Beyond the MDS Bound in Distributed Cloud Storage

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

***Abstract*—Distributed storage plays a crucial role in the current cloud computing framework. After the theoretical bound for distributed storage was derived by the pioneer work of the regenerating code, Reed-Solomon code based regenerating codes were developed. The RS code based minimum storage regeneration code (RS-MSR) and the RS code based minimum bandwidth regeneration code (RS-MBR) can achieve the theoretical bounds on the MSR point and the MBR point respectively in code regeneration. They can also maintain the MDS property in code reconstruction. However, in the hostile network where the storage nodes can be compromised and the packets can be tampered with, the storage capacity of the network can be significantly affected. In this paper, we propose a Hermitian code based regenerating (H-MSR) code. We first prove that this code can achieve the theoretical MSR bound. We then propose data regeneration and reconstruction algorithms for the H-MSR code in both error-free network and hostile network. Theoretical evaluation shows that our proposed schemes can detect the erroneous decodings and correct more errors in the hostile network than the RS-MSR code with the same code rate. Our analysis also demonstrates that the proposed H-MSR code has a lower complexity than the RS-MSR code in both code regeneration and code reconstruction.**

***Index Terms*—regenerating code, Reed-Solomon code, error-correction, Hermitian code.**

I. Introduction

Cloud storage is an on-demand network data storage and access paradigm. To ensure accessibility of remotely stored data at any time, a typical solution is to store the data across multiple servers or clouds, often in a replicated fashion. Data replication not only lacks flexibility in data recovery, but also requires secure data management for the stored data.

It is well known that security data management is generally very costly and very hard to defend against compromising attacks. Distributed data storage provides an elegant tradeoff between the costly secure data management task and the cheap storage media. The main idea is instead of storing the entire data in one server, we can split the data into n data components. The original data can be recovered only when the required (threshold) number of components, say k, are collected. The original data is information theoretically secure for anyone who can access either an individual component or multiple components when the number of components combined is less than the threshold k. In this case, when the individual components are stored distributively across multiple cloud storage servers, each cloud storage server only needs to assure data integrity and data availability. The costly data encryption and secure key management may no longer be needed any more. The distributed cloud storage can also increase data availability while reducing network congestion that leads to increased resiliency. A popular approach is to employ an (n, k) maximum distance separable (MDS) code such as an Reed-Solomon (RS) code [1], [2]. For RS code, the data is stored in n storage nodes in the network. The data collector (DC) can reconstruct the data by connecting to any k healthy nodes.

While RS code works perfect in reconstructing the data, it lacks scalability in repairing or regenerating a failed node. To deal with this issue, the concept of regenerating code was introduced in [3]. The main idea of the regenerating code is to allow a replacement node to connect to some individual nodes directly and 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 an optimal tradeoff between bandwidth and storage within the minimum storage regeneration (MSR) and the minimum bandwidth regeneration (MBR) points. RS code based MSR (RS-MSR) code and MBR (RS-MBR) code [4] have been explicitly constructed. However, the existing research either has no error detection capability, or has the error correction capability limited by the RS code. Moreover, the schemes with error correction capability are unable to determine whether the error correction is successful.

In this paper, we construct the H-MSR code by combining the Hermitian code and regenerating code at the MSR point. Our proposed code can detect the errors in the network while achieving the error correction capability *beyond the RS code*. Moreover, our proposed code can determine whether the error correction is successful.

The rest of this paper is organized as follows: In Section II, related work is reviewed. In Section III, the preliminary of this paper is presented. In Section IV, our proposed encoding of H-MSR code is described. In Section V, regeneration of the H-MSR code is discussed. Reconstruction of the H-MSR code is analyzed in Section VI. We conduct performance analysis in Section VII. The paper is concluded in Section VIII.

II. Related Work

When a storage node in the distributed storage network that employing the conventional (n, k) RS code (such as OceanStore [1] and Total Recall [2]) fails, the replacement node connects to k nodes and downloads the whole file to

recover 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.* [3] introduced the conception of $\{n, k, d, \alpha, \beta, B\}$ regenerating code. In the context of regenerating code, the contents stored in a failed node can be regenerated by the replacement node through 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. In [3], the authors proved that there is a tradeoff between bandwidth γ and per node storage α. They find two optimal points: minimum storage regeneration (MSR) and minimum bandwidth regeneration (MBR) points.

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 original stored component. [5] formulated the data regeneration as a multicast network coding problem and constructed functional regenerating codes. [6] implemented a random linear regenerating codes for distributed storage systems. [7] proved that by allowing data exchange among the replacement nodes, a better tradeoff between repair bandwidth γ and per node storage α can be achieved. In the exact regeneration, the replacement node regenerates the exact symbols of a failed node. [8] proposed to reduce the regeneration bandwidth through algebraic alignment. [9] provided a code structure for exact regeneration using interference alignment technique. [4] presented optimal exact constructions of MBR codes and MSR codes under product-matrix framework. This is the first work that allows independent selection of the nodes number n in the network.

However, none of these works above considered code regeneration under node corruption or adversarial manipulation attacks. In fact, all these schemes will fail in both regeneration and reconstruction.

In [10], the authors discussed the amount of information that can be safely stored against passive eavesdropping and active adversarial attacks based on the regeneration structure. In [11], 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 also be manipulated by the malicious nodes, resulting in the failure of the regeneration and reconstruction. In [12], the authors analyzed the error resilience of the RS code based regenerating code in the network with errors and erasures. They provided the theoretical error correction capability. Their result is an extension of the MDS code to the regenerating code and their scheme is unable to determine whether the errors in the network are successfully corrected.

In this paper, we proposed a Hermitian code based minimum storage regeneration (H-MSR) code. The proposed H-MSR code can correct more errors than the RS-MSR code and can always determine whether the error correction is successful. Our design is based on the structural analysis of the Hermitian code and the efficient decoding algorithm proposed in [13].

III. Preliminary and Assumptions

A. Regenerating Code

Regenerating code introduced in [3] is a linear code over 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 $k \leq d$ randomly selected storage nodes. In [3], the following theoretical bound was derived:

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

From equation (1), a tradeoff 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. Hermitian Code

A Hermitian curve $\mathcal{H}(q)$ over $GF(q^2)$ in affine coordinates is defined by:

$$\mathcal{H}(q) : y^q + y = x^{q+1}. \tag{4}$$

The genus of $\mathcal{H}(q)$ is $g = (q^2 - q)/2$ and there are q^3 points that satisfy equation (4), denoted as $P_{0,0}, \ldots, P_{0,q-1}, \ldots, P_{q^2-1,0}, \ldots, P_{q^2-1,q-1}$ (See Table I), where $\theta_0, \theta_1, \cdots, \theta_{q-1}$ are the q solutions to $y^q + y = 0$ and ϕ is a primitive element in $GF(q^2)$. $L(mQ)$ is defined as:

$$\begin{aligned} L(mQ) \quad = \quad & \{f_0(x) + y f_1(x) + \cdots + y^{q-1} f_{q-1}(x) | \\ & \deg f_j(x) < \kappa(j), j = 0, 1, \cdots, q-1\}, \end{aligned} \tag{5}$$

where

$$\kappa(j) = \max\{t | tq + j(q+1) \leq m\} + 1, \tag{6}$$

for $m \geq q^2 - 1$. A codeword of the Hermitian code [13] $\mathcal{H}_m$ is defined as $(g(P_{0,0}), \ldots, g(P_{0,q-1}), \ldots, g(P_{q^2-1,0}), \ldots, g(P_{q^2-1,q-1}))$, where $g \in L(mQ)$. The dimension of the message before encoding can be calculated as $\dim(\mathcal{H}_m) = \sum_{j=0}^{j=q-1} (\deg f_j(x) + 1)$.

TABLE I
q^3 RATIONAL POINTS OF THE HERMITIAN CURVE

$P_{0,0}=(0,\theta_0)$	$P_{1,0}=(1,\phi+\theta_0)$	$\cdots$	$P_{q^2-1,0}=(\phi^{q^2-2},\phi^{(q^2-2)(q+1)+1})+\theta_0$
$P_{0,1}=(0,\theta_1)$	$P_{1,1}=(1,\phi+\theta_1)$	$\cdots$	$P_{q^2-1,1}=(\phi^{q^2-2},\phi^{(q^2-2)(q+1)+1})+\theta_1$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$P_{0,q-1}=(0,\theta_{q-1})$	$P_{1,q-1}=(1,\phi+\theta_{q-1})$	$\cdots$	$P_{q^2-1,q-1}=(\phi^{q^2-2},\phi^{(q^2-2)(q+1)+1})+\theta_{q-1}$

C. Adversarial Model

In this paper, we assume some network nodes can be corrupted due to hardware failure or communication errors, and/or be compromised by malicious users. As a result, upon request, these nodes may send out incorrect response to disrupt the data regeneration and reconstruction.

We will refer these symbols as *bogus* symbols without making distinction between the corrupted symbols and compromised symbols. We will also use corrupted nodes, malicious nodes and compromised nodes interchangeably without making any distinction.

IV. ENCODING H-MSR CODE

In this section, we will analyze the H-MSR code for $d = 2k-2 = 2\alpha$. The code with $d > 2k-2$ can be derived the same way through truncating operations.

Let $\alpha_0, \cdots, \alpha_{q-1}$ be a strictly decreasing integer sequence satisfying $0 < \alpha_i \leq \kappa(i)$, $0 \leq i \leq q-1$. The least common multiple of $\alpha_0, \cdots, \alpha_{q-1}$ is A. Suppose the data contains $B = A \cdot \sum_{i=0}^{q-1}(\alpha_i + 1)$ message symbols from the finite field $GF(q^2)$. In practice, if the actual data is larger than B symbols, we can fragment it into blocks of size B and process each block individually.

We arrange the B symbols into two matrices S, T as below:

$$S = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{q-1} \end{bmatrix}, \quad T = \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{q-1} \end{bmatrix}, \tag{7}$$

where

$$\begin{aligned} S_i &= [S_{\alpha_i,1}, S_{\alpha_i,2}, \cdots, S_{\alpha_i,A/\alpha_i}], \\ T_i &= [T_{\alpha_i,1}, T_{\alpha_i,2}, \cdots, T_{\alpha_i,A/\alpha_i}]. \end{aligned} \tag{8}$$

$S_{\alpha_i,j}$, $0 \leq i \leq q-1, 1 \leq j \leq A/\alpha_i$, is a symmetric matrix of size $\alpha_i \times \alpha_i$ with the upper-triangular entries filled by data symbols. Thus $S_{\alpha_i,j}$ contains $\alpha_i \cdot (\alpha_i+1)/2$ symbols. The number of columns of each submatrix S_i, $0 \leq i \leq q-1$, is the same: $\alpha_i \cdot A/\alpha_i = A$. The size of matrix S is $(\sum_{i=0}^{q-1}\alpha_i) \times A$. So it contains $\sum_{i=0}^{q-1}(\alpha_i \cdot (\alpha_i+1)/2) \cdot A/\alpha_i = (A \cdot \sum_{i=0}^{q-1}(\alpha_i + 1))/2$ data symbols.

$T_{\alpha_i,j}$, $0 \leq i \leq q-1, 1 \leq j \leq A/\alpha_i$, is constructed the same as $S_{\alpha_i,j}$. So T has the same structure as S and contains the other $(A \cdot \sum_{i=0}^{q-1}(\alpha_i+1))/2$ data symbols.

Definition 1. *For a Hermitian code $\mathcal{H}_m$ over $GF(q^2)$, we encode matrix $M_{\dim(\mathcal{H}_m)\times A} = [M_1, M_2 \cdots, M_A]$ by encoding each column M_i, $i = 1, 2, \cdots, A$, individually using $\mathcal{H}_m$. The codework matrix $\mathcal{H}_m(M) = [\mathcal{H}_m(M_1), \mathcal{H}_m(M_2), \cdots, \mathcal{H}_m(M_A)]$, where $\mathcal{H}_m(M_i)$ has the following form ($g \in L(mQ)$):*

$$[g(P_{0,0}), \cdots, g(P_{0,q-1}), \cdots, g(P_{q^2-1,0}), \cdots, g(P_{q^2-1,q-1})]^T.$$

For distributed storage, we encode each pair of matrices (S, T) using Algorithm 1. Let Γ be a $q^3 \times q^3$ diagonal matrix formed by q^2 diagonal submatrices $\Lambda_0, \cdots, \Lambda_{q^2-1}$ of size $q \times q$. The q diagonal elements in Λ_i, $0 \leq i \leq q^2-1$, are identical to $\lambda_i \in GF(q^2)$. λ_i is chosen using the following two criteria: (i) $\lambda_i \neq \lambda_j$, $\forall i \neq j$, $0 \leq i, j \leq q^2-1$. (ii) Any $d_i = 2\alpha_i$ rows of the matrix $[\Phi_i, \Delta \cdot \Phi_i]$, $0 \leq i \leq q-1$, are linearly independent, where

$$\Delta = \begin{bmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & \cdots & 0 \\ \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & \cdots & \lambda_{q^2-1} \end{bmatrix}, \tag{9}$$

$$\Phi_i = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & \phi & \phi^2 & \cdots & \phi^{\alpha_i-1} \\ \cdots & \cdots & \cdots & \ddots & \cdots \\ 1 & \phi^{q^2-2} & (\phi^{q^2-2})^2 & \cdots & (\phi^{q^2-2})^{\alpha_i-1} \end{bmatrix}. \tag{10}$$

Algorithm 1 Encoding H-MSR Code

1: Encode the data matrices S, T defined above using a Hermitian code $\mathcal{H}_m$ over $GF(q^2)$ with parameters $\kappa(j)$ $(0 \leq j \leq q-1)$ and m $(m \geq q^2-1)$.
2: Calculate the $q^3 \times A$ codeword matrix $Y = \mathcal{H}_m(S) + \Gamma\mathcal{H}_m(T)$.
3: Divide Y into q^2 submatrices $Y_0, \cdots, Y_{q^2-1}$ of the size $q \times A$. Store one submatrix in each of the q^2 storage nodes.

Theorem 1. *By processing the data symbols using Algorithm 1, we can achieve the MSR point in distributed storage.*

Proof: We first study the structure of the codeword matrix $\mathcal{H}_m(S)$. Since every column of the matrix is an independent Hermitian codeword, we can decode the first column $\mathbf{h} = [h_{0,0}, \cdots, h_{0,q-1}, \cdots, h_{q^2-1,0}, \cdots, h_{q^2-1,q-1}]^T$ as an example without loss of generality. We arrange the q^3 rational points of the Hermitian curve following the order in Table I. In the table, we can find that for each i, $i = 0, 1, \cdots, q^2-1$,

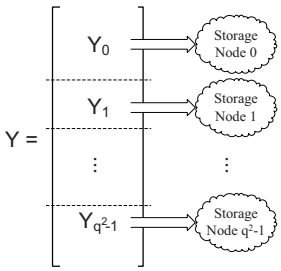

Fig. 1. Store the codeword matrices in distributed storage nodes

the rational points $P_{i,0}, P_{i,1}, \cdots, P_{i,q-1}$ all have the same first coordinate.

Suppose $g \in L(mQ)$: $g(P_{i,l}) = f_0(P_{i,l}) + y(P_{i,l})f_1(P_{i,l}) + \cdots + (y(P_{i,l}))^{q-1}f_{q-1}(P_{i,l})$, $0 \le i \le q^2-1$, $0 \le l \le q-1, \deg f_j(x) = \alpha_j - 1$ for $0 \le j \le q-1$. Since $P_{i,0}, P_{i,1}, \cdots, P_{i,q-1}$ all have the same first coordinate and $f_j(P_{i,l})$ is only applied to the first coordinate of $P_{i,l}$, we have $f_j(P_{i,l}) = f_j(\phi^{s_i})$, $s_0 = -\infty, s_i = i-1$, for $i \ge 1, \phi^{-\infty} = 0$, which does not depend on l. Therefore, we can derive q^2 sets of equations for $0 \le i \le q^2-1$:

$$\begin{cases} f_0(\phi^{s_i}) + y(P_{i,0})f_1(\phi^{s_i}) + \cdots + (y(P_{i,0}))^{q-1}f_{q-1}(\phi^{s_i}) = h_{i,0} \\ f_0(\phi^{s_i}) + y(P_{i,1})f_1(\phi^{s_i}) + \cdots + (y(P_{i,1}))^{q-1}f_{q-1}(\phi^{s_i}) = h_{i,1} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ f_0(\phi^{s_i}) + y(P_{i,q-1})f_1(\phi^{s_i}) + \cdots + (y(P_{i,q-1}))^{q-1}f_{q-1}(\phi^{s_i}) = h_{i,q-1} \end{cases}. \tag{11}$$

If we store the codeword matrix in storage nodes according to Fig. 1, each set of the equations corresponds to a storage node. As we mentioned above, the set of equations in equation (11) can be derived in storage node i.

Since the coefficient matrix B_i is a Vandermonde matrix:

$$B_i = \begin{bmatrix} 1 & y(P_{i,0}) & \cdots & y(P_{i,0})^{q-1} \\ 1 & y(P_{i,1}) & \cdots & y(P_{i,1})^{q-1} \\ \cdots & \cdots & \ddots & \cdots \\ 1 & y(P_{i,q-1}) & \cdots & y(P_{i,q-1})^{q-1} \end{bmatrix}. \tag{12}$$

We can solve $\mathbf{u}_i = [f_0(\phi^{s_i}), f_1(\phi^{s_i}), \cdots, f_{q-1}(\phi^{s_i})]^T$ from

$$\mathbf{u}_i = B_i^{-1} \cdot \mathbf{h}_i, \tag{13}$$

where $\mathbf{h}_i = [h_{i,0}, h_{i,1}, \cdots, h_{i,q-1}]^T$.

From all the q^2 storage nodes, we can get vectors $\mathcal{F}_i = [f_i(0), f_i(1), \cdots, f_i(\phi^{q^2-2})]^T$, $i = 0, \cdots, q-1$, which can be viewed as extended Reed-Solomon codes.

Now consider all the columns of $\mathcal{H}_m(S)$, we can get the following equation:

$$\Phi_i \cdot S_{\alpha_i,j} = F_{i,j}, \tag{14}$$

where $F_{i,j} = [\mathcal{F}_i^{(1)}, \cdots, \mathcal{F}_i^{(\alpha_i)}]$, $0 \le i \le q-1, 1 \le j \le A/\alpha_i$, and $\mathcal{F}_i^{(l)}$ corresponds to the l^{th} column of the submatrix $S_{\alpha_i,j}$.

Next we will consider the structure of the codeword matrix $\mathcal{H}_m(T)$. Because the encoding process for $\mathcal{H}_m(T)$ is the same as that of $\mathcal{H}_m(S)$, for $\Gamma \cdot \mathcal{H}_m(T)$, we can derive

$$\Delta \cdot \Phi_i \cdot T_{\alpha_i,j} = \Delta \cdot E_{i,j}, \tag{15}$$

where $\mathcal{E}_i = [e_i(0), e_i(1), \cdots, e_i(\phi^{q^2-2})]^T$, $E_{i,j} = [\mathcal{E}_i^{(1)}, \cdots, \mathcal{E}_i^{(\alpha_i)}]$, $0 \le i \le q-1$, $1 \le j \le A/\alpha_i$, and $\mathcal{E}_i^{(l)}$ corresponds to the l^{th} column of the submatrix $T_{\alpha_i,j}$.

Thirdly, we will study the optimality of the code in the sense of the MSR point. For $\Phi_i \cdot S_{\alpha_i,j} + \Delta \cdot \Phi_i \cdot T_{\alpha_i,j}$, $0 \le i \le q-1, 1 \le j \le A/\alpha_i$, since $S_{\alpha_i,j}, T_{\alpha_i,j}$ are symmetric and satisfy the requirements for MSR point according to [4] with parameters $d = 2\alpha_i, k = \alpha_i + 1, \alpha = \alpha_i, \beta = 1, B = \alpha_i \cdot (\alpha_i + 1)$. By encoding S, T using $\mathcal{H}_m(S) + \Gamma \cdot \mathcal{H}_m(T)$ and distributing $Y_0, \cdots, Y_{q^2-1}$ into q^2 storage nodes, each row of the matrix $\Phi_i \cdot S_{\alpha_i,j} + \Delta \cdot \Phi_i \cdot T_{\alpha_i,j}$, $0 \le i \le q-1$, $1 \le j \le A/\alpha_i$, can be derived in a corresponding storage node. Because $\Phi_i \cdot S_{\alpha_i,j} + \Delta \cdot \Phi_i \cdot T_{\alpha_i,j}$ achieves the MSR point, data related to matrices $S_{\alpha_i,j}, T_{\alpha_i,j}$, $0 \le i \le q-1$, $1 \le j \le A/\alpha_i$, can be regenerated at the MSR point. Therefore, Algorithm 1 can achieve the MSR point. ■

V. Regeneration of the H-MSR Code

In this section, we will first discuss the regeneration for the H-MSR code in error-free network. Then we will discuss the regeneration in hostile network.

A. Regeneration in Error-free Network

Let $\mathbf{v}_i = [e_0(\phi^{(s_i)}), e_1(\phi^{(s_i)}), \cdots, e_{q-1}(\phi^{(s_i)})]^T$, then $\mathbf{u}_i + \Lambda_i \cdot \mathbf{v}_i = B_i^{-1} \cdot \mathbf{y}_i = [f_0(\phi^{s_i}) + \lambda_i e_0(\phi^{s_i}), \cdots, f_{q-1}(\phi^{s_i}) + \lambda_i e_{q-1}(\phi^{s_i})]^T$, for every column $\mathbf{y}_i$ of Y_i.

The main idea of the regeneration algorithms is to regenerate $f_l(\phi^{s_i}) + \lambda_i e_l(\phi^{s_i})$, $0 \le l \le q-1$, by downloading help symbols from $d_l = 2\alpha_l$ nodes. Since $d_{l_1} > d_{l_2}$ for $l_1 < l_2$, for efficiency consideration, only d_{q-1} helper nodes need to send out symbols calculated by $[f_0(\phi^{s_i}) + \lambda_i e_0(\phi^{s_i}), f_1(\phi^{s_i}) + \lambda_i e_1(\phi^{s_i}), \ldots, f_{q-1}(\phi^{s_i}) + \lambda_i e_{q-1}(\phi^{s_i})]^T$. Then $d_j - d_{j+1}$ nodes only need to send out symbols calculated by $[f_0(\phi^{s_i}) + \lambda_i e_0(\phi^{s_i}), f_1(\phi^{s_i}) + \lambda_i e_1(\phi^{s_i}), \ldots, f_j(\phi^{s_i}) + \lambda_i e_j(\phi^{s_i})]^T$ for $0 \le j \le q-2$. In this way, the total number of helper nodes that send out symbols calculated by $f_l(\phi^{s_i}) + \lambda_i e_l(\phi^{s_i})$ is $d_{q-1} + \sum_{j=l}^{q-2}(d_j - d_{j+1}) = d_l$.

Suppose node z fails, we devise Algorithms 2 to Algorithm 4 in the network to regenerate the exact H-MSR code symbols of node z in a replacement node z'. For convenience, we suppose $d_q = 2\alpha_q = 0$ and define

$$\mathbf{V}_{i,j,l} = \begin{bmatrix} \nu_{i,l} \\ \nu_{i+1,l} \\ \cdots \\ \nu_{j,l} \end{bmatrix}, \tag{16}$$

where $\nu_{t,l}$, $i \le t \le j$, is the t^{th} row of $[\Phi_l, \Delta \cdot \Phi_l]$.

Algorithm 2 z' sends requests for H-MSR regeneration

1: **for** $j = q-1 \to 0$ **do**
2: z' sends the integer j to $d_j - d_{j+1} = 2\alpha_j - 2\alpha_{j+1}$ helper nodes.
3: **end for**

From Algorithm 2 to Algorithm 4, we can derive the equivalent storage parameters for each symbol block of size

Algorithm 3 Helper node i sends help symbols for H-MSR

Require: Integer j sent from node z'.
Ensure: Corresponding help symbols.
1: Calculate $\tilde{Y}_i = B_i^{-1} \cdot Y_i$.
2: **for** $l = 0 \to j$ **do**
3: Divide the $(l+1)^{th}$ row of $\tilde{Y}_i$ into A/α_l equal row vectors: $[\tilde{\mathbf{y}}_{i,l,1}, \tilde{\mathbf{y}}_{i,l,2}, \ldots, \tilde{\mathbf{y}}_{i,l,A/\alpha_l}]$.
4: **for** $t = 1 \to A/\alpha_l$ **do**
5: Calculate $\tilde{p}_{i,l,t} = \tilde{\mathbf{y}}_{i,l,t} \cdot \mu_{z,l}^T$, where $\mu_{z,l}$ is the z^{th} row of the encoding matrix Φ_l defined in equation (10).
6: **end for**
7: **end for**
8: Send out the help symbols $\tilde{p}_{i,l,t}$ $(0 \le l \le j, 1 \le t \le A/\alpha_l)$

Algorithm 4 z' regenerates symbols of the failed node z

Require: All the symbols requested in Algorithm 2.
Ensure: Exact symbols stored in node z: Y_z.
1: **for** $l = 0 \to q-1$ **do**
2: **for** $t = 1 \to A/\alpha_l$ **do**
3: Suppose $\tilde{p}_{i,l,t}$ is the response from i^{th} helper node. Without loss of generality, we assume $0 \le i \le d_l - 1$. Let $\mathbf{p} = [\tilde{p}_{0,l,t}, \tilde{p}_{1,l,t}, \cdots, \tilde{p}_{d_l-1,l,t}]^T$.
4: Solve the equation: $\mathbf{V}_{0,d_l-1,l} \cdot \mathbf{x} = \mathbf{p}$.
5: Since $\mathbf{x} = \begin{bmatrix} S_{\alpha_l,t} \\ T_{\alpha_l,t} \end{bmatrix} \cdot \mu_{z,l}^T$ and $S_{\alpha_l,t}, T_{\alpha_l,t}$ are symmetric, we can calculate $\mathbf{x}^T = [\mu_{z,l} \cdot S_{\alpha_l,t}, \mu_{z,l} \cdot T_{\alpha_l,t}]$.
6: Compute $\tilde{\mathbf{y}}_{z,l,t} = \mu_{z,l} \cdot S_{\alpha_l,t} + \lambda_z \cdot \mu_{z,l} \cdot T_{\alpha_l,t} = \nu_{z,l} \cdot \begin{bmatrix} S_{\alpha_l,t} \\ T_{\alpha_l,t} \end{bmatrix}$.
7: **end for**
8: **end for**
9: Let $\tilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $[\tilde{\mathbf{y}}_{z,l,1}, \cdots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}]$, $0 \le l \le q-1$.
10: Calculate $Y_{z'} = Y_z = B_z \cdot \tilde{Y}_z$.

$B_j = A \cdot (\alpha_j + 1), d_j = 2\alpha_j, k_j = \alpha_j + 1, \alpha_j = A, \beta_j = A/\alpha_j$, $0 \le j \le q-1$ and equation (2) of the MSR point holds for these parameters. Theorem 1 guarantees that Algorithm 2 to Algorithm 4 can achieve the MSR point for data regeneration of the H-MSR code.

B. Regeneration in the Hostile Network

In a hostile network, Algorithm 4 may be unable to regenerate the failed node due to the possible bogus symbols received from the responses. In fact, even if the replacement node z' can derive the symbol matrix $Y_{z'}$ using Algorithm 4, it cannot verify the correctness of the result.

There are two modes for the helper nodes to regenerate the contents of a failed storage node in the hostile network. One mode is the detection mode, in which no error has been found in the symbols received from the helper nodes. Once errors are detected, the recovery mode will be used to correct the errors and locate the malicious nodes.

1) Detection Mode: In the detection mode, the replacement node z' will send requests using the algorithm similar to Algorithm 2. The only difference is that when $j = q-1$, z' sends requests to $d_{q-1} - d_q + 1$ nodes instead of $d_{q-1} - d_q$ nodes. Helper nodes will still use Algorithm 3 to send the help symbols. The regeneration algorithm is described in Algorithm 5 with the detection probability characterized in the following theorem.

Algorithm 5 z' regenerates symbols of the failed node z in hostile network – detection mode

Require: All the symbols requested.
Ensure: The exact symbols stored in node z: Y_z, in case no error is detected; otherwise flag for errors.
1: **for** $l = 0 \to q-1$ **do**
2: **for** $t = 1 \to A/\alpha_l$ **do**
3: Suppose $\tilde{p}'_{i,l,t} = \tilde{p}_{i,l,t} + e_{i,l,t}$ is the response from the i^{th} helper node. If $\tilde{p}_{i,l,t}$ has been modified by the malicious node i, we have $e_{i,l,t} \in GF(q^2) \backslash \{0\}$. Otherwise we have $e_{i,l,t} = 0$. Without loss of generality, we assume $0 \le i \le d_l$. Let $\mathbf{p}' = [\tilde{p}'_{0,l,t}, \tilde{p}'_{1,l,t}, \cdots, \tilde{p}'_{d_l-1,l,t}, \tilde{p}'_{d_l,l,t}]^T$.
4: Solve $\mathbf{V}_{0,d_l-1,l} \cdot \mathbf{x}_1 = \mathbf{p}_1{}'$ using symbols collected from node 0 to node $d_l - 1$ where $\mathbf{p}_1{}' = [\tilde{p}'_{0,l,t}, \tilde{p}'_{1,l,t}, \cdots, \tilde{p}'_{d_l-1,l,t}]^T$.
5: Solve $\mathbf{V}_{1,d_l,l} \cdot \mathbf{x}_2 = \mathbf{p}_2{}'$ using symbols collected from node 1 to node d_l where $\mathbf{p}_2{}' = [\tilde{p}'_{1,l,t}, \tilde{p}'_{2,l,t}, \cdots, \tilde{p}'_{d_l,l,t}]^T$.
6: **if** $\mathbf{x}_1 = \mathbf{x}_2$ **then**
7: Compute $\tilde{\mathbf{y}}_{z,l,t} = \mu_{z,l} \cdot S_{\alpha_l,t} + \lambda_z \cdot \mu_{z,l} \cdot T_{\alpha_l,t}$ as described in Algorithm 4.
8: **else**
9: Switch to recovery regeneration mode and exit the algorithm.
10: **end if**
11: **end for**
12: **end for**
13: Let $\tilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $[\tilde{\mathbf{y}}_{z,l,1}, \cdots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}]$, $0 \le l \le q-1$.
14: Calculate $Y_{z'} = Y_z = B_z \cdot \tilde{Y}_z$.

Theorem 2. *For bogus symbols $\tilde{p}'_{i,l,t}$, $l = 0, \cdots, q-1, t = 1, \cdots, A/\alpha_l$, received from the malicious nodes, the probability for the bogus symbols to be detected using Algorithm 5 is at least $1 - 1/q^2$.*

Proof: Since $\mathbf{V}_{0,d_l-1,l}$ and $\mathbf{V}_{1,d_l,l}$ are full rank matrices, $\mathbf{x}_1$ can be calculated by (For convenience, use e_i to represent $e_{i,l,t}$):

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{V}_{0,d_l-1,l}^{-1} \cdot \left[\tilde{p}_{0,l,t} + e_0, \cdots, \tilde{p}_{d_l-1,l,t} + e_{d_l-1}\right]^T \\ &= \mathbf{x} + \mathbf{V}_{0,d_l-1,l}^{-1} \cdot [e_0, e_1, \cdots, e_{d_l-1}]^T \\ &= \mathbf{x} + \hat{\mathbf{x}}_1. \end{aligned} \quad (17)$$

$\mathbf{x}_2$ can be calculated the same way:

$$\mathbf{x}_2 = \mathbf{x} + \mathbf{V}_{1,d_l,l}^{-1} \cdot [e_1, e_2, \cdots, e_{d_l}]^T = \mathbf{x} + \hat{\mathbf{x}}_2. \quad (18)$$

If $\hat{\mathbf{x}}_1 = \hat{\mathbf{x}}_2$, equivalently $\mathbf{V}_{0,d_l-1,l} \cdot \hat{\mathbf{x}}_1 = \mathbf{V}_{0,d_l-1,l} \cdot \hat{\mathbf{x}}_2$, Algorithm 5 will fail to detect the errors. So we will focus on the relationship between $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$.

First, we have

$$\mathbf{V}_{0,d_l-1,l} \cdot \hat{\mathbf{x}}_1 = [e_0, e_1, \cdots, e_{d_l-1}]^T. \tag{19}$$

Suppose $\mathbf{V}_{1,d_l,l}^{-1} = [\eta_0, \eta_1, \cdots, \eta_{d_l-1}]$, then we have:

$$\nu_{r,l} \cdot \eta_s = \begin{cases} 1, & r = s+1 \\ 0, & r \neq s+1 \end{cases}, \ 1 \leq r \leq d_l, 0 \leq s \leq d_l - 1. \tag{20}$$

$$\begin{aligned} &\mathbf{V}_{0,d_l-1,l} \cdot \hat{\mathbf{x}}_2 = \mathbf{V}_{0,d_l-1,l} \cdot \mathbf{V}_{1,d_l,l}^{-1} \cdot [e_1, e_2, \cdots, e_{d_l}]^T \\ &= \mathbf{V}_{0,d_l-1,l} \cdot [\eta_0, \eta_1, \cdots, \eta_{d_l-1}] \cdot [e_1, e_2, \cdots, e_{d_l}]^T \\ &= [x_{2,0}, e_1, \cdots, e_{d_l-1}]^T. \end{aligned} \tag{21}$$

To calculate $x_{2,0}$, we first derive the expression of $\nu_{0,l}$. Because $\nu_{1,l}, \nu_{2,l}, \cdots, \nu_{d_l,l}$ are linearly independent, they can be viewed as a set of bases of the d_l dimensional linear space. So we have

$$\nu_{0,l} = \sum_{r=1}^{d_l} \zeta_r \cdot \nu_{r,l}, \tag{22}$$

where $\zeta_r \neq 0$, $r = 1, \cdots, d_l$, because any d_l vectors out of $\nu_{0,l}, \nu_{1,l}, \cdots, \nu_{d_l,l}$ are linearly independent. Then

$$\begin{aligned} x_{2,0} &= \left(\sum_{r=1}^{d_l} \zeta_r \cdot \nu_{r,l} \right) [\eta_0, \eta_1, \cdots, \eta_{d_l-1}][e_1, e_2, \cdots, e_{d_l}]^T \\ &= \sum_{r=1}^{d_l} \zeta_r \cdot e_r. \end{aligned} \tag{23}$$

If

$$e_0 = \sum_{r=1}^{d_l} \zeta_r \cdot e_r, \tag{24}$$

then $\mathbf{V}_{0,d_l-1,l} \cdot \hat{\mathbf{x}}_1 = \mathbf{V}_{0,d_l-1,l} \cdot \hat{\mathbf{x}}_2$ and $\hat{\mathbf{x}}_1 = \hat{\mathbf{x}}_2$. Therefore, Algorithm 5 will fail.

When only one symbol in the $d_l + 1$ symbols is bogus, then only one element of $e_0, e_1, \cdots, e_{d_l}$ will be nonzero. Since $\zeta_1, \cdots, \zeta_{d_l}$ are all nonzero, equation (24) will never hold. In this case, the detection probability is 1.

Since the malicious nodes create bogus symbol independently, we can treat $e_0, e_1, \cdots, e_{d_l}$ as independent random numbers from $GF(q^2)$. Thus when there are more than one bogus symbols, the probability that equation (24) holds is $1/q^2$. In this case, the detection probability is $1 - 1/q^2$. ■

2) Recovery Mode: Once the replacement node z' detects errors using Algorithm 5, it will send integer $j = q - 1$ to all the other $q^2 - 1$ nodes in the network requesting help symbols. Helper node i will send help symbols using Algorithm 3. z' can regenerate symbols using Algorithm 6.

VI. Reconstruction of the H-MSR code

In this section, we will first discuss the reconstruction of the H-MSR code in error-free network. Then we will discuss the reconstruction when there are corrupted nodes in the network.

Algorithm 6 z' regenerates symbols of the failed node z in hostile network – Recovery mode

Require: All the symbols requested.

Ensure: The node identities of the malicious nodes and the exact symbols stored in node z: Y_z, in case the errors can be corrected.

1: **for** $l = q - 1 \to 0$ **do**
2: **for** $t = 1 \to A/\alpha_l$ **do**
3: Suppose $\tilde{p}'_{i,l,t}$ is the response from the i^{th} helper node, without loss of generality, we assume $0 \leq i \leq q^2 - 2$. Let $\mathbf{p}' = [\tilde{p}'_{0,l,t}, \tilde{p}'_{1,l,t}, \cdots, \tilde{p}'_{q^2-2,l,t}]^T$.
4: Since $\mathbf{V}_{0,q^2-2,l} \cdot \mathbf{x} = \mathbf{p}'$, $\mathbf{p}'$ can be viewed as an MDS code with parameters $(q^2 - 1, d_l, q^2 - d_l)$.
5: Substitute $\tilde{p}'_{i,l,t}$ in $\mathbf{p}'$ with the symbol $\otimes$ representing an erasure if node i has been detected to be corrupted in the previous loops.
6: **if** # of erasures $> \min\{q^2 - d_l - 1, \lfloor (q^2 - d_{q-1} - 1)/2 \rfloor\}$ **then**
7: Flag the errors cannot be corrected and exit the algorithm.
8: **end if**
9: Decode $\mathbf{p}'$ to $\mathbf{p}'_{cw}$ and recover $\mathbf{x}$.
10: Mark node i as corrupted if the i^{th} position of $\mathbf{p}'_{cw}$ and $\mathbf{p}'$ are different.
11: Compute $\tilde{\mathbf{y}}_{z,l,t} = \mu_{z,l} \cdot S_{\alpha_l,t} + \lambda_z \cdot \mu_{z,l} \cdot T_{\alpha_l,t}$ as described in Algorithm 4.
12: **end for**
13: **end for**
14: Let $\tilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $[\tilde{\mathbf{y}}_{z,l,1}, \cdots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}]$, $0 \leq l \leq q - 1$.
15: Calculate $Y_{z'} = Y_z = B_z \cdot \tilde{Y}_z$.

A. Reconstruction in Error-free Network

In this subsection, we devise Algorithm 7, Algorithm 8 and Algorithm 9 in the network for the data collector DC to reconstruct the original file. For convenience, we suppose $\alpha_q = 0$.

Algorithm 7 DC sends requests for H-MSR reconstruction

1: **for** $j = q - 1 \to 0$ **do**
2: z' sends the integer j to $k_j - k_{j+1} = \alpha_j - \alpha_{j+1}$ helper nodes, where $k_j = \alpha_j + 1$.
3: **end for**

Algorithm 8 Node i sends symbols for reconstruction

Require: Integer j sent from DC.

Ensure: Corresponding symbols for reconstruction.

1: Calculate $\tilde{Y}_i = B_i^{-1} \cdot Y_i$.
2: **for** $l = 0 \to j$ **do**
3: Send out the $(l+1)^{th}$ row of $\tilde{Y}_i$: $\tilde{\mathbf{y}}_{i,l}$.
4: **end for**

Algorithm 9 DC reconstructs the original file

Require: All the symbols requested in Algorithm 7.
Ensure: The original file.
1: **for** $l = 0 \to q-1$ **do**
2: Divide $\tilde{\mathbf{y}}_{i,l}$ into A/α_l equal row vectors: $[\tilde{\mathbf{y}}_{i,l,1}, \tilde{\mathbf{y}}_{i,l,2}, \cdots, \tilde{\mathbf{y}}_{i,l,A/\alpha_l}]$.
3: **for** $t = 1 \to A/\alpha_l$ **do**
4: Suppose the symbol vector $\tilde{\mathbf{y}}_{i,l,t}$ is the response from the i^{th} helper node, without loss of generality, we assume $0 \le i \le k_l - 1$. Let $R = [\tilde{\mathbf{y}}_{0,l,t}^T, \tilde{\mathbf{y}}_{1,l,t}^T, \cdots, \tilde{\mathbf{y}}_{k_l-1,l,t}^T]^T$.
5: $\mathbf{V}_{0,k_l-1,l} \cdot \begin{bmatrix} S_{\alpha_l,t} \\ T_{\alpha_l,t} \end{bmatrix} = R$.
6: DC reconstructs $S_{\alpha_l,t}, T_{\alpha_l,t}$ using techniques similar to [4].
7: **end for**
8: **end for**
9: DC reconstruct the original file from S, T.

B. Reconstruction in Hostile Network

Similar to the regeneration algorithms, the reconstruction algorithms in the error-free network do not work in the hostile network. Even if the data collecter can calculate the symbol matrices S, T using Algorithm 9, it cannot verify whether the result is correct or not. There are two modes for the original file to be reconstructed in the hostile network. One mode is the detection mode, in which no error has been found in the symbols received from the storage nodes. Once errors are detected in the detection mode, the recovery mode will be used to correct the errors and locate the malicious nodes.

1) Detection Mode: In the detection mode, DC will send requests using the algorithm similar to Algorithm 7. The only difference is that when $j = q-1$, DC will send requests to $k_{q-1} - k_q + 1$ nodes instead of $k_{q-1} - k_q$ nodes. Storage nodes will still send symbols using Algorithm 8. The reconstruction algorithm is described in Algorithm 10 with the detection probability described in Theorem 3.

Theorem 3. *For bogus symbols $\tilde{\mathbf{y}}'_{i,l,t}$ received from the malicious nodes, $l = 0, \cdots, q-1, t = 1, \cdots, A/\alpha_l$, the probability for the bogus symbols to be detected using Algorithm 10 is at least $1 - (1/q^2)^{\alpha_l - 1}$.*

Due to page limitation, the proof is omitted here.

2) Recovery Mode: Once DC detects errors using Algorithm 10, it will send integer $j = q-1$ to all the q^2 nodes in the network requesting symbols. Helper node i will send symbols using Algorithm 8. The reconstruct procedures are described in Algorithm 11.

C. Recover Matrices $S_{\alpha_l,t}, T_{\alpha_l,t}$ from q^2 Storage Nodes

When there are bogus symbols $\tilde{p}'_{i,l,t}$ sent by the corrupted nodes for certain l, t, we can recover the matrices $S_{\alpha_l,t}, T_{\alpha_l,t}$ as follows:

Algorithm 10 DC reconstructs the original file in hostile network – Detection mode

Require: All the symbols requested.
Ensure: The original file in case no error is detected, otherwise, flag for errors.
1: **for** $l = 0 \to q-1$ **do**
2: Divide $\tilde{\mathbf{y}}'_{i,l}$ into A/α_l equal row vectors: $[\tilde{\mathbf{y}}'_{i,l,1}, \tilde{\mathbf{y}}'_{i,l,2}, \cdots, \tilde{\mathbf{y}}'_{i,l,A/\alpha_l}]$.
3: **for** $t = 1 \to A/\alpha_l$ **do**
4: Suppose $\tilde{\mathbf{y}}'_{i,l,t} = \tilde{\mathbf{y}}_{i,l,t} + \mathbf{e}_{i,l,t}$ is the response from the i^{th} storage node, without loss of generality, we assume $0 \le i \le k_l$. If symbol $\tilde{\mathbf{y}}_{i,l,t}$ has been modified, we have $\mathbf{e}_{i,l,t} \in (GF(q^2))^{\alpha_l} \backslash \{\mathbf{0}\}$. Let $R' = [\tilde{\mathbf{y}}'^T_{0,l,t}, \tilde{\mathbf{y}}'^T_{1,l,t}, \cdots, \tilde{\mathbf{y}}'^T_{k_l,l,t}]^T$.
5: Solve S_1, T_1 using symbols collected from node 0 to node $k_l - 1 = \alpha_l$. Define ${R_1}' = [\tilde{\mathbf{y}}'^T_{0,l,t}, \tilde{\mathbf{y}}'^T_{1,l,t}, \cdots, \tilde{\mathbf{y}}'^T_{\alpha_l,l,t}]^T$, then we have $\mathbf{V}_{0,\alpha_l,l} \cdot \begin{bmatrix} S_1 \\ T_1 \end{bmatrix} = {R_1}'$.
6: Solve S_2, T_2 using symbols collected from node 0 to node $k_l = \alpha_l + 1$ except node α_l. Define ${R_2}' = [\tilde{\mathbf{y}}'^T_{0,l,t}, \cdots, \tilde{\mathbf{y}}'^T_{\alpha_l-1,l,t}, \tilde{\mathbf{y}}'^T_{\alpha_l+1,l,t}]^T$ and $\Psi_{DC2} = \begin{bmatrix} \nu_{0,l} \\ \cdots \\ \nu_{\alpha_l-1,l} \\ \nu_{\alpha_l+1,l} \end{bmatrix}$, then we have $\Psi_{DC2} \cdot \begin{bmatrix} S_2 \\ T_2 \end{bmatrix} = {R_2}'$.
7: **if** $[S_1, T_1] = [S_2, T_2]$ **then**
8: $[S_{\alpha_l,t}, T_{\alpha_l,t}] \leftarrow [S_1, T_1]$.
9: **else**
10: Switch to recovery mode and exit the algorithm.
11: **end if**
12: **end for**
13: **end for**
14: Reconstruct the original file from S, T.

For R' in Algorithm 11, we have $\Psi_{DC} \cdot \begin{bmatrix} S' \\ T' \end{bmatrix} = R'$,

$$\Phi_{DC} S' \Phi_{DC}^T + \Delta_{DC} \Phi_{DC} T' \Phi_{DC}^T = R' \Phi_{DC}^T, \quad (25)$$

where $\Psi_{DC} = [\Phi_{DC}, \Delta_{DC} \cdot \Phi_{DC}]$, $\Phi_{DC} = \begin{bmatrix} \mu_0 \\ \mu_1 \\ \cdots \\ \mu_{q^2-1} \end{bmatrix}$ and μ_i represents $\mu_{i,l}$ which is the i^{th} row of the encoding matrix Φ_l in the proof of Theorem 1.

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

$$C + \Delta_{DC} D = \hat{R}'. \quad (26)$$

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

$$\begin{cases} C_{i,j} + \lambda_i \cdot D_{i,j} = \hat{R}'_{i,j} \\ C_{i,j} + \lambda_j \cdot D_{i,j} = \hat{R}'_{j,i} \end{cases}. \quad (27)$$

Because matrices C and D have the same structure, here we only focus on C (corresponding to S'). It is straightforward to

Algorithm 11 DC reconstructs the original file in hostile network – Recovery mode

Require: All the symbols requested.

Ensure: The node identities of the malicious nodes and the original file in case the errors can be corrected.

```
for l = q − 1 → 0 do
    Divide ỹ'_{i,l} into A/α_l equal row vectors: [ỹ'_{i,l,1}, ỹ'_{i,l,2}, ··· , ỹ'_{i,l,A/α_l}].
    for t = 1 → A/α_l do
        Suppose ỹ'_{i,l,t} is the response from the i^{th} storage node, without loss of generality, we assume 0 ≤ i ≤ q^2 − 1. Let R' = [ỹ'^T_{0,l,t}, ỹ'^T_{1,l,t}, ··· , ỹ'^T_{q^2−1,l,t}]^T.
        if # of corrupted nodes detected > min{q^2 − k_l, ⌊(q^2 − k_{q−1})/2⌋} then
            Flag the errors cannot be corrected and exit the algorithm.
        else
            Substitute ỹ'_{i,l,t} in R' with the symbol ⊗ representing an erasure vector if node i has been detected to be corrupted in the previous loops.
            Solve S', T' using the method in section VI-C. If symbols from node i are detected to be erroneous during the calculation, mark node i as corrupted.
            [S_{α_l,t}, T_{α_l,t}] ← [S', T'].
        end if
    end for
end for
Reconstruct the original file from S, T.
```

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 $\hat{R}'$. Furthermore, there will be errors in the i^{th} row and i^{th} column of C. Define $S'\Phi_{DC}^T = \hat{S}'$, we have

$$\Phi_{DC}\hat{S}' = C. \tag{28}$$

Here we can view each column of C as a $(q^2-1, \alpha_l, q^2-\alpha_l)$ MDS code because Φ_{DC} is a Vandermonde matrix. The length of the code is q^2-1 since the diagonal elements of C is unknown. Suppose node j is uncorrupted. If the number of erasures σ (corresponding to the previously detected corrupted nodes) and the number of the corrupted nodes τ that have not been detected satisfy:

$$\sigma + 2\tau + 1 \leq q^2 - \alpha_l, \tag{29}$$

then the j^{th} column of C can be recovered and the error locations (corresponding to the corrupted nodes) can be pinpointed. The non-diagonal elements of C can be recovered. So DC can reconstructs $S_{\alpha_l,t}$ using the method similar to [4]. For $T_{\alpha_l,t}$, the recovering process is similar.

VII. Performance Analysis

In this section, we will analyze the performance of the H-MSR code and compare it with the performance of the RS-MSR code.

A. Scalable Error Correction

1) Error correction for data regeneration: The RS-MSR code in [12] can correct up to τ errors by downloading symbols from $d+2\tau$ nodes. However, the number of errors may vary in the symbols sent by helper nodes. When there is no error or the number of errors is far less than τ, downloading symbols from extra nodes will be a waste of bandwidth. When the number of errors is larger than τ, the decoding process will fail without being detected. In this case, the symbols stored in the replacement node will be erroneous. If this erroneous node becomes a helper node later, the errors will propagate to other nodes.

The H-MSR code can detect the erroneous decodings using Algorithm 5. If no error is detected, the regeneration of H-MSR only needs to download symbols from one more node than the regeneration in the error-free network, while the extra cost for the RS-MSR code is 2τ. If errors are detected in the symbols received from the helper nodes, the H-MSR code can correct the errors using Algorithm 6. Moreover, the algorithm can determine whether the decoding is successful, while the RS-MSR code is unable to provide such information.

2) Error correction for data reconstruction: The evaluation result is similar to the data regeneration. The RS-MSR code can correct up to τ errors with support from 2τ additional helper nodes. The H-MSR code is more flexible. For error detection, it only requires symbols from one additional node using Algorithm 10. The errors can then be corrected using Algorithm 11. The algorithm can also determine whether the decoding is successful.

B. Error Correction Capability

For data regeneration described in Algorithm 6, H-MSR code can be viewed as q MDS codes with parameters (q^2-1, d_l, q^2-d_l), $l=0,\cdots,q-1$. Since $\alpha_l \leq \kappa(l)$ and $\kappa(l)$ is strictly decreasing, we can choose the sequence α_l to be strictly decreasing. So d_l is also strictly decreasing. For the q MDS codes, the minimum distance of the $(q^2-1, d_{q-1}, q^2-d_{q-1})$ code is the largest. In Algorithm 6, this code is decoded first and it can correct up to $\tau_{q-1} = \lfloor (q^2-d_{q-1}-1)/2 \rfloor$ errors, where $\lfloor x \rfloor$ is the floor function of x. Then the code (q^2-1, d_l, q^2-d_l), $l = q-2,\cdots,0$, will be decoded sequentially. The (q^2-1, d_l, q^2-d_l) code can correct at most $\tau_l = \tau_{q-1}$ errors when $q^2 - d_0 - 1 \geq \tau_{q-1}$. Thus, the total number of errors that the H-MSR code can correct is $\tau_{H-MSR} = q \cdot \tau_{q-1}$. While the $(q^3-q, \sum_{l=0}^{q-1} d_l, q^3-q-\sum_{l=0}^{q-1} d_l + 1)$ RS-MSR code with the same rate can correct $\tau_{RS-MSR} = \lfloor (q^3-q-\sum_{l=0}^{q-1} d_l)/2 \rfloor$ errors. Therefore, we have the following theorem.

Theorem 4. *For the data regeneration, the number of errors that the H-MSR code and the RS-MSR code can correct satisfy* $\tau_{H-MSR} > \tau_{RS-MSR}$ *when* $q \geq 3$.

Proof: For τ_{RS-MSR}, we have

$$\begin{aligned}\tau_{RS-MSR} &= \left\lfloor \left(q^3 - q - \sum_{l=0}^{q-1} d_l \right) /2 \right\rfloor \qquad (30)\\ &\leq \left\lfloor (q^3 - q - q \cdot d_{q-1} - \frac{q}{2}(q-1))/2 \right\rfloor \\ &= \left\lfloor q \cdot (q^2 - d_{q-1} - 1)/2 - \frac{q(q-1)}{4} \right\rfloor . \\ &\leq q \cdot (q^2 - d_{q-1} - 1)/2 - \frac{q(q-1)}{4}\end{aligned}$$

For τ_{H-MSR}, we have

$$\tau_{H-MSR} = q \cdot \lfloor (q^2 - d_{q-1} - 1)/2 \rfloor \qquad (31)$$

When $q = 3$, it is easy to verify that $\tau_{H-MSR} > \tau_{RS-MSR}$. When $q > 3$, We can rewrite equation (31) as

$$\tau_{H-MSR} \geq q \cdot (q^2 - d_{q-1} - 1)/2 - q/2. \qquad (32)$$

The gap between τ_{H-MSR} and τ_{RS-MSR} is at least

$$\frac{q(q-1)}{4} - \frac{q}{2} = \frac{q^2 - 3q}{4} > 0 \, , q > 3, \qquad (33)$$

so we have $\tau_{H-MSR} > \tau_{RS-MSR}$.

■

Example 1. *Suppose $q = 4$ and $m = 37$, the Hermitian curve is defined by $y^4 + y = x^5$ over $GF(4^2)$. From the previous discussion, we have $\kappa(0) = 10, \kappa(1) = 9, \kappa(2) = 7, \kappa(3) = 6$. Choose $\alpha_0 = 6, \alpha_1 = 5, \alpha_2 = 4, \alpha_3 = 3$. So $d_0 = 12, d_1 = 10, d_2 = 8, d_3 = 6$. According to the analysis above, we have $\tau_{H-MSR} = 4 \cdot \tau_3 = 4 \cdot \lfloor (15-6)/2 \rfloor = 16$, which is larger than $\tau_{RS-MSR} = \lfloor (60-36)/2 \rfloor = 12$.*

For the data reconstruction in Algorithm 11, H-MSR code can be seen as q MDS codes with parameters $(q^2 - 1, k_l - 1, q^2 - k_l + 1)$. The decoding for the reconstruction is performed from the code with the largest minimum distance to the code with the smallest minimum distance as in the data regeneration case. Similarly, we can conclude that the number of errors that can be corrected by the H-MSR code is larger than the RS-MSR code under the same code rate.

C. Complexity Discussion

For the complexity of the H-MSR code, we consider two scenarios.

1) H-MSR regeneration: For the H-MSR regeneration, compared with RS-MSR code, the H-MSR code will slightly increase the complexity of the helper nodes. For each helper node, the extra operation is a matrix multiplication between B_i^{-1} and Y_i. The complexity is $O(q^2) = \mathcal{O}((n^{1/3})^2) = \mathcal{O}(n^{2/3})$. Similar to [13], for a replacement node, from Algorithm 4 and Algorithm 5, we can derive that the complexity to regenerate symbols for RS-MSR is $\mathcal{O}(n^2)$, while the complexity for H-MSR is only $\mathcal{O}(n^{5/3})$. Likewise, for Algorithm 6, the complexity to recover the H-MSR code is $\mathcal{O}(n^{5/3})$, and $\mathcal{O}(n^2)$ for RS-MSR code.

2) H-MSR reconstruction: For the reconstruction, compared with RS-MSR code, the additional complexity of the H-MSR code for each storage node is $\mathcal{O}(q^2)$, which is $\mathcal{O}(n^{2/3})$. The computational complexity for DC to reconstruct the data is $\mathcal{O}(n^{5/3})$ for the H-MSR code and $\mathcal{O}(n^2)$ for the RS-MSR code.

VIII. Conclusion

In this paper, we developed a Hermitian code based minimum storage regeneration (H-MSR) code for distributed cloud storage. Due to the structure of Hermitian code, our proposed H-MSR code can significantly improve the performance of the regenerating code under malicious attacks. In particular, H-MSR code can deal with errors beyond the maximum distance separable (MDS) code. Our theoretical analysis demonstrated that the H-MSR has a lower complexity than the Reed-Solomon based minimum storage regeneration (RS-MSR) code in both regeneration and reconstruction. As a future research task, we will further analyze the optimal design of regenerating code based on the Hermitian-like codes.

References

[1] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz, "Maintenance-free global data storage," *IEEE Internet Computing*, vol. 5, pp. 40 – 49, 2001.

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

[3] 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.

[4] 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.

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

[6] 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.

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

[8] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *IEEE International Symposium on Information Theory, 2009. ISIT 2009.*, pp. 2276–2280, 2009.

[9] 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.

[10] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Transactions on Information Theory*, vol. 57, pp. 6734 – 6753, 2011.

[11] 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.

[12] 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.

[13] J. Ren, "On the structure of hermitian codes and decoding for burst errors," *IEEE Transactions on Information Theory*, vol. 50, pp. 2850–2854, 2004.