Rate-matched Regenerating Code in Hostile Networks

Jian Li Tongtong Li Jian Ren
Department of ECE, Michigan State University, East Lansing, MI 48824-1226.
Email: {lijian6, tongli, renjian}@msu.edu

***Abstract*—Regenerating code is a class of code very suitable for distributed storage systems, which can maintain optimal bandwidth and storage space. Two types of important regenerating code have been constructed: the minimum storage regeneration (MSR) code and the minimum bandwidth regeneration (MBR) code. However, in hostile networks where adversaries can compromise storage nodes, the storage capacity of the network can be significantly affected. In this paper, we propose a rate-matched MSR code that can combat against this kind of adversaries in hostile networks. We optimize the code parameters for given system requirements. Our comprehensive analysis shows that our code can detect and correct malicious nodes with higher storage efficiency compared to the normal error correction MSR code.**

***Index terms*—regenerating code, MDS code, error-correction, adversary.**

I. Introduction

Distributed storage is a popular method to store files securely without requiring data encryption. Instead of storing a file and its replications in multiple servers, we can break the file into components and store the components into multiple servers. In this way, both the reliability and the security of the file can be increased. 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 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]. The bandwidth needed for repairing a failed node could be much less than the size of the whole file. In fact the regenerating code achieves an optimal trade-off between bandwidth and storage within the minimum storage regeneration (MSR) and the minimum bandwidth regeneration (MBR) points.

The construction of the regenerating code has been well studied. In [2]–[4], the authors studied the functional regenerating code, where the replacement node regenerates a new component that can functionally replace a failed component instead of being the same as the original stored component. The exact regeneration was studied in [5], where the replacement node regenerates the exact symbols of a failed node. [6] presented optimal exact constructions of MBR code and MSR code under product-matrix framework.

This work was supported in part by the NSF under grants CNS-0845812, CNS-1117831, CNS-1217206, and ECCS-1232109.

In recent year, the security of the regenerating code is being studied. In [7], the authors proposed to add CRC codes in the regenerating code to check the integrity of the data in hostile networks. Unfortunately, the CRC checks can be easily manipulated by the malicious nodes. In [8], the authors analyzed the error resilience of the regenerating code in the network with errors and erasures. They provided the theoretical error correction capability. In [9] the authors proposed a Hermitian code based regenerating code, which could provide better error correction capability. In [10] the authors proposed the universally secure regenerating code to achieve information theoretic data confidentiality. But the extra computational cost and bandwidth have to be considered for this code. In [11] the authors proposed to apply linear feedback shift register (LFSR) to protect the data confidentiality.

In this paper, we focus on error correction and malicious node locating in the data regeneration and reconstruction in distributed storage. We propose a rate-matched MSR code for hostile networks. This code can achieve storage efficiency higher than the code proposed in [8]. When no error occurs and no malicious node exists, the data regeneration and reconstruction can be processed the same as the existing works.

The rest of this paper is organized as follows: in Section II, the preliminary of this paper is presented. In Section III, we propose the rate-matched MSR code schemes. We conduct security and performance analyses in Section IV. The paper is concluded in Section V.

II. Preliminary and Assumptions

A. Regenerating Code

Regenerating code introduced in [1] is a linear code over finite field GF_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. γ and α

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 in charge of encoding and distributing the data to storage nodes. Replacement nodes will also be initialized by the secure server. DC and the secure server can be implemented in the same computer and can never be compromised.

We also assume that when a normal storage node is compromised by an adversary, it will become a malicious node. After compromising a storage node, the adversary can get full control of the node and can modify the symbols sent to the replacement nodes and DC. Corrupted node and malicious node will be used interchangeably in this paper without making any distinction.

III. Rate-matched MSR Code Design

In this section, we will analyze 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 the same way through truncating operations. In the rate-matched MSR code, there are two types of MSR codes with different code rates: low rate code and high rate code. The purpose of the low rate code is to correct the erroneous symbols sent by malicious nodes and locate the corresponding malicious nodes. Then we can treat the errors in the received symbols as erasures when regenerating with the high rate code. However, the rates of the two codes must match to achieve an optimal performance. In this paper, we mainly focus on the rate match for data regeneration. We can see in the later analysis that the performance of data reconstruction can also be improved with this design criterion. For better illustration, we will first introduce the high rate code and the low rate code, then the rate-matched MSR code.

A. High Rate Code

1) Encoding: The high rate code is encoded based on the product-matrix code framework in [6]. According to equation (2), we have $\alpha_H = d/2$, $\beta_H = 1$ for one block of data with the size $B_H = (\alpha+1)\alpha$. The data will be arranged into two $\alpha\times\alpha$ symmetric matrices S_1, S_2, each of which will contain $B_H/2$ data. The codeword CH is defined as

$$CH = [\Phi \;\; \Lambda\Phi]\begin{bmatrix}S_1\\ S_2\end{bmatrix} = \Psi S_H, \tag{4}$$

where

$$\Phi = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \phi & \phi^2 & \dots & \phi^{\alpha-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \phi^{n-1} & (\phi^{n-1})^2 & \dots & (\phi^{n-1})^{\alpha-1} \end{bmatrix} \tag{5}$$

is a Vandermonde matrix and $\Lambda = \mathrm{diag}[\lambda_1,\lambda_2,\cdots,\lambda_\alpha]$ such that $\lambda_i \neq \lambda_j$ for $1\le i,j\le\alpha, i\neq j$, where $\lambda_i \in GF_q$ for $1\le i\le\alpha$, ϕ is a primitive element in GF_q, and any d rows of Ψ are linear independent. Then each row $\mathbf{ch}_i$, $0\le i<n$, of the codeword matrix CH will be stored in storage node i, in which the encoding vector ν_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 $n-1$ helper nodes. Upon receiving the regeneration request, helper node i will calculate and send out the help symbol $p_i = \mathbf{ch}_i\mu_z^T$, where μ_z is the z^{th} row of Φ. z' will perform Algorithm 1 to regenerate the contents of the failed node z. For convenience, we define $\mathbf{V}_{i,j} = \left[\nu_i^T,\nu_{i+1}^T\cdots,\nu_j^T\right]^T$, where ν_t, $i\le t\le j$, is the t^{th} row of Ψ and $\mathbf{x}^{(j)}$ is the vector containing the first j symbols of $S_H\mu_z^T$.

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

Suppose $p_i' = p_i + e_i$ is the response from the i^{th} helper node. If p_i has been modified by the malicious node i, we have $e_i \in GF_q\backslash\{0\}$. We can successfully regenerate the symbols in node z when the errors in the received help symbols p_i' from $n-1$ helper nodes can be corrected. Without loss of generality, we assume $0\le i\le n-2$.

Step 1: Decode $\mathbf{p}'$ to $\mathbf{p}_{cw}$, where $\mathbf{p}' = [p_0',p_1',\cdots,p_{n-2}']^T$ can be viewed as an MDS code with parameters $(n-1,d,n-d)$ since $\mathbf{V}_{0,n-2}\cdot\mathbf{x}^{(n-1)} = \mathbf{p}'$. If the i^{th} positions of $\mathbf{p}_{cw}$ and $\mathbf{p}'$ are different, mark node i as corrupted.

Step 2: Solve $\mathbf{V}_{0,n-2}\cdot\mathbf{x}^{(n-1)} = \mathbf{p}_{cw}$ and compute $\mathbf{ch}_z = \mu_z S_1 + \lambda_z\mu_z S_2$ as described in [6].

B. Low Rate Code

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

$$B_L = \begin{cases} xd(1+xd)/2, x\in(0,0.5] \\ \alpha(\alpha+1)/2 + (x-0.5)d(1+(x-0.5)d), x\in(0.5,1) \end{cases}, \tag{6}$$

where x is the match factor of the rate-matched MSR code and is determined by the number of storage nodes n and the number of maximum malicious nodes M. The data $\mathbf{m} = [m_1,m_2,\dots,m_{B_L}] \in GF_q^{B_L}$ will be processed as follows:

When $x\le 0.5$, the data will be arranged into a symmetric matrix S_1 of the size $\alpha\times\alpha$:

$$S_1 = \begin{bmatrix} m_1 & m_2 & \dots & m_{xd} & 0 & \dots & 0 \\ m_2 & m_{xd+1} & \dots & m_{2xd-1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ m_{xd} & m_{2xd-1} & \dots & m_{B_L/2} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{7}$$

The codeword CL is defined as

$$CL = [\Phi \ \Lambda\Phi] \begin{bmatrix} S_1 \\ \mathbf{0} \end{bmatrix} = \Psi S_L, \tag{8}$$

where $\mathbf{0}$ is the $\alpha \times \alpha$ zero matrix and Φ, Λ, Ψ are the same as the high rate code.

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 $m_{\alpha(\alpha+1)/2+1}, \cdots, m_{B_L}$ will be arranged into another $\alpha \times \alpha$ symmetric matrix S_2:

$$S_2 = \begin{bmatrix} m_{\alpha(\alpha+1)/2+1} & \cdots & m_{\alpha(\alpha+1)/2+xd} & 0 \ldots 0 \\ m_{\alpha(\alpha+1)/2+2} & \cdots & m_{\alpha(\alpha+1)/2+2xd-1} & 0 \ldots 0 \\ \vdots & \ddots & \vdots & \vdots \ddots \vdots \\ m_{\alpha(\alpha+1)/2+xd} & \cdots & m_{B_L/2} & 0 \ldots 0 \\ 0 & \cdots & 0 & 0 \ldots 0 \\ \vdots & \ddots & \vdots & \vdots \ddots \vdots \\ 0 & \cdots & 0 & 0 \ldots 0 \end{bmatrix}. \tag{9}$$

The codeword CL is defined the same as equation (4) with the same parameters Φ, Λ and Ψ.

Then each row $\mathbf{cl}_i$, $0 \le i < n$, of the codeword matrix CL will be stored in storage node i respectively, in which the encoding vector ν_i is the i^{th} row of Ψ.

2) Regeneration: The regeneration for the low rate code is the same as the regeneration for the high rate code described in section III-A2 with only a minor difference. If we define $\mathbf{x}^{(j)}$ as the vector containing the first j symbols of $S_L \mu_z^T$, there will be only xd nonzero elements in the vector. According to $\mathbf{V}_{0,n-2} \cdot \mathbf{x}^{(n-1)} = \mathbf{p}'$, the received symbol vector $\mathbf{p}'$ for the low 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 low rate code than using the high rate code.

C. Rate-matched MSR Code

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

$$\lfloor (n-xd-1)/2 \rfloor = n-d-1, \tag{10}$$

from which we can derive the match factor x.

1) Encoding: To encode a file with size B_F using the rate-matched MSR code, the file will first be divided into θ_H blocks of data with the size B_H and θ_L blocks of data with the size B_L, where the parameters should satisfy

$$B_F = \theta_H B_H + \theta_L B_L. \tag{11}$$

Then the θ_H blocks of data will be encoded into code matrices $CH_1, \ldots, CH_{\theta_H}$ using the high rate code and the θ_L blocks of data will be encoded into code matrices $CL_1, \ldots, CL_{\theta_L}$ using the low rate code. To prevent the malicious nodes from corrupting the high rate code only, the secure server will randomly concatenate all the matrices together to form the final $n \times \alpha(\theta_H + \theta_L)$ codeword matrix:

$$CM = [\text{Perm}(CH_1, \ldots, CH_{\theta_H}, CL_1, \ldots, CL_{\theta_L})], \tag{12}$$

where Perm$(\cdot)$ is the random matrices permutation operation. The secure sever will also record the order of the permutation for future code regeneration and reconstruction. Then each row $\mathbf{c}_i = [\text{Perm}(\mathbf{ch_1}_{,i}, \ldots, \mathbf{ch_{\theta_H}}_{,i}, \mathbf{cl_1}_{,i}, \ldots, \mathbf{cl_{\theta_L}}_{,i})]$, $0 \le i \le n-1$, of the codeword matrix CM will be stored in storage node i, where $\mathbf{ch_j}_{,i}$ is the i^{th} row of CH_j, $1 \le j \le \theta_H$, and $\mathbf{cl_j}_{,i}$ is the i^{th} row of CL_j, $1 \le j \le \theta_L$. The encoding vector ν_i for storage node i is the i^{th} row of Ψ in equation (4).

2) Regeneration: Suppose node z fails, the security server will initialize a replacement node z' with the order information of the low rate code and the high rate code in the rate-matched MSR code. Then 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 symbol $p_i = \mathbf{c}_i \mu_z^T$. z' will perform Algorithm 2 to regenerate the contents of the failed node z. After the regeneration is finished, z' will erase the order information. So even if z' was compromised later, the adversary would not get the permutation order of the low rate code and the high rate code.

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

Step 1: According to the order information, regenerate all the symbols related to the θ_L data blocks encoded by the low rate code, using Algorithm 1. If errors are detected in the symbols sent by node i, it will be marked as malicious node.

Step 2: Regenerate all the symbols related to the θ_H data blocks encoded by the high rate code, using Algorithm 1. During the regeneration, all the symbols sent from nodes marked as malicious nodes will be replaced by erasures $\otimes$.

It is easy to see that Algorithm 2 can correct errors and locate malicious node using the low rate code while achieve high storage efficiency using the high rate code.

3) Parameters Optimization: We have the following design requirements for a given distributed storage system applying the rate-matched MSR code:

- The maximum number of malicious nodes M that the system can detect and locate using the low rate code. We have
$$\lfloor (n-xd-1)/2 \rfloor = M. \tag{13}$$
- The probability P_{det} that the system can detect all the malicious nodes. The detection will be successful if each malicious node modifies at least one help symbol corresponding to the low rate code and sends it to the replacement node. Suppose the malicious nodes modify

each help symbol to be sent to the replacement node with probability P, we have

$$(1-(1-P)^{\theta_L})^M \geq P_{det}. \qquad (14)$$

So there is a trade-off between θ_L and θ_H: the number of data blocks encoded by the low rate code and the number of data blocks encoded by the high rate code. If we encode using too much high rate code, we may not meet the detection probability P_{det} requirement. If too much low rate code is used, the redundancy may be too high.

The storage efficiency is defined as the ratio between the actual size of data to be stored and the total storage space needed by the encoded data:

$$\delta = \frac{\theta_H B_H + \theta_L B_L}{(\theta_H+\theta_L)n\alpha} = \frac{B_F}{(\theta_H+\theta_L)n\alpha}. \qquad (15)$$

Thus we can calculate the optimized parameters x, d, θ_H, θ_L by maximizing equation (15) under the constraints defined by equations (10), (11), (13), (14).

d and x can be determined by equation (10) and (13):

$$d = n-M-1, \qquad (16)$$
$$x = (n-2M-1)/(n-M-1). \qquad (17)$$

Since B_F is constant, to maximize δ is equal to minimize $\theta_H + \theta_L$. So we can rewrite the optimization problem as follows:

$$\text{Minimize } \theta_H + \theta_L, \text{ subject to } (11) \text{ and } (14). \qquad (18)$$

This is a simple linear programming problem. Here we will show the optimization results directly:

$$\theta_L = log_{(1-P)}(1-P_{det}^{1/M}), \qquad (19)$$
$$\theta_H = (B_F - \theta_L B_L)/B_H. \qquad (20)$$

In this paper we assume that we are storing large files, which means $B_F > \theta_L B_L$. So we can always find an optimal solution for the rate-matched MSR code.

D. Reconstruction for the Rate-matched MSR Code

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 $\mathbf{c}_i$ has been modified by the malicious node i, we have $\mathbf{e}_i \in (GF_q)^{\alpha(\theta_L+\theta_H)} \backslash \{\mathbf{0}\}$. Since DC has the permutation information of the low rate code and the high rate code, similar to the regeneration of the rate-matched MSR code, DC will perform the reconstruction using Algorithm 3.

Algorithm 3. *z' reconstructs the original file for the rate-matched MSR code*

Step 1: According to the order information, reconstruct each of the θ_L data blocks encoded by the low rate code and locate the malicious nodes.

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

For better illustration, we will discuss the reconstruction for the high rate code first then the low rate code below.

1) Reconstruction for the High Rate Code: Here we will use the encoded data block CH_1 as an example. Let $R' = [\mathbf{ch'_{1}}_{,0}{}^T, \mathbf{ch'_{1}}_{,1}{}^T, \cdots, \mathbf{ch'_{1}}_{,n-1}{}^T]^T$, we have

$$\Psi_{DC}\begin{bmatrix} S'_1 \\ S'_2 \end{bmatrix} = [\Phi_{DC} \;\; \Lambda_{DC}\Phi_{DC}]\begin{bmatrix} S'_1 \\ S'_2 \end{bmatrix} = \mathbf{V}_{0,n-1}\begin{bmatrix} S'_1 \\ S'_2 \end{bmatrix} = R',$$
$$\Phi_{DC}S'_1\Phi_{DC}^T + \Delta_{DC}\Phi_{DC}S'_2\Phi_{DC}^T = R'\Phi_{DC}^T. \qquad (21)$$

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

$$C + \Delta_{DC}D = \widehat{R}'. \qquad (22)$$

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

$$\begin{cases} C_{i,j} + \lambda_i \cdot D_{i,j} = \widehat{R}'_{i,j} \\ C_{i,j} + \lambda_j \cdot D_{i,j} = \widehat{R}'_{j,i}. \end{cases} \qquad (23)$$

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 malicious 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 $S'_1\Phi_{DC}^T = \widehat{S}'_1$, we have $\Phi_{DC}\widehat{S}'_1 = C$. Here we can view each column of C as an $(n-1, \alpha, n-\alpha)$ MDS code because Φ_{DC} 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 malicious nodes. Eventually C can be recovered. So DC can reconstructs S_1 using the method similar to [6]. For S_2, the recovering process is similar.

2) Reconstruction for the Low Rate Code: Here we will use the encoded data block CL_1 as an example. Let $R' = [\mathbf{cl'_{1}}_{,0}{}^T, \mathbf{cl'_{1}}_{,1}{}^T, \cdots, \mathbf{cl'_{1}}_{,n-1}{}^T]^T$.

When the match factor $x > 0.5$, reconstruction for the low rate code is the same to that for the high rate code.

When $x \leq 0.5$, equation (21) can be written as:

$$\Phi_{DC}S'_1 = R'. \qquad (24)$$

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 $\Phi_{DC}S_1 = R_{cw}$. Meanwhile, if the i^{th} rows of R' and R_{cw} are different, we can mark node i as corrupted.

3) Optimized Parameters: In section III-C3, we optimized the parameters for the data regeneration, considering the trade-off between the successful malicious node detection probability and the storage efficiency. Here we will show that the same parameters can guarantee that the same constraints be satisfied for the data reconstruction.

- The maximum number of malicious nodes can be detected for the data reconstruction is no smaller that M: if $x > 0.5$, the number is $\lfloor (n-\alpha-1)/2 \rfloor$. We have $\lfloor (n-\alpha-1)/2 \rfloor \geq \lfloor (n-xd-1)/2 \rfloor = M$. If $x \leq 0.5$, the number is $\lfloor (n-xd)/2 \rfloor$. We have $\lfloor (n-xd)/2 \rfloor \geq \lfloor (n-xd-1)/2 \rfloor = M$
- The successful malicious node detection probability for the data reconstruction is larger than P_{det}: the probability is $(1-(1-P)^{\alpha\theta_L})^M$, so we have $(1-(1-P)^{\alpha\theta_L})^M > (1-(1-P)^{\theta_L})^M \geq P_{det}$.

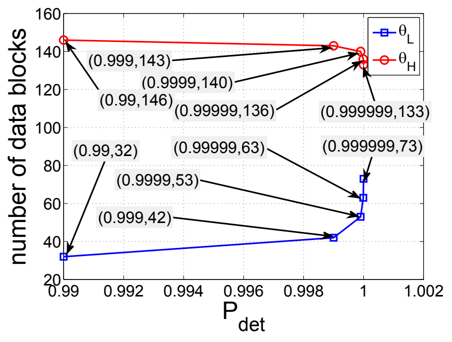

Fig. 1. The number of low/high rate code blocks for different P_{det}

Although the rate match equation (10) 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 low rate code can detect and correct $\lfloor (n - xd)/2 \rfloor$ malicious nodes, which are more than $\lfloor (n - d/2 - 1)/2 \rfloor$ malicious nodes the high rate code can detect. When $x > 0.5$, the high rate code and the low rate code can detect and correct the same number of malicious nodes: $\lfloor (n - \alpha - 1)/2 \rfloor$.

From the analysis above we can see that the same optimized parameters, which are obtained for the data regeneration, can maintain the optimized trade-off between the malicious node detection and storage efficiency for the data reconstruction.

IV. Security Analysis and Performance Evaluation

A. Security Analysis

From the analysis above, we know that for a distributed storage system with n storage nodes out of which at most M nodes are malicious, the rate-matched MSR code can guarantee detection and correction of the malicious nodes during the data regeneration and reconstruction with the probability at least P_{det}.

B. Performance Evaluation

For a distributed storage system with $n = 30$, $M = 11$ and $P = 0.2$, suppose we have a file with the size $B_F = 14000M$ symbols to be stored in the system. The number of the low rate code blocks θ_L and the number of the high rate code blocks θ_H for different detection probabilities P_{det} are shown in Fig. 1. From the figure we can see that the number of low rate code blocks will increase when the detection probability becomes larger. Accordingly, the number of high rate code blocks will decrease.

For the normal error correction MSR code constructed in [8], the efficiency of the code with the same regeneration performance as the rate-matched MSR code is defined as

$$\delta' = \frac{\alpha'(\alpha'+1)}{\alpha' n} = \frac{\alpha'+1}{n} = \frac{xd/2+1}{n}. \qquad (25)$$

In Fig. 2 we will show the efficiency ratios $\eta = \delta/\delta'$ between the rate-matched MSR code and the normal error correction MSR code under different detection probabilities P_{det}. From the figure we can see that the rate-matched MSR code has higher efficiency than the normal error correction MSR code. When the successful malicious nodes detection probability is

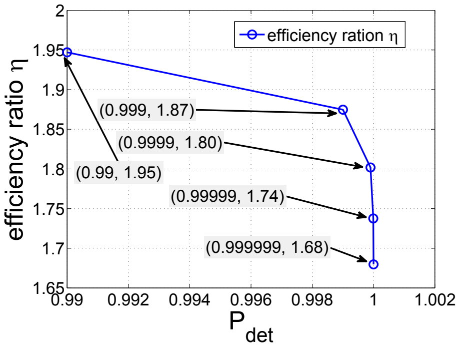

Fig. 2. Efficiency ratios between the rate-matched MSR code and the normal error correction MSR code for different P_{det}

0.999999, the efficiency of the rate-matched MSR code is about 70% higher.

V. Conclusion

In this paper, we develop a rate-matched minimum storage regeneration (MSR) code for malicious nodes detection and correction in hostile networks. We propose the encoding, regeneration and reconstruction algorithms for the rate-matched MSR code. We also optimize the parameters for the data regeneration, considering the trade-off between the malicious nodes detection probability and the storage efficiency. Theoretical analysis shows that the rate-matched MSR code can successfully detect and correct malicious nodes using the optimized parameters. Our analysis also shows that the rate-matched MSR code has higher storage efficiency compared to the normal error correction MSR code (70% higher for the detection probability 0.999999).

References

[1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” *IEEE Transactions on Information Theory*, vol. 56, pp. 4539 – 4551, 2010.

[2] Y. Wu, A. G. Dimakis, and K. Ramchandran, “Deterministic regenerating codes for distributed storage,” in *45th Annu. Allerton Conf. Control, Computing, and Communication*, 2007.

[3] A. Duminuco and E. Biersack, “A practical study of regenerating codes for peer-to-peer backup systems,” in *Distributed Computing Systems, 2009. ICDCS ’09. 29th IEEE International Conference on*, pp. 376 – 384, June 2009.

[4] K. Shum, “Cooperative regenerating codes for distributed storage systems,” in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–5, 2011.

[5] N. Shah, K. Rashmi, P. Kumar, and K. Ramchandran, “Interference alignment in regenerating codes for distributed storage: Necessity and code constructions,” *IEEE Transactions on Information Theory*, vol. 58, pp. 2134 – 2158, 2012.

[6] K. Rashmi, N. Shah, and P. Kumar, “Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction,” *IEEE Transactions on Information Theory*, vol. 57, pp. 5227–5239, 2011.

[7] Y. Han, R. Zheng, and W. H. Mow, “Exact regenerating codes for byzantine fault tolerance in distributed storage,” in *Proceedings IEEE INFOCOM*, pp. 2498 – 2506, 2012.

[8] K. Rashmi, N. Shah, K. Ramchandran, and P. Kumar, “Regenerating codes for errors and erasures in distributed storage,” in *International Symposium on Information Theory (ISIT) 2012*, pp. 1202–1206, 2012.

[9] J. Li, T. Li, and J. Ren, “Beyond the mds bound in distributed cloud storage,” in *IEEE INFOCOM 2014, accepted, to appear.*

[10] N. B. Shah, K. V. Rashmi, K. Ramchandran, and P. V. Kumar, “Privacy-preserving and secure distributed storage codes,” *http://www.eecs.berkeley.edu/~nihar/publications/privacy_security.pdf/*.

[11] J. Li, T. Li, and J. Ren, “Secure regenerating code,” in *IEEE GLOBECOM 2014, accepted, to appear.*