509.04764

[29] C. Tian, B. Sasidharan, V. Aggarwal, V. A. Vaishampayan, and P. V. Kumar, "Layered exact-repair regenerating codes via embedded error correction and block designs," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1933–1947, Apr. 2015.

[30] F. Oggier and A. Datta, "Byzantine fault tolerance of regenerating codes," in *Proc. IEEE Int. Conf. Peer-Peer Comput. (P2P)*, Aug./Sep. 2011, pp. 112–121.

[31] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6734–6753, Oct. 2011.

[32] Y. S. Han, R. Zheng, and W. H. Mow, "Exact regenerating codes for Byzantine fault tolerance in distributed storage," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2498–2506.

[33] H. C. H. Chen and P. P. C. Lee, "Enabling data integrity protection in regenerating-coding-based cloud storage," in *Proc. IEEE 31st Symp. Rel. Distrib. Syst. (SRDS)*, Oct. 2012, pp. 51–60.

[34] C. Cachin and S. Tessaro, "Optimal resilience for erasure-coded Byzantine distributed storage," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, 2006, pp. 115–124.

[35] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Lazy verification in fault-tolerant distributed storage systems," in *Proc. 24th IEEE Symp. Rel. Distrib. Syst. (SRDS)*, Oct. 2005, pp. 179–190.

[36] N. B. Shah, K. V. Rashmi, K. Ramchandran, and P. V. Kumar. *Privacy-Preserving and Secure Distributed Storage Codes*, accessed on (2013). [Online]. Available: https://www.eecs.berkeley.edu/~nihar/publications/privacy_security.pdf/

[37] J. Li, T. Li, and J. Ren, "Secure regenerating code," in *Proc. IEEE GLOBECOM*, Dec. 2014, pp. 770–774.

[38] R. Tandon, S. Amuru, T. C. Clancy, and R. M. Buehrer, "Toward optimal secure distributed storage systems with exact repair," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3477–3492, Jun. 2016.

[39] S. Lin and D. J. Costello, Jr., *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[41] J. Ren, "On the structure of Hermitian codes and decoding for burst errors," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2850–2854, Nov. 2004.

[42] D. Dabiri and I. F. Blake, "Fast parallel algorithms for decoding Reed–Solomon codes based on remainder polynomials," *IEEE Trans. Inf. Theory*, vol. 41, no. 4, pp. 873–885, Jul. 1995.

Jian Li received the BS and MS degrees both in electrical engineering from Tsinghua University, China in 2005 and 2008 respectively, and the PhD degree in electrical engineering from Michigan State University, East Lansing in 2015. He is an assistant professor in the School of Electronic and Information Engineering, Beijing Jiaotong University, China. His current research interests include network security, cloud storage, wireless sensor network in Internet of things, privacy-preserving communications, and cognitive networks.

Jian Ren (SM'09) received the BS and MS degrees both in mathematics from Shaanxi Normal University, and received the Ph.D. degree in EE from Xidian University, China. He is an Associate Professor in the Department of ECE at Michigan State University. His current research interests include network security, cloud computing security, privacy-preserving communications, distributed network storage, and Internet of Things. He is a recipient of the US National Science Foundation Faculty Early Career Development (CAREER) award in 2009. Dr. Ren is the TPC Chair of IEEE ICNC'17 and General Chair of ICNC'18. Dr. Ren is a senior member of the IEEE.

Tongtong Li (SM'08) Tongtong Li received her Ph.D. degree in Electrical Engineering in 2000 from Auburn University. She is currently an Associate Professor in the Depart- ment of Electrical and Computer Engineering at Michigan State University. Her research interests fall into the areas of communication system design and networking, wireless security, and statistical signal processing, with applications in computational neu- roscience. Dr. Li is currently serving as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING. She is a recipient of the National Science Foundation (NSF) CAREER Award and a senior member of the IEEE.