Beyond the MDS Bound in Distributed Cloud Storage

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 with the amount of data stored per node to repair a failed node. There are two extreme points in the optimal regenerating trade-off curve, which correspond to minimum-storage regenerating (MSR) and minimum-bandwidth regenerating (MBR). Recently, Reed-Solomon (RS) code based regenerating codes (RS-RC) were constructed under the product-matrix framework. It can also achieve the maximum distance separable (MDS) property in code regeneration and reconstruction. However, in case that the network is hostile and the storage nodes could be compromised or packets be modified, the storage capacity and the bandwidth required to regenerate or reconstruct the original file can be significantly affected. In this paper, we propose Hermitian code based regenerating codes (H-RC) by developing constructions under the product-matrix framework for minimum storage regenerating (H-MSR) and the minimum bandwidth regenerating (H-MBR). We also propose data regeneration and reconstruction algorithms for both H-MSR and H-MBR codes under both error-free and hostile networks. We demonstrate that the proposed algorithms can also successfully determine the erroneous decodings in hostile networks. Theoretical evaluation shows that our proposed H-RC can detect and correct more errors in hostile networks well beyond the RS-RC with the same code rate. Our analysis shows that the proposed H-RC have lower computational complexity than the RS-RC for 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 data across multiple servers or clouds, often in a replicated fashion. However, data replication lacks flexibility in data recovery, and requires costly secure data management when content confidentiality is needed. Moreover, it is vulnerable to data sabotage attacks.

Manuscript received December 21, 2017; revised May 25, 2019; accepted May 4, 2020. Date of publication May 7, 2020; date of current version June 18, 2020. This work was supported in part by NSF under Grant CCF-1919154 and Grant ECCS-1923409. This article was presented in part at the 2014 IEEE International Conference on Computer Communications. *(Corresponding author: Jian Ren.)*

Jian Li was with the Department of ECE, Michigan State University, East Lansing, MI 48824-1226 USA. He is now with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: lijian@bjtu.edu.cn).

Tongtong Li and Jian Ren are with the Department of ECE, Michigan State University, East Lansing, MI 48824-1226 USA (e-mail: tongli@msu.edu; renjian@msu.edu).

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

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

Digital Object Identifier 10.1109/TIT.2020.2993038

Distributed storage is an alternative approach to data replication, which can achieve an elegant tradeoff between the costly secure data management and the cheap storage media. The main idea is that: instead of storing the entire data in one server, the data to be stored is split into n components and stored across up to n servers. The original data can be recovered only when the required number of components, say k, are collected. In this way, the stored data is information theoretically secure for anyone who can access up to $k-1$ data components. When individual components are stored distributively across multiple cloud storage servers, each cloud storage server only needs to ensure data integrity and availability services. The costly data encryption and secure key management are no longer needed. Moreover, distributed storage can also increase data availability while reducing network congestion, leading to increased resiliency.

A. Related Work

When a storage node in the distributed storage network that employs the conventional (n,k) RS code (such as OceanStore [1] and Total Recall [2]) fails, the replacement node first connects to k nodes and recovers the whole file, then regenerates 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 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 β help symbols from each of d helper nodes. Therefore, the total bandwidth required to regenerate a failed node is $\gamma = d\beta$. γ could be far less than the whole file B. A data collector (DC) can reconstruct the original file stored in the network by downloading α symbols from each of k storage nodes. In [3], the authors proved that there is a trade-off between bandwidth γ and per node storage α. They found two optimal points in the optimal tradeoff curve: minimum storage regeneration (MSR) and minimum bandwidth regeneration (MBR) points. The existing work has largely focused on the optimal regenerating codes design [4]–[8], and implementation of the regenerating code.

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 a failed component instead of being the same as the originally stored component. In [9], the data regeneration was formulated as a multicast network

0018-9448 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

coding problem. The paper also constructed functional regenerating code. A random linear regenerating codes for distributed storage systems was implemented in [10]. 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 [11]. In [12], the authors 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. In [13], the authors proposed to reduce the regeneration bandwidth through algebraic alignment. A code structure for exact regeneration using interference alignment technique was provided in [14]. In [15], RS code based MSR (RS-MSR) code and MBR (RS-MBR) code were explicitly constructed. It was the first work that allows independent selection of the node number n in the network. It also presented optimal exact regeneration methods for the MBR and MSR codes under the product-matrix framework. In [16], repair performance of the RS 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 [17].

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

For general verification of the contents stored in distributed storage, many schemes were proposed. In [18], the verification cost for both the client read and write operations in workloads with idle periods was analyzed. In [19], the authors proposed to use erasure-coding and threshold cryptography to achieve storage efficiency and resilience. To check data integrity of the regenerating code in hostile networks, CRC code was adopted in [20]. Unfortunately, the CRC checks can be easily manipulated by the malicious nodes, resulting in failure of regeneration and reconstruction. In [21], data integrity protection (DIP) was designed under a mobile Byzantine adversarial model to enable a client to verify the integrity of outsourced data against general or malicious corruptions in distributed storage.

For corrupted node detection and correction in regenerating codes, the following studies were conducted. In [22], 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 [23]. In [24], error resilience of the RS code based regenerating code in the network with both errors and erasures was evaluated. The paper provided the theoretical error correction capability. In [25], the authors discussed the optimal trade-off between the storage space and the repair bandwidth in presence of two types of wiretapper. In [26], the authors revealed some general properties of MSR codes and a generally applicable upper bound on secrecy capacity with passive eavesdroppers in the storage network. A secure MSR coding scheme that could overcome the limitations of previous eavesdropper model was proposed in [27]. The achievable trade-off regions between the normalized storage capacity and repair bandwidth for the secure exact-repair regenerating codes against an eavesdropper were studied in [28].

Nevertheless, since regenerating codes in these works are all an extension of the maximum distance separable (MDS) code, the error correction capability is constrained by the MDS bound. Moreover, none of the schemes presented is able to determine whether the errors in network are successfully corrected.

B. Our Contributions

In this paper, we develop Hermitian code based regeneration codes (H-RC) under the product-matrix framework [15]. Compared with the existing RS code based regeneration codes (RS-RC), the proposed H-RC can correct more errors and can always determine whether the error correction is successful. The performance breakthrough in error correction is achieved through our discovery that the Hermitian code is composed of the direct sum of multiple layers of RS codes [29], with parameters of each layer determined by the Hermitian curve. More specifically, the major contributions of this paper can be summarized as follows:

1) We develop a novel construction of H-RC at the MSR (H-MSR) and the MBR (H-MBR) points. The design is based on our discovery that each Hermitian code can be decomposed into multiple RS codes. Therefore, we can construct the H-MSR and the H-MBR codes by concurrent processing of multiple layers of RS codes.
2) We propose data regeneration and reconstruction algorithms for the H-MSR and the H-MBR codes in both error-free and hostile networks. Our construction is fundamentally different from the general network communication based data reconstruction due in part by our focus on the capability in regeneration and reconstruction of the corrupted code components.
3) We design efficient algorithms that can achieve significant performance improvement in error correction over the RS-RC. More specifically, the error correction capability of the RS-RC is *well beyond* the MDS bound due to the structure of the underlying Hermitian code.
4) We provide theoretical proof that the proposed algorithms can successfully detect erroneous decodings in hostile networks.

C. Organization

The rest of this paper is organized as follows: The preliminary of this paper is presented in Section II. In Section III, our proposed encoding of H-MSR code is described. In Section IV, regeneration of the H-MSR code is discussed. Reconstruction of the H-MSR code is analyzed in Section V. In Section VI, our proposed encoding of the H-MBR code is described.

TABLE I
q^3 RATIONAL POINTS OF THE HERMITIAN CURVE

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

In Section VII, regeneration of the H-MBR code is discussed. Reconstruction of the H-MBR code is analyzed in Section VIII. We conduct performance analysis in Section IX. The paper is concluded in Section X.

II. PRELIMINARY AND ASSUMPTIONS

A. Regenerating Code

Regenerating code was first introduced in [3]. It is a linear code over the finite field $\mathbb{F}_q$ with a set of parameters $\{n,k,d,\alpha,\beta,B\}$, where q is a prime number or some power of a prime number. 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 [3], 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. There are two special cases: the minimum storage regeneration (MSR) point, where the storage parameter α is minimized:

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

and the minimum bandwidth regeneration (MBR) point, where the regeneration 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. Regenerating Code Construction Under the Product-Matrix Framework

In [15], the authors proposed the product-matrix framework and constructed the RS-MSR and RS-MBR codes. Under the product-matrix framework, data encoding, regeneration and reconstruction are carried out in the form of matrices. Let V be a data matrix with the size $d\times\alpha$, Ψ be an $n\times d$ encoding matrix, and W be the corresponding codeword matrix with the size $n\times\alpha$. The encoding of the codeword matrix can be written as $W=\Psi V$. Then each row of the codeword matrix $W_i=\Psi_i V$ $(0\le i\le n-1)$ is stored in one of the storage nodes, where Ψ_i is the i^{th} row of Ψ. For data regeneration, the replacement node will request help symbols p_i $(0\le i\le d-1)$ from d storage nodes, where $p_i=W_i\Phi_z^T$, z is the failed node and Φ_z is the vector comprised of the first α elements in the encoding vector Ψ_z. Upon receiving all the requested help symbols, the replacement node can regenerate the symbols in the failed node. For data reconstruction, the data collecter will request symbols W_i $(0\le i\le k-1)$ stored in k storage nodes and reconstruct the original data.

At the MSR point, the RS-MSR code was constructed for the parameters $d=2k-2$, $\alpha=k-1$, $\beta=1$ and $B=\alpha(\alpha+1)$. The message matrix V is defined as $V=\begin{bmatrix}S\\T\end{bmatrix}$, where S,T are symmetric matrices with the upper-triangular part filled by data symbols. The encoding matrix Ψ is defined as $\Psi=[\Phi,\Lambda\Phi]$, where Φ is a Vandermonde matrix and Λ is a diagonal matrix with all the elements λ_i $(0\le i\le n-1)$ different from each other. To regenerate the symbols stored in a failed node z, the replacement node collects d help symbols from d helper nodes and obtains the following vector: $\Psi_{repair}V\Phi_z^T$, where $\Psi_{repair}=\left[\Psi_0^T,\Psi_1^T,\dots,\Psi_{d-1}^T\right]^T$ is a $d\times d$ invertible matrix. Then the replacement node can calculate the vector $(V\Phi_z^T)^T=\Phi_z V^T=\Phi_z[S,T]$ and regenerate the symbols through $W_z=\Phi_z S+\lambda_z\Phi_z T=\Psi_z V$. A data reconstruction algorithm to reconstuct the original file is also proposed. The details of the data reconstruction can be found in [15].

At the MBR point, the RS-MBR code was constructed for the parameters $\alpha=d$, $\beta=1$ and $B=k(2d-k+1)/2$. The message matrix V is defined as $V=\begin{bmatrix}S & T\\T^T & 0\end{bmatrix}$, where S is a $k\times k$ symmetic matrix with the upper-triangular part filled by data symbols, and T is a $k\times(d-k)$ matrix filled by data symbols. The encoding matrix Ψ is a Vandermonde matrix. To regenerate the symbols stored in a failed node z, the replacement node collects d help symbols from d helper nodes and obtains the following vector: $\Psi_{repair}V\Psi_z^T$, where $\Psi_{repair}=\left[\Psi_0^T,\Psi_1^T,\dots,\Psi_{d-1}^T\right]^T$ is a $d\times d$ invertible matrix. Then the replacement node can calculate the vector $(V\Psi_z^T)^T=\Psi_z V=W_z$, which consists of the exact symbols in the failed node z. For data reconstruction, [15] also proposed an algorithm to reconstruct the original data.

C. Hermitian Code

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

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

There are q^3 points that satisfy equation (4), denoted as $P_{0,0},\dots,P_{0,q-1},\dots,P_{q^2-1,0},\dots,P_{q^2-1,q-1}$ (See Table I),

where $\theta_0, \theta_1, \ldots, \theta_{q-1}$ are the q solutions to $y^q + y = 0$ and ϕ is a primitive element in $\mathbb{F}_{q^2}$. Define

$$L(mQ) = \{f_0(x) + yf_1(x) + \cdots + y^{q-1}f_{q-1}(x) \mid \deg f_l(x) < \kappa(l), l = 0, 1, \ldots, q-1\}, \quad (5)$$

where

$$\kappa(l) = \max\{t \mid tq + l(q+1) \leq m\} + 1, \quad (6)$$

$m \geq q^2 - 1$. A codeword of the Hermitian code [29] $\mathcal{H}_m$ is defined as

$$\left(\varrho(P_{0,0}), \ldots, \varrho(P_{0,q-1}), \ldots, \varrho(P_{q^2-1,0}), \ldots, \varrho(P_{q^2-1,q-1})\right), \quad (7)$$

where $\varrho \in L(mQ)$. The dimension of the message can be calculated as $\dim(\mathcal{H}_m) = \sum_{l=0}^{q-1}(\deg f_l(x) + 1)$.

According to the decoding algorithm proposed in [29], $\mathcal{H}_m$ could be viewed as the concatenation of q extended RS codes $(q^2, \kappa(l))$, $l = 0, 1, \ldots, q-1$. This inspires us to incorporate q MSR codes into the H-MSR code construction. For more information of the Hermitian code, please refer to [29].

D. Adversarial Model

In this paper, our adversarial model is similar as the one adopted in [24]. We assume some network nodes may be corrupted due to hardware failure or communication errors, and/or be controlled by malicious users. As a result, upon request, these nodes may provide incorrect responses to disrupt data regeneration and reconstruction. These incorrect responses are described as errors/erasures in [24]. We assume that the malicious users can take full control of up to τ ($\tau \leq n$) storage nodes and perform possible collusion attacks. Other than information stored in the compromised nodes, the adversary is unable to obtain the contents or the distribution of the encoding vectors in other intact nodes.

We will refer the corrupted and the compromised symbols as *bogus* symbols. We will also use corrupted nodes, malicious nodes and compromised nodes interchangeably throughout the paper.

III. Encoding of the H-MSR Code

Motivated by [15], in this section we construct H-MSR code with $d = 2k - 2 = 2\alpha$ under the similar product-matrix framework. The code with $d > 2k - 2$ can be derived the same way through truncating operations. As we have discussed earlier, Hermitian code can be decomposed into q RS codes. Therefore, our construction consists of concurrent processing of q RS-MSR codes. We also present an encoding example at the end of this section.

Let $\alpha_0, \ldots, \alpha_{q-1}$ be a strictly decreasing integer sequence satisfying $0 < \alpha_l \leq \kappa(l)$, $0 \leq l \leq q-1$, where α_l corresponds to the parameter α for the underlying regenerating code. The least common multiple of $\alpha_0, \ldots, \alpha_{q-1}$ is A. Suppose the data contains $B = A\sum_{l=0}^{q-1}(\alpha_l + 1)$ message symbols from the finite field $\mathbb{F}_{q^2}$. In practice, if the size of the actual data is larger than B symbols, we can fragment it into blocks of size B and process each block individually.

TABLE II
Notations

α_l, $0 \leq l \leq q-1$	strictly decreasing regeneration parameter α for the l^{th} underlying RS-MSR code of the H-MSR code, or the regeneration parameter α for the l^{th} underlying RS-MBR code of the H-MBR code
d_l, k_l, $0 \leq l \leq q-1$	the regeneration parameters d, k for the l^{th} underlying RS-MSR code of the H-MSR code, or the regeneration parameters d, k for the l^{th} underlying RS-MBR code of the H-MBR code
A	least common multiple of $\alpha_0, \ldots, \alpha_{q-1}$
S, T	data matrices for the H-MSR code
S_l, T_l	data matrices for the l^{th} underlying RS-MSR code of the H-MSR code
Φ_l	$q^2 \times \alpha_l$ Vandermonde matrix, can be viewed as the encoding matrix for the l^{th} underlying RS-MBR code of the H-MBR code
$\Phi_{i,l}$	i^{th} row of the matrix Φ_l
$\Phi_{i\to j,l}$	From the i^{th} to the j^{th} rows of the matrix Φ_l
Ψ_l	$[\Phi_l, \Delta\Phi_l]$
$\Psi_{i,l}$	i^{th} row of $\Psi_l = [\Phi_l, \Delta\Phi_l]$
$\Psi_{i\to j,l}$	From the i^{th} to the j^{th} rows of the matrix Ψ_l
$P_{i,l}$	point on the Hermitian curve
$\varrho(P_{i,l})$	Hermitian codeword for the point $P_{i,l}$
B_i	Hermitian coefficient matrix for node i
$\mathbf{f}_i$	vectors related to data matrix S, stored in node i
$\mathbf{g}_i$	vectors related to data matrix T, stored in node i
M	data matrix for the H-MBR code
M_l	data matrix for the l^{th} underlying RS-MBR code of the H-MBR code
$\mathbf{u}_i$	vectors related to data matrix M, stored in node i
Y	H-MSR/H-MBR codeword matrix
Y_i	H-MSR/H-MBR codeword submatrix, stored in node i
$\widetilde{Y}_i$	codeword matrix with the coefficient matrix B_i removed in node i
$\widetilde{\mathbf{Y}}_{i,l}$	l^{th} row of $\widetilde{Y}_i$
$\tilde{\mathbf{y}}_{i,l,t}$	sub-row vectors of $\widetilde{\mathbf{Y}}_{i,l}$ with the size $1 \times \alpha_l$
$\tilde{p}_{i,l,t}$	help symbols of H-MSR/H-MBR code sent from node i in error-free networks during regeneration
$\tilde{p}'_{i,l,t}$	help symbols of H-MSR/H-MBR code sent from node i in hostile networks during regeneration

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

$$S = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{q-1} \end{bmatrix}, \quad T = \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{q-1} \end{bmatrix}, \quad (8)$$

where

$$S_l = [S_{l,1}, S_{l,2}, \ldots, S_{l,A/\alpha_l}], \quad T_l = [T_{l,1}, T_{l,2}, \ldots, T_{l,A/\alpha_l}]. \quad (9)$$

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

$T_{l,j}$ $(0 \leq l \leq q-1, 1 \leq j \leq A/\alpha_l)$ is constructed in the same way as $S_{l,j}$. So T contains the other $\frac{A}{2}\sum_{l=0}^{q-1}(\alpha_l+1)$ data symbols.

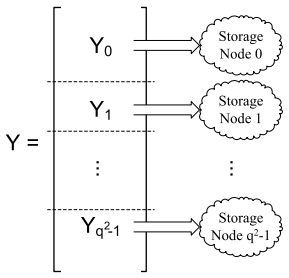

Fig. 1. Illustration of storing the codeword matrices in distributed storage nodes.

Definition 1: For a Hermitian code $\mathcal{H}_m$ over $\mathbb{F}_{q^2}$, we encode matrix $M_{\dim(\mathcal{H}_m)\times A} = [M_1, M_2 \ldots, M_A]$ by encoding each column M_i, $i = 1, 2, \ldots, A$, individually using $\mathcal{H}_m$. The codeword matrix is defined as

$$\mathcal{H}_m(M) = [\mathcal{H}_m(M_1), \mathcal{H}_m(M_2), \ldots, \mathcal{H}_m(M_A)], \tag{10}$$

where $\mathcal{H}_m(M_i)$ has the following form ($\varrho \in L(mQ)$):

$$[\varrho(P_{0,0}), \ldots, \varrho(P_{0,q-1}), \ldots, \varrho(P_{q^2-1,0}), \ldots, \varrho(P_{q^2-1,q-1})]^T. \tag{11}$$

The elements of M_i are viewed as the coefficients of the polynomials $f_0(x), \ldots, f_{q-1}(x)$ in ϱ when M_i is encoded.

Let

$$\Phi_l = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & 1 & \ldots & 1 \\ 1 & \phi & \phi^2 & \ldots & \phi^{\alpha_l-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \phi^{q^2-2} & (\phi^{q^2-2})^2 & \ldots & (\phi^{q^2-2})^{\alpha_l-1} \end{bmatrix} \tag{12}$$

be a Vandermonde matrix, where ϕ is the primitive element in $\mathbb{F}_{q^2}$ defined in section II-C and $0 \le l \le q-1$.

Let $\Delta = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{q^2-1})$ be a diagonal matrix, where λ_i $(0 \le i \le q^2-1)$ is chosen using the following two criteria: (i) All λ_i's are different. (ii) Any $d_l = 2\alpha_l$ rows of the matrix $[\Phi_l, \Delta\Phi_l]$ $(0 \le l \le q-1)$ are linearly independent.

For the $q \times q$ identity matrix I, define

$$\Lambda_i = \lambda_i I \ (0 \le i \le q^2-1) \text{ and } \Gamma = \text{diag}(\Lambda_0, \Lambda_1, \ldots, \Lambda_{q^2-1}). \tag{13}$$

For distributed storage, we encode each pair of matrices (S, T) using the encoding of ***H-MSR code*** algorithm defined below.

Algorithm 1: Encoding of H-MSR code

Step 1: Encode the data matrices S, T defined in equation (8) using a Hermitian code $\mathcal{H}_m$ over $\mathbb{F}_{q^2}$ with parameters $\kappa(l)$ $(0 \le l \le q-1)$ and m $(m \ge q^2-1)$. Denote the generated $q^3 \times A$ codeword matrices as $\mathcal{H}_m(S)$ and $\mathcal{H}_m(T)$.

Step 2: Compute the $q^3 \times A$ codeword matrix $Y = \mathcal{H}_m(S) + \Gamma\mathcal{H}_m(T)$.

Step 3: Divide Y into q^2 submatrices $Y_0, \ldots, Y_{q^2-1}$ of size $q \times A$ and store each submatrix in a storage node as shown in Fig. 1.

For encoding of H-MSR code, we have the following theorem.

Theorem 1: The encoding of H-MSR code described in Algorithm 1 can achieve the MSR point in distributed storage.

Proof: The proof will be divided into three steps.

First, we study the structure of the codeword matrix $\mathcal{H}_m(S)$. Since every column of the matrix is an independent Hermitian codeword, without loss of generality we will decode the first column $\mathbf{h} = [h_{0,0}, \ldots, h_{0,q-1}, \ldots, h_{q^2-1,0}, \ldots, h_{q^2-1,q-1}]^T$ as an example. Arrange the q^3 rational points of the Hermitian curve following the order in Table I, we can find that for each i, $i = 0, 1, \ldots, q^2-1$, the rational points $P_{i,0}, P_{i,1}, \ldots, P_{i,q-1}$ all have the same first coordinate.

Suppose $\varrho \in L(mQ)$, $\varrho(P_{i,l}) = f_0(P_{i,l}) + y(P_{i,l})f_1(P_{i,l}) + \cdots + (y(P_{i,l}))^{q-1}f_{q-1}(P_{i,l})$, $0 \le i \le q^2-1$, $0 \le l \le q-1, \deg f_l(x) = \alpha_l - 1$. Since $P_{i,0}, P_{i,1}, \ldots, P_{i,q-1}$ 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 $f_j(P_{i,l}) = f_j(\phi^{s_i})$, meaning $f_j(P_{i,l})$ does not depend on l, where $s_0 = -\infty, s_i = i-1$ for $i \ge 1$, and $\phi^{-\infty} = 0$. Therefore, we can derive q^2 sets of equations for $0 \le i \le q^2-1$:

$$\begin{cases} f_0(\phi^{s_i}) + y(P_{i,0})f_1(\phi^{s_i}) + \ldots + [y(P_{i,0})]^{q-1}f_{q-1}(\phi^{s_i}) = h_{i,0} \\ f_0(\phi^{s_i}) + y(P_{i,1})f_1(\phi^{s_i}) + \ldots + [y(P_{i,1})]^{q-1}f_{q-1}(\phi^{s_i}) = h_{i,1} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ f_0(\phi^{s_i}) + y(P_{i,q-1})f_1(\phi^{s_i}) + \ldots + [y(P_{i,q-1})]^{q-1}f_{q-1}(\phi^{s_i}) = h_{i,q-1}. \end{cases} \tag{14}$$

If we store the codeword matrix in storage nodes according to Fig. 1, then each set of the equations corresponds to a storage node. As we mentioned above, the set of equations in equation (14) can be derived from storage node i. If we treat $y(P_{i,j})$'s as coefficients, then the coefficient matrix B_i in equation (14) is a Vandermonde matrix:

$$B_i = \begin{bmatrix} 1 & y(P_{i,0}) & \ldots & y(P_{i,0})^{q-1} \\ 1 & y(P_{i,1}) & \ldots & y(P_{i,1})^{q-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & y(P_{i,q-1}) & \ldots & y(P_{i,q-1})^{q-1} \end{bmatrix}. \tag{15}$$

Let $\mathbf{h}_i = [h_{i,0}, h_{i,1}, \ldots, h_{i,q-1}]^T$. Then the solution $\mathbf{f}_i = [f_0(\phi^{s_i}), f_1(\phi^{s_i}), \ldots, f_{q-1}(\phi^{s_i})]^T$ can be expressed as:

$$\mathbf{f}_i = B_i^{-1}\mathbf{h}_i. \tag{16}$$

From all the q^2 storage nodes, we can get vectors $\mathcal{F}_l = [f_l(0), f_l(1), \ldots, f_l(\phi^{q^2-2})]^T$, $l = 0, \ldots, q-1$, which can be viewed as extended RS codes.

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

$$\Phi_l S_{l,j} = F_{l,j}, \tag{17}$$

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

Second, 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 $\Gamma\mathcal{H}_m(T)$, we can derive

$$\Delta\Phi_l T_{l,j} = \Delta G_{l,j}, \tag{18}$$

where $\mathcal{G}_l = [g_l(0), g_l(1), \ldots, g_l(\phi^{q^2-2})]^T$, g_l is a polynomial defined the same as f_l with coefficients from the columns of

the data matrix T, $G_{l,j} = [\mathcal{G}_l^{(1)}, \ldots, \mathcal{G}_l^{(\alpha_l)}], 0 \le l \le q-1, 1 \le j \le A/\alpha_l$, and $\mathcal{G}_l^{(t)}$ corresponds to the t^{th} column of the submatrix $T_{l,j}$.

Third, we will study the optimality of the code in the sense of the MSR point. For $\Phi_l S_{l,j} + \Delta\Phi_l T_{l,j}$, $0 \le l \le q-1, 1 \le j \le A/\alpha_l$, since $S_{l,j}, T_{l,j}$ are symmetric and satisfy the requirements for MSR point according to [15] with parameters $d = 2\alpha_l, k = \alpha_l + 1, \alpha = \alpha_l, \beta = 1, B = \alpha_l(\alpha_l+1)$. By encoding S, T using $\mathcal{H}_m(S) + \Gamma\mathcal{H}_m(T)$ and distributing $Y_0, \ldots, Y_{q^2-1}$ into q^2 storage nodes, each row of the matrix $\Phi_l S_{l,j} + \Delta\Phi_l T_{l,j}$, $0 \le l \le q-1$, $1 \le j \le A/\alpha_l$, can be derived in a corresponding storage node. Because $\Phi_l S_{l,j} + \Delta\Phi_l T_{l,j}$ achieves the MSR point, data related to matrices $S_{l,j}, T_{l,j}$, $0 \le l \le q-1$, $1 \le j \le A/\alpha_l$, can be regenerated at the MSR point. Therefore, Algorithm 1 can achieve the MSR point. ∎

According to [29], the Hermitian code $\mathcal{H}_m$ could be viewed as the concatenation of q RS codes. For H-MSR code, we have a similar result:

Theorem 2: The H-MSR code could be viewed as the concatenation of q RS-MSR codes. Each storage node contains symbols related to each of the q RS-MSR codes encoded in Algorithm 1.

Proof: (sketch) According to equation (17) and (18) in the proof of Theorem 1, the H-MSR code can be viewed as q concatenated RS-MSR codes, where the encoding matrix of the l^{th} RS-MSR code, $0 \le l \le q-1$, is

$$\Psi_l = [\Phi_l, \Delta\Phi_l]. \tag{19}$$

Similar to equation (16) in the proof of Theorem 1, for any column $\mathrm{Col}(Y_i)$ of the submatrix Y_i in storage node i, we have the following equation:

$$\begin{aligned}\mathbf{f}_i + \Lambda_i \mathbf{g}_i &= B_i^{-1}\mathrm{Col}(Y_i)\\ &= [f_0(\phi^{s_i}) + \lambda_i g_0(\phi^{s_i}), f_1(\phi^{s_i}) + \lambda_i g_1(\phi^{s_i}),\\ &\quad \ldots, f_{q-1}(\phi^{s_i}) + \lambda_i g_{q-1}(\phi^{s_i})]^T,\end{aligned} \tag{20}$$

where $\mathbf{g}_i = [g_0(\phi^{s_i}), g_1(\phi^{s_i}), \ldots, g_{q-1}(\phi^{s_i})]^T$, with $f_l(\phi^{s_i}) + \lambda_i g_l(\phi^{s_i})$ corresponding to the l^{th} RS-MSR code, $0 \le l \le q-1$. The codewords related to all of the q RS-MSR codes are stored in each of the storage node i. ∎

In [15], construction of the RS-MSR code under the product-matrix framework was proposed along with data regeneration and reconstruction algorithms. While Theorem 1 provides theoretic construction and efficient algorithms of the H-MSR code, Theorem 2 presents the relationship between H-MSR and RS-MSR codes, which makes the construction of the H-MSR code under the product-matrix framework more intuitive.

Example 1: Fig. 2 is an illustrative example with parameters $q = 3, \alpha_0 = 4, \alpha_1 = 3, \alpha_2 = 2$. For this setting, the total number of symbols B to be stored is $\mathrm{LCM}(\alpha_0, \alpha_1, \alpha_2) \sum_{i=0}^{q-1}(\alpha_i + 1) = 12 \times (5+4+3) = 144$, where LCM stands for least common multiple. Data matrix S stores the first 72 symbols. Data matrix T has the same structure and stores data symbols from b_{72} to b_{143}. When we encode S, T using Algorithm 1, each column of the matrix S and T will be encoded first by the

$$\left[\begin{array}{cccc|cccc|cccc}
b_0 & b_1 & b_2 & b_3 & b_{10} & b_{11} & b_{12} & b_{13} & b_{20} & b_{21} & b_{22} & b_{23}\\
b_1 & b_4 & b_5 & b_6 & b_{11} & b_{14} & b_{15} & b_{16} & b_{21} & b_{24} & b_{25} & b_{26}\\
b_2 & b_5 & b_7 & b_8 & b_{12} & b_{15} & b_{17} & b_{18} & b_{22} & b_{25} & b_{27} & b_{28}\\
b_3 & b_6 & b_8 & b_9 & b_{13} & b_{16} & b_{18} & b_{19} & b_{23} & b_{26} & b_{28} & b_{29}\\
\hline
b_{30} & b_{31} & b_{32} & b_{36} & b_{37} & b_{38} & b_{42} & b_{43} & b_{44} & b_{48} & b_{49} & b_{50}\\
b_{31} & b_{33} & b_{34} & b_{37} & b_{39} & b_{40} & b_{43} & b_{45} & b_{46} & b_{49} & b_{51} & b_{52}\\
b_{32} & b_{34} & b_{35} & b_{38} & b_{40} & b_{41} & b_{44} & b_{46} & b_{47} & b_{50} & b_{52} & b_{53}\\
\hline
b_{54} & b_{55} & b_{57} & b_{58} & b_{60} & b_{61} & b_{63} & b_{64} & b_{66} & b_{67} & b_{69} & b_{70}\\
b_{55} & b_{56} & b_{58} & b_{59} & b_{61} & b_{62} & b_{64} & b_{65} & b_{67} & b_{68} & b_{70} & b_{71}
\end{array}\right]$$

Fig. 2. An illustrative example of matrix S.

Hermitian code, resulting in the $q^3 \times A = 27 \times 12$ codeword matrix Y. Then the 3×12 submatrices $Y_0, \ldots, Y_8$ will be stored in 9 storage nodes. As we have elaborated in the proof of Theorem 1, the 4×4 submatrices with solid line brackets in Fig. 2 corresponds to the MSR code with $\alpha_0 = 4$, the 3×3 submatrices with long dash line brackets corresponds to the MSR code with $\alpha_1 = 3$ and the 2×2 submatrices with short dash line brackets corresponds to the MSR code with $\alpha_2 = 2$.

IV. Regeneration of the H-MSR Code

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

A. Regeneration in Error-Free Networks

The main idea of the regeneration algorithms is to regenerate the l^{th} RS-MSR code described in the proof of Theorem 2 by downloading help symbols from $d_l = 2\alpha_l$ nodes, where d_l $(0 \le l \le q-1)$ represents the regeneration parameter d for the l^{th} RS-MSR code.

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

$$\Psi_{i\to j,l} = \begin{bmatrix} \Psi_{i,l} \\ \Psi_{i+1,l} \\ \vdots \\ \Psi_{j,l} \end{bmatrix}, \tag{21}$$

where $\Psi_{t,l}$, $i \le t \le j$, is the t^{th} row of Ψ_l. Each node i, $0 \le i \le q^2-1$, only stores its own encoding vector $\Psi_{i,l}$, $0 \le l \le q-1$. Replacement node z' sends integer j from $q-1$ to 0 in descending order to $d_j - d_{j+1}$ helper nodes that it has not requested before. Upon receiving the integer j, helper node i calculates $\widetilde{Y}_i = B_i^{-1} Y_i$, which eliminates the coefficient matrix B_i from the codeword matrix. Since the l^{th} row of $\widetilde{Y}_i$ corresponds to the symbols related to the l^{th} RS-MSR code $(0 \le l \le j)$, node i can divide the l^{th} row of $\widetilde{Y}_i$ into $1 \times \alpha_l$ row vectors $\tilde{\mathbf{y}}_{i,l,t}$ $(1 \le t \le A/\alpha_l)$. Define $\widetilde{\mathbf{Y}}_{i,l} = [\tilde{\mathbf{y}}_{i,l,1}, \tilde{\mathbf{y}}_{i,l,2}, \ldots, \tilde{\mathbf{y}}_{i,l,A/\alpha_l}]$. Then for every $0 \le l \le j$ and $1 \le t \le A/\alpha_l$, node i can calculate the help symbol $\tilde{p}_{i,l,t} = \tilde{\mathbf{y}}_{i,l,t}\Phi_{z,l}^T$, where $\Phi_{z,l}$ is the z^{th} row of the encoding matrix Φ_l defined in equation (12).

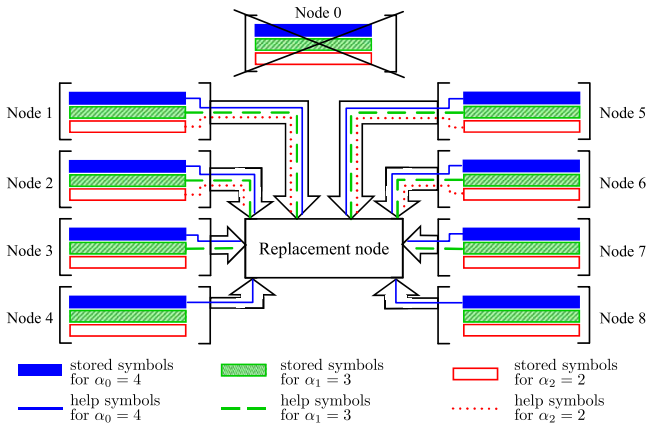

Fig. 3. An illustrative example of the H-MSR data regeneration.

Since $d_{l_1} > d_{l_2}$ for $l_1 < l_2$, for efficiency consideration, only d_{q-1} helper nodes need to provide symbols $\tilde{p}_{i,l,t}$ ($0 \leq l \leq q-1$ and $1 \leq t \leq A/\alpha_l$) for all the q RS-MSR codes, while $d_j - d_{j+1}$ nodes only need to provide symbols $\tilde{p}_{i,l,t}$ ($0 \leq l \leq j$ and $1 \leq t \leq A/\alpha_l$) for the first $j+1$ RS-MSR codes ($0 \leq j \leq q-2$). In this way, the total number of symbols $\tilde{p}_{i,l,t}$ ($1 \leq t \leq A/\alpha_l$) for the l^{th} RS-MSR code that the helper nodes need to provide is $d_{q-1} + \sum_{j=l}^{q-2}(d_j - d_{j+1}) = d_l$.

Once the replacement node z' receives all the requested symbols, it can regenerate the symbols stored in the failed node z. The entire process is summarized in the following algorithm:

Algorithm 2: z' regenerates symbols of the failed node z

Step 1: **Calculate the regenerated symbols $\tilde{\mathbf{y}}_{z,l,t}$:** For every $0 \leq l \leq q-1$ and $1 \leq t \leq A/\alpha_l$, calculate the regenerated symbols based on $\tilde{p}_{i,l,t}$ from d_l helper nodes similar to [15]. Without loss of generality, we assume $0 \leq i \leq d_l - 1$:

Step 1.1: Let $\mathbf{p} = [\tilde{p}_{0,l,t}, \tilde{p}_{1,l,t}, \ldots, \tilde{p}_{d_l-1,l,t}]^T$, solve the equation: $\Psi_{0\to d_l-1,l}\mathbf{x} = \mathbf{p}$.

Step 1.2: Since $\mathbf{x} = \begin{bmatrix} S_{l,t} \\ T_{l,t} \end{bmatrix} \Phi_{z,l}^T$ and $S_{l,t}, T_{l,t}$ are symmetric, we can calculate $\mathbf{x}^T = [\Phi_{z,l}S_{l,t}, \Phi_{z,l}T_{l,t}]$.

Step 1.3: Compute $\tilde{\mathbf{y}}_{z,l,t} = \Phi_{z,l}S_{l,t} + \lambda_z\Phi_{z,l}T_{l,t} = \psi_{z,l}\begin{bmatrix} S_{l,t} \\ T_{l,t} \end{bmatrix}$.

Step 2: **Regenerate symbols of the failed node z:** Let $\widetilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $\widetilde{\mathbf{Y}}_{z,l} = [\tilde{\mathbf{y}}_{z,l,1}, \ldots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}]$ ($0 \leq l \leq q-1$). The symbols of the failed node z can be generated as: $Y_{z'} = Y_z = B_z\widetilde{Y}_z$.

Theorem 1 guarantees that Algorithm 2 can achieve the MSR point for data regeneration of the H-MSR code. Moreover, from Algorithm 2, we can derive the equivalent storage parameters for each symbol block of size $B_l = A(\alpha_l + 1)$: $d = 2\alpha_l$, $k = \alpha_l + 1$, $\alpha = A$, $\beta = A/\alpha_l$, $0 \leq l \leq q-1$ and equation (2) of the MSR point holds for these parameters.

Example 2: Fig. 3 is an example for the data regeneration of the H-MSR code illustrated in Fig. 2 with parameters $q = 3, \alpha_0 = 4, \alpha_1 = 3, \alpha_2 = 2$. According to the order of the points on the Hermitian curve in equation (11), the distribution of the codeword matrix Y in Algorithm 1 and the analysis in equation (14), symbols for α_0, α_1 and α_2 are stored in each of the storage nodes simultaneously. When storage node 0 fails, the contents stored in node 0 can be regenerated according to Algorithm 2 as shown in Fig. 3: α_0 requires help symbols from $d_0 = 8$ nodes, α_1 requires help symbols from $d_1 = 6$ nodes, α_2 requires help symbols from $d_2 = 4$ nodes.

B. Regeneration in Hostile Networks

In case that the network environment is hostile, Algorithm 2 may not be able to regenerate the failed node due to possible bogus symbols received in the responses. In fact, even if the replacement node z' can derive the symbol matrix $Y_{z'}$ using Algorithm 2, it cannot verify the correctness of the result.

We develop two modes for the helper nodes to regenerate the contents of a failed storage node in hostile networks: detection mode and recovery mode. The purpose of the detection mode is to detect whether the received help symbols contain errors. 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 in the way similar to that of the error-free networks in Section IV-A. 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. The regeneration algorithm is described in Algorithm 3 with the detection probability characterized in Lemma 1 and Theorem 3.

Algorithm 3 (Detection Mode): z' detects the bogus symbols and regenerates the symbols of the failed node z in hostile networks.

Step 1: **Error Detection:** For every $0 \leq l \leq q-1$ and $1 \leq t \leq A/\alpha_l$, we can calculate the regenerated symbols using the help symbols $\tilde{p}'_{i,l,t} = \tilde{p}_{i,l,t} + e_{i,l,t}$ from helper node i. If $\tilde{p}_{i,l,t}$ has been modified by the malicious node i, we have $e_{i,l,t} \in \mathbb{F}_{q^2}\backslash\{0\}$. Otherwise, we have $e_{i,l,t} = 0$. To detect whether there are $i's$ such that $e_{i,l,t} \neq 0$, we will compare regenerated symbols calculated from two sets of helper nodes. Without loss of generality, we assume $0 \leq i \leq d_l$.

Step 1.1: Let $\mathbf{p}_1{}' = [\tilde{p}'_{0,l,t}, \tilde{p}'_{1,l,t}, \ldots, \tilde{p}'_{d_l-1,l,t}]^T$, where the symbols are collected from node 0 to node $d_l - 1$, and solve the equation $\Psi_{0\to d_l-1,l}\mathbf{x}_1 = \mathbf{p}_1{}'$.

Step 1.2: Let $\mathbf{p}_2{}' = [\tilde{p}'_{1,l,t}, \tilde{p}'_{2,l,t}, \ldots, \tilde{p}'_{d_l,l,t}]^T$, where the symbols are collected from node 1 to node d_l, and solve the equation $\Psi_{1\to d_l,l}\mathbf{x}_2 = \mathbf{p}'_2$.

Step 1.3: If $\mathbf{x}_1 = \mathbf{x}_2$, no error has been detected. Compute $\tilde{\mathbf{y}}_{z,l,t} = \Phi_{z,l}S_{l,t} + \lambda_z\Phi_{z,l}T_{l,t}$ as described in Algorithm 2. Otherwise, at least one error has been detected in the help symbols. Exit the algorithm and switch to recovery regeneration mode.

Step 2: **Failed Node Regeneration:** Let $\widetilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $\widetilde{\mathbf{Y}}_{z,l} = [\tilde{\mathbf{y}}_{z,l,1}, \ldots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}]$ ($0 \leq l \leq q-1$). The symbols of the failed node z can be regenerated as: $Y_{z'} = Y_z = B_z\widetilde{Y}_z$.

Lemma 1: Let $e_i = e_{i,l,t}$ $(i = 0, \cdots, d_l)$ be defined as in Algorithm 3, $\hat{\mathbf{x}}_1 = \Psi^{-1}_{0\to d_l-1,l}[e_0, \ldots, e_{d_l-1}]^T$ and $\hat{\mathbf{x}}_2 = \Psi^{-1}_{1\to d_l,l}[e_1, \ldots, e_{d_l}]^T$. Then when the number of malicious nodes in the d_l+1 helper nodes is no more than d_l, the probability that $\hat{\mathbf{x}}_1 = \hat{\mathbf{x}}_2$ is at most $1/q^2$.

The proof of this lemma is given in the Appendix.

Theorem 3 (*H-MSR Regeneration–Detection Mode):* When the number of malicious nodes in the $d_l + 1$ helper nodes of Algorithm 3 is no more than d_l, the probability for the bogus symbols sent from the malicious nodes to be detected is at least $1 - 1/q^2$.

The proof of this theorem is given in the Appendix.

2) Recovery Mode: Once the replacement node z' detects errors using Algorithm 3, 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 provide help symbols similar to that of Section IV-A. z' can regenerate symbols using Algorithm 4.

Algorithm 4 (*Recovery Mode):* z' regenerates symbols of the failed node z in hostile networks.

Step 1: Calculate the regenerated symbols $\tilde{\mathbf{y}}_{z,l,t}$: For $l = q-1$ to 0 and $t = 1$ to A/α_l, we can regenerate the symbols when the errors in the $q^2 - 1$ received help symbols $\tilde{p}'_{i,l,t}$ can be corrected. Without loss of generality, we assume $0 \le i \le q^2 - 2$.

Step 1.1: Let $\mathbf{p}' = [\tilde{p}'_{0,l,t}, \tilde{p}'_{1,l,t}, \cdots, \tilde{p}'_{q^2-2,l,t}]^T$. 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.

Step 1.2: Since $\Psi_{0\to q^2-2,l}\mathbf{x} = \mathbf{p}'$, $\mathbf{p}'$ can be viewed as an MDS code with parameters (q^2-1, d_l, q^2-d_l). Decode $\mathbf{p}'$ to $\mathbf{p}'_{cw}$. If the i^{th} position symbols of $\mathbf{p}'_{cw}$ and $\mathbf{p}'$ are different, mark node i as corrupted. If the number of corrupted nodes detected is larger than $q^2 - d_{l-1} - 1$ for $l > 0$, flag the decoding as failure and exit the algorithm.

Step 1.3: Solve $\mathbf{x}$ in $\Psi_{0\to q^2-2,l}\mathbf{x} = \mathbf{p}'_{cw}$ and compute $\tilde{\mathbf{y}}_{z,l,t} = \Phi_{z,l}S_{l,t} + \lambda_z\Phi_{z,l}T_{l,t}$ as described in Algorithm 2.

Step 2: Regenerate symbols of the failed node z: Let $\widetilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $\widetilde{\mathbf{Y}}_{z,l} = [\tilde{\mathbf{y}}_{z,l,1}, \ldots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}], 0 \le l \le q-1$. The symbols of the failed node can be regenerated as z: $Y_{z'} = Y_z = B_z\widetilde{Y}_z$.

For data regeneration described in Algorithm 4, we have the following theorem:

Theorem 4 (*H-MSR Regeneration–Recovery Mode):* For data regeneration, the number of errors that the H-MSR code can correct is

$$\tau_{H-MSR} = q\lfloor(q^2 - d_{q-1} - 1)/2\rfloor. \tag{22}$$

Proof: Since H-MSR code can be viewed as q MDS codes with parameters (q^2-1, d_l, q^2-d_l), $l = 0, \ldots, q-1$, $\alpha_l \le \kappa(l)$ and $\kappa(l)$ is strictly decreasing, we can choose the sequence α_l to be strictly decreasing so that 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 4, 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. Next the code (q^2-1, d_l, q^2-d_l), $l = q-2, \ldots, 0$, will be decoded sequentially. The (q^2-1, d_l, q^2-d_l) code can correct at least $\tau_l = \tau_{q-1}$ errors when $q^2 - d_0 - 1 \ge \tau_{q-1}$. Thus, the total number of errors that the H-MSR code can correct is $\tau_{H-MSR} = q\tau_{q-1} = q\lfloor(q^2 - d_{q-1} - 1)/2\rfloor$. ■

V. Reconstruction of the H-MSR Code

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

A. Reconstruction in Error-Free Networks

The main idea of the algorithms is to reconstruct the l^{th} RS-MSR code, $0 \le l \le q-1$, by downloading help symbols from $k_l = \alpha_l + 1$ nodes, where k_l represents the reconstruction parameter k of the l^{th} RS-MSR code in the H-MSR code.

The reconstruction includes two major steps. For convenience, we assume $k_q = 0$. First, DC sends integer j to $k_j - k_{j+1}$ helper nodes that it has not sent requests before for j from $q-1$ to 0 in descending order. Upon receiving the requested integer j, node i will calculate $\widetilde{Y}_i = B_i^{-1}Y_i$, which eliminates the coefficient matrix B_i from the codeword matrix. Since the l^{th} row of $\widetilde{Y}_i$ is related to the symbols of the l^{th} RS-MSR code $(0 \le l \le j)$, node i will provide the l^{th} row of $\widetilde{Y}_i$: $\widetilde{\mathbf{Y}}_{i,l}$ to the DC. Thus j indicates that DC requests symbols related to the first $j+1$ RS-MSR codes.

Since $k_{l_1} > k_{l_2}$ if $l_1 < l_2$, for efficiency consideration, only k_{q-1} helper nodes need to provide symbols of $\widetilde{\mathbf{Y}}_{i,l}$ $(0 \le l \le q-1)$ related to all of the q RS-MSR codes. Then $k_j - k_{j+1}$ nodes only need to provide symbols of $\widetilde{\mathbf{Y}}_{i,l}$, $0 \le l \le j$, which are related to the first $j+1$ RS-MSR codes for $0 \le j \le q-2$. In this way, the total number of helper nodes that need to provide symbols of $\widetilde{\mathbf{Y}}_{i,l}$ related to the l^{th} RS-MSR code is $k_{q-1} + \sum_{j=l}^{q-2}(k_j - k_{j+1}) = k_l$.

Second, after DC receives all the requested symbols, it can reconstruct the original file using the following algorithm:

Algorithm 5: DC reconstructs the original file

Step 1: Prepare data: For every $0 \le l \le q-1$, divide the response symbol vector $\widetilde{\mathbf{Y}}_{i,l}$ from the i^{th} node into A/α_l equal row vectors: $[\tilde{\mathbf{y}}_{i,l,1}, \tilde{\mathbf{y}}_{i,l,2}, \ldots, \tilde{\mathbf{y}}_{i,l,A/\alpha_l}]$. Without loss of generality, we assume $0 \le i \le k_l - 1$.

Step 2: Reconstruct data matrices: For every $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$, DC reconstructs the matrices related to the original file:

Step 2.1: Let $R_{l,t} = [\tilde{\mathbf{y}}^T_{0,l,t}, \tilde{\mathbf{y}}^T_{1,l,t}, \ldots, \tilde{\mathbf{y}}^T_{k_l-1,l,t}]^T$, we have the equation: $\Psi_{0\to k_l-1,l}\begin{bmatrix}S_{l,t}\\ T_{l,t}\end{bmatrix} = R_{l,t}$ according to the encoding algorithm.

Step 2.2: DC reconstructs $S_{l,t}, T_{l,t}$ using techniques similar to [15].

Step 3: Reconstruct original file: DC reconstructs the original file from all the matrices $S_{l,t}, T_{l,t}$, $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$.

B. Reconstruction in Hostile Networks

Similar to the regeneration algorithms, the reconstruction algorithms in error-free networks do not work in hostile

networks. Even if the data collector can derive the symbol matrices S, T using Algorithm 5, it cannot verify whether the result is correct or not. There are two modes for the original file to be reconstructed in hostile networks: detection mode, which detects whether the received symbols contain errors, and recovery mode, which will correct errors and locate the malicious nodes.

1) Detection Mode: In detection mode, DC requests help symbols in the way similar to that of the error-free networks in Section V-A. 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. The reconstruction algorithm is described in Algorithm 6 with the detection probability described in Theorem 5.

Algorithm 6 (*Detection Mode):* DC detects the bogus symbols to reconstruct the original file in hostile networks

Step 1: Prepare data: For every $0 \le l \le q-1$, we can divide the symbol vector $\widetilde{\mathbf{Y}}'_{i,l}$ into A/α_l equal row vectors: $[\tilde{\mathbf{y}}'_{i,l,1}, \tilde{\mathbf{y}}'_{i,l,2}, \ldots, \tilde{\mathbf{y}}'_{i,l,A/\alpha_l}]$. $\widetilde{\mathbf{Y}}'_{i,l} = \widetilde{\mathbf{Y}}_{i,l} + \mathbf{e}_{i,l}$ is the response from the i^{th} storage node. If $\widetilde{\mathbf{Y}}_{i,l}$ has been modified by the malicious node i, we have $\mathbf{e}_{i,l} \in \mathbb{F}_{q^2}^A \backslash \{\mathbf{0}\}$. To detect whether there are $i's$ such that $\mathbf{e}_{i,l} \neq \mathbf{0}$, we will compare reconstructed symbols calculated from two sets of storage nodes. Without loss of generality, we assume $0 \le i \le k_l$.

Step 2: Reconstruct data matrices: For every $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$, DC reconstructs the matrices related to the original file:

Step 2.1: Let $R'_{1,l,t} = [\tilde{\mathbf{y}}'^T_{0,l,t}, \tilde{\mathbf{y}}'^T_{1,l,t}, \ldots, \tilde{\mathbf{y}}'^T_{\alpha_l,l,t}]^T$, which are the symbols collected from node 0 to node $k_l - 1 = \alpha_l$, then we have $\Psi_{0\to\alpha_l,l} \begin{bmatrix} S_1 \\ T_1 \end{bmatrix} = R'_{1,l,t}$. Solve S_1, T_1 using the method same to algorithm 5.

Step 2.2: Let $R'_{2,l,t} = [\tilde{\mathbf{y}}'^T_{0,l,t}, \ldots, \tilde{\mathbf{y}}'^T_{\alpha_l-1,l,t}, \tilde{\mathbf{y}}'^T_{\alpha_l+1,l,t}]^T$, which are the symbols collected from node 0 to node $k_l = \alpha_l + 1$ except node α_l, then we have $\begin{bmatrix} \Psi_{0\to\alpha_l-1,l} \\ \Psi_{\alpha_l+1,l} \end{bmatrix} \begin{bmatrix} S_2 \\ T_2 \end{bmatrix} = R'_{2,l,t}$. Solve S_2, T_2 using the method same to algorithm 5.

Step 2.3: If $[S_1, T_1] = [S_2, T_2]$, let $[S_{l,t}, T_{l,t}] = [S_1, T_1]$. Otherwise, errors are detected in the received symbols. Exit the algorithm and switch to recovery reconstruction mode.

Step 3: Reconstruct original file: DC reconstructs the original file from all the matrices $S_{l,t}, T_{l,t}$, $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$.

Theorem 5 (*H-MSR Reconstruction–Detection Mode):* When the number of malicious nodes in the $k_l + 1$ nodes of Algorithm 6 is less than $k_l + 1$, the probability for the bogus symbols sent from the malicious nodes to be detected is at least $1 - (1/q^2)^{2(\alpha_l - 2)}$.

The proof of this theorem is given in the Appendix.

2) Recovery Mode: Once DC detects errors using Algorithm 6, it will send integer $j = q-1$ to all the q^2 nodes in the network requesting symbols. The reconstruction procedures are described in Algorithm 7.

Algorithm 7 (*Recovery Mode):* DC reconstructs the original file in hostile networks

Step 1: Prepare data: For $l = 0$ to $q-1$, divide the symbol vector $\widetilde{\mathbf{Y}}'_{i,l}$ into A/α_l equal row vectors: $[\tilde{\mathbf{y}}'_{i,l,1}, \tilde{\mathbf{y}}'_{i,l,2}, \ldots, \tilde{\mathbf{y}}'_{i,l,A/\alpha_l}]$. Without loss of generality, we assume $0 \le i \le q^2 - 1$.

Step 2: Reconstruct data matrices: For $l = q-1$ to 0 and $t = 1$ to A/α_l, DC reconstructs the matrices related to the original file when the errors in the received symbol vectors $\tilde{\mathbf{y}}'_{i,l,t}$ from q^2 storage nodes can be corrected:

Step 2.1: Let $R'_{l,t} = [\tilde{\mathbf{y}}'^T_{0,l,t}, \tilde{\mathbf{y}}'^T_{1,l,t}, \ldots, \tilde{\mathbf{y}}'^T_{q^2-1,l,t}]^T$. Substitute $\tilde{\mathbf{y}}'_{i,l,t}$ in $R'_{l,t}$ with the symbol $\otimes$ representing an erasure vector if node i has been detected to be corrupted.

Step 2.2: Solve $S_{l,t}, T_{l,t}$ using the method described in section V-C. If symbols from node i are detected to be erroneous during the calculation, mark node i as corrupted. If the number of corrupted nodes detected is larger than $q^2 - k_{l-1}$ for $l > 0$, flag the decoding failure and exit the algorithm.

Step 3: Recover original file: DC recovers the original file from all the matrices $S_{l,t}, T_{l,t}$, $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$.

For data reconstruction described in Algorithm 7, we have the following theorem:

Theorem 6 (*H-MSR Reconstruction–Recovery Mode):* For data reconstruction, the number of errors that the H-MSR code can correct is

$$\tau_{H-MSR-REC} = q\lfloor (q^2 - k_{q-1})/2 \rfloor. \tag{23}$$

Proof: Similar to the proof of Theorem 4, for data reconstruction Algorithm 7, H-MSR code can be viewed 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. Therefore, we have results similar to that of equation (22). ■

C. Recover Matrices $S_{l,t}, T_{l,t}$ From q^2 Storage Nodes

Reconstruction of data matrices $S_{l,t}, T_{l,t}$ using symbols collected from k storage nodes was first proposed in [15], which is also employed in our proposed Algorithms 5 and 6. When the symbols $\tilde{p}'_{i,l,t}$ collected are bogus for some l, t, we propose the following technique to correct errors and recover the matrices $S_{l,t}, T_{l,t}$ from q^2 storage nodes. (For convenience, we write $R'_{l,t}$ in Algorithm 7 as R'):

For R' in Algorithm 7 and $\Psi_l = [\Phi_l, \Delta\Phi_l]$, we have $\Psi_l \begin{bmatrix} S' \\ T' \end{bmatrix} = R'$, and

$$\Phi_l S' \Phi_l^T + \Delta\Phi_l T' \Phi_l^T = R' \Phi_l^T. \tag{24}$$

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

$$C + \Delta D = \widehat{R}'. \tag{25}$$

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

$$\begin{cases} C_{i,j} + \lambda_i D_{i,j} = \widehat{R}'_{i,j} \\ C_{i,j} + \lambda_j D_{i,j} = \widehat{R}'_{j,i}. \end{cases} \tag{26}$$

Because matrices C and D have the same structure, we only focus on C (corresponding to S'). 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 $S'\Phi_l^T = \widehat{S}'$, we have

$$\Phi_l \widehat{S}' = C. \tag{27}$$

Here we can view each column of C as a $(q^2-1, \alpha_l, q^2-\alpha_l)$ MDS code because Φ_l 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:

$$\sigma + 2\tau + 1 \leq q^2 - \alpha_l, \tag{28}$$

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 reconstruct $S_{l,t}$ using the method similar to [15]. For $T_{l,t}$, the recovering process is similar.

VI. Encoding of the H-MBR Code

In this section, we propose the H-MBR code based on the MBR point with $\beta = 1$ under the product-matrix framework. According to equation (3), we have $d = \alpha$.

Let $\alpha_0, \ldots, \alpha_{q-1}$ be a strictly decreasing integer sequence satisfying $0 < \alpha_l \leq \kappa(l)$, $0 \leq l \leq q-1$. The least common multiple of $\alpha_0, \ldots, \alpha_{q-1}$ is A. Let $k_0, \ldots, k_{q-1}$ be an integer sequence satisfying $0 < k_l \leq \alpha_l$, $0 \leq l \leq q-1$. Suppose the data contains $B = A\sum_{l=0}^{q-1}(k_l(2\alpha_l - k_l + 1)/(2\alpha_l))$ message symbols from the finite field $\mathbb{F}_{q^2}$. In practice, if the size of 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 matrix M as below:

$$M = \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{q-1} \end{bmatrix}, \tag{29}$$

where

$$M_l = [M_{l,1}, M_{l,2}, \ldots, M_{l,A/\alpha_l}], \tag{30}$$

$$M_{l,j} = \begin{bmatrix} S_{l,j} & T_{l,j} \\ T_{l,j}^T & \mathbf{0} \end{bmatrix}, \tag{31}$$

and $S_{l,j}$ $(0 \leq l \leq q-1, 1 \leq j \leq A/\alpha_l)$ is a symmetric matrix of size $k_l \times k_l$ with the upper-triangular entries filled by data symbols. $T_{l,j}$ is a $k_l \times (\alpha_l - k_l)$ matrix. Thus $M_{l,j}$ contains $k_l(2\alpha_l - k_l + 1)/2$ symbols, M_l contains $Ak_l(2\alpha_l - k_l + 1)/(2\alpha_l)$ symbols and M contains B symbols.

For distributed storage, we encode M using Algorithm 8:

Algorithm 8: Encoding H-MBR Code

Step 1: First we encode the data matrices M defined above using a Hermitian code $\mathcal{H}_m$ over $\mathbb{F}_{q^2}$ with parameters $\kappa(l)$ $(0 \leq l \leq q-1)$ and m $(m \geq q^2-1)$. The $q^3 \times A$ codeword matrix can be written as $Y = \mathcal{H}_m(M)$.

Step 2: Then we divide the codeword matrix Y into q^2 submatrices $Y_0, \ldots, Y_{q^2-1}$ of size $q \times A$ and store one submatrix in each of the q^2 storage nodes as shown in Fig. 1.

Then we have the following theorem:

Theorem 7: By processing the data symbols using Algorithm 8, we can achieve the MBR point in distributed storage.

Proof: Similar to the proof of Theorem 1, we can get the following equation considering all the columns of $\mathcal{H}_m(M)$:

$$\Phi_l M_{l,j} = U_{l,j}, \tag{32}$$

where $U_{l,j} = [\mathcal{U}_l^{(1)}, \ldots, \mathcal{U}_l^{(\alpha_l)}]$, $0 \leq l \leq q-1$, $1 \leq j \leq A/\alpha_l$, $\mathcal{U}_l^{(t)}$ corresponds to the t^{th} column of the submatrix $M_{l,j}$, each element of $\mathcal{U}_l = [u_l(0), u_l(1), \ldots, u_l(\phi^{q^2-2})]^T$ can be derived from a distinct storage node, and Φ_l is defined in equation (12).

Next we will study the optimality of the code in the sense of the MBR point. For $\Phi_l M_{l,j}$, $0 \leq l \leq q-1, 1 \leq j \leq A/\alpha_l$, $M_{l,j}$ is symmetric and satisfies the requirements for MBR point according to [15] with parameters $d = \alpha_l, k = k_l, \alpha = \alpha_l, \beta = 1, B = k_l(2\alpha_l - k_l + 1)/2$. By encoding M using $\mathcal{H}_m(M)$ and distributing $Y_0, \ldots, Y_{q^2-1}$ into q^2 storage nodes, each row of the matrix $\Phi_l M_{l,j}$, $0 \leq l \leq q-1$, $1 \leq j \leq A/\alpha_l$, can be derived in a corresponding storage node. Because $\Phi_l M_{l,j}$ achieves the MBR point, data related to matrices $M_{l,j}$, $0 \leq l \leq q-1$, $1 \leq j \leq A/\alpha_l$, can be regenerated at the MBR point. Therefore, Algorithm 8 can achieve the MBR point. ∎

Similar to the decomposition of the H-MSR code into q concatenated RS-MSR codes, we have Theorem 8. In [15], the RS-MBR code under the product-matrix framework was explicitly constructed along with the corresponding algorithms for data regeneration and reconstruction. According to Theorem 8, the construction of the H-MBR code under the product-matrix framework is feasible intuitively. Then we provide accurate algorithms and strict proofs to demonstrate the correctness and efficacy of the H-MBR code in the paper.

Theorem 8: The H-MBR code could be viewed as the concatenation of q RS-MBR codes. Every storage node will contain symbols related to each of the q RS-MBR codes encoded using Algorithm 8.

Proof: (sketch) According to equation (32) in the proof of theorem 7, the H-MBR code can be viewed as q concatenated RS-MSR codes, where the encoding matrix of the l^{th} RS-MSR code is Φ_l for $0 \leq l \leq q-1$.

For any column $\mathrm{Col}(Y_i)$ of the submatrix Y_i in storage node i, we have the following equation:

$$\mathbf{u}_i = B_i^{-1}\mathrm{Col}(Y_i) = [u_0(\phi^{s_i}), u_1(\phi^{s_i}), \ldots, u_{q-1}(\phi^{s_i})]^T. \tag{33}$$

With $u_l(\phi^{s_i}), 0 \le l \le q-1$ corresponding to the l^{th} RS-MBR code, the codewords related to all of the q RS-MBR codes are stored in each of the storage node i. ■

VII. Regeneration of the H-MBR Code

In this section, we will first discuss regeneration of the H-MBR code in error-free networks. Then we will discuss regeneration in hostile networks.

A. Regeneration in Error-Free Networks

The main idea of the regeneration algorithms is similar to that of the H-MSR code: regenerate the l^{th} RS-MBR code, $0 \le l \le q-1$, by downloading help symbols from $d_l = \alpha_l$ nodes, where d_l is the regeneration parameter d for the l^{th} RS-MBR code in the H-MBR code regeneration.

Suppose node z fails, we use Algorithm 9 to regenerate the exact H-MBR code symbols of node z. For convenience, we suppose $d_q = \alpha_q = 0$ and define

$$\Phi_{i\to j,l} = \begin{bmatrix} \Phi_{i,l} \\ \Phi_{i+1,l} \\ \vdots \\ \Phi_{j,l} \end{bmatrix}, \tag{34}$$

where $\Phi_{t,l}$, $i \le t \le j$, is the t^{th} row of Φ_l.

Similar to the H-MSR code, replacement node z' will send requests to helper nodes in the way same as that in Section IV-A. Upon receiving the request integer j, helper node i will calculate and send the help symbols similar to that of Section IV-A.

When the replacement node z' receives all the requested symbols, it can regenerate the symbols stored in the failed node z using the following algorithm:

Algorithm 9: z' regenerates symbols of the failed node z

Step 1: **Calculate the regenerated symbols $\tilde{\mathbf{y}}_{z,l,t}$:** For every $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$, calculate the regenerated symbols based on $\tilde{p}_{i,l,t}$ from d_l helper nodes, using techniques similar to [15]. Without loss of generality, we assume $0 \le i \le d_l - 1$.
Step 1.1: Let $\mathbf{p} = [\tilde{p}_{0,l,t}, \tilde{p}_{1,l,t}, \ldots, \tilde{p}_{d_l-1,l,t}]^T$, solve the equation: $\Phi_{0\to d_l-1,l}\mathbf{x} = \mathbf{p}$.
Step 1.2: Since $\mathbf{x} = M_{l,t}\Phi_{z,l}^T$ and $M_{l,t}$ is symmetric, we can calculate $\tilde{\mathbf{y}}_{z,l,t} = \mathbf{x}^T = \Phi_{z,l}M_{l,t}$.

Step 2: **Regenerate symbols of the failed node z:** Let $\widetilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $\widetilde{\mathbf{Y}}_{z,l} = [\tilde{\mathbf{y}}_{z,l,1}, \ldots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}], 0 \le l \le q-1$. Calculate the regenerated symbols of the failed node z: $Y_{z'} = Y_z = B_z\widetilde{Y}_z$.

For Algorithm 9 we can derive the equivalent storage parameters for each symbol block of size $B_l = Ak_l(2\alpha_l - k_l + 1)/(2\alpha_l)$: $d = \alpha_l, k = k_l, \alpha = A, \beta = A/\alpha_l$, $0 \le l \le q-1$ and equation (3) of the MBR point holds for these parameters. Theorem 7 guarantees that Algorithm 9 can achieve the MBR point for data regeneration of the H-MBR code.

B. Regeneration in Hostile Networks

In hostile networks, Algorithm 9 may be unable to regenerate the failed node due to possible bogus symbols received in the responses. In fact, even if the replacement node z' can derive the symbol matrix $Y_{z'}$ using Algorithm 9, it cannot verify the correctness of the result.

Similar to the H-MSR code, there are two modes for the helper nodes to regenerate the H-MBR code of a failed storage node in hostile networks. One mode is the detection mode. The purpose of this mode is to detect possible errors 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, both the replacement node z' and the helper nodes will exchange information similar to that of the error-free networks in Section VII-A. 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. The regeneration algorithm is described in Algorithm 10 with the detection probability characterized in Theorem 9.

Algorithm 10 (*Detection Mode*): z' detects the bogus symbols and regenerates the symbols of the failed node z in hostile networks

Step 1: **Error Detection:** For every $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$, we can calculate the regenerated symbols using the help symbols $\tilde{p}'_{i,l,t} = \tilde{p}_{i,l,t} + e_{i,l,t}$ from helper node i. If $\tilde{p}_{i,l,t}$ has been modified by the malicious node i, we have $e_{i,l,t} \in \mathbb{F}_{q^2}\backslash\{0\}$. To detect whether there are $i's$ such that $e_{i,l,t} \ne 0$, we will compare regenerated symbols calculated from two sets of helper nodes. Without loss of generality, we assume $0 \le i \le d_l$.
Step 1.1: Let $\mathbf{p}_1{}' = [\tilde{p}'_{0,l,t}, \tilde{p}'_{1,l,t}, \ldots, \tilde{p}'_{d_l-1,l,t}]^T$, where the symbols are collected from node 0 to node $d_l - 1$, and solve the equation $\Phi_{0\to d_l-1,l}\mathbf{x}_1 = \mathbf{p}'_1$.
Step 1.2: Let $\mathbf{p}_2{}' = [\tilde{p}'_{1,l,t}, \tilde{p}'_{2,l,t}, \ldots, \tilde{p}'_{d_l,l,t}]^T$, where the symbols are collected from node 1 to node d_l, and solve the equation $\Phi_{1\to d_l,l}\mathbf{x}_2 = \mathbf{p}_2{}'$.
Step 1.3: If $\mathbf{x}_1 = \mathbf{x}_2$, compute $\tilde{\mathbf{y}}_{z,l,t} = \Phi_{z,l}M_{l,t}$ as described in Algorithm 9. Otherwise, errors are detected in the help symbols. Exit the algorithm and switch to recovery regeneration mode.

Step 2: **Failed Node Regeneration:** Let $\widetilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $\widetilde{\mathbf{Y}}_{z,l} = [\tilde{\mathbf{y}}_{z,l,1}, \ldots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}], 0 \le l \le q-1$. Calculate the regenerated symbols of the failed node z: $Y_{z'} = Y_z = B_z\widetilde{Y}_z$.

Theorem 9 (*H-MBR Regeneration–Detection Mode*): When the number of malicious nodes in the $d_l + 1$ helper nodes of Algorithm 10 is less than $d_l + 1$, the probability for the bogus symbols sent from the malicious nodes to be detected is at least $1 - 1/q^2$.

The proof of this theorem is given in the Appendix.

2) Recovery Mode: Once the replacement node z' detects errors using Algorithm 10, it will send integer $j = q-1$ to all the other $q^2 - 1$ nodes in the network requesting help symbols. z' can regenerate symbols using Algorithm 11.

Algorithm 11 (*Recovery Mode*): z' regenerates symbols of the failed node z in hostile networks

Step 1: Calculate the regenerated symbols $\tilde{\mathbf{y}}_{z,l,t}$: For $l = q-1$ to 0 and $t = 1$ to A/α_l, we can regenerate the symbols when the errors in the q^2-1 received help symbols $\tilde{p}'_{i,l,t}$ can be corrected. Without loss of generality, we assume $0 \le i \le q^2-2$.

Step 1.1: Let $\mathbf{p}' = [\tilde{p}'_{0,l,t}, \tilde{p}'_{1,l,t}, \ldots, \tilde{p}'_{q^2-2,l,t}]^T$. 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.

Step 1.2: Since $\Phi_{0\to q^2-2,l}\mathbf{x} = \mathbf{p}'$, $\mathbf{p}'$ can be viewed as an MDS code with parameters (q^2-1, d_l, q^2-d_l). Decode $\mathbf{p}'$ to $\mathbf{p}'_{cw}$. If the i^{th} position symbols of $\mathbf{p}'_{cw}$ and $\mathbf{p}'$ are different, mark node i as corrupted. If the number of corrupted nodes detected is larger than $q^2-d_{l-1}-1$ for $l > 0$, flag the decoding failure and exit the algorithm.

Step 1.3: Solve $\mathbf{x}$ in $\Phi_{0\to q^2-2,l}\mathbf{x} = \mathbf{p}'_{cw}$. Compute $\tilde{\mathbf{y}}_{z,l,t} = \Phi_{z,l}M_{l,t}$ as described in Algorithm 9.

Step 2: Regenerate symbols of the failed node z: Let $\widetilde{Y}_z$ be a $q \times A$ matrix with the l^{th} row defined as $\widetilde{\mathbf{Y}}_{z,l} = [\tilde{\mathbf{y}}_{z,l,1}, \ldots, \tilde{\mathbf{y}}_{z,l,A/\alpha_l}], 0 \le l \le q-1$. Calculate the regenerated symbols of the failed node: $Y_{z'} = Y_z = B_z\widetilde{Y}_z$.

For data regeneration described in Algorithm 11, since the structures of the underlying Hermitian codes of H-MSR code and H-MBR code with the same code rates are the same, we have a similar result as that in Theorem 4.

Theorem 10 (*H-MBR Regeneration–Recovery Mode):* For data regeneration, the number of errors that the H-MBR code can correct is

$$\tau_{H-MBR} = q\lfloor (q^2 - d_{q-1} - 1)/2 \rfloor. \tag{35}$$

VIII. Reconstruction of the H-MBR Code

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

A. Reconstruction in Error-Free Networks

The main idea of the reconstruction algorithms is similar to that of the H-MSR code: reconstruct the l^{th} RS-MBR code, $0 \le l \le q-1$, by downloading help symbols from k_l nodes, where k_l represents the reconstruction parameter k for the l^{th} RS-MBR code in the H-MBR code. We use Algorithm 12 in the network for the data collector DC to reconstruct the original file. For convenience, we suppose $k_q = 0$.

Similar to the H-MSR code described in Section V-A, DC will send requests to storage nodes. Upon receiving the request integer j, node i will calculate and send symbols. After DC receives all the requested symbols, it can reconstruct the original file using the following algorithm:

Algorithm 12: DC reconstructs the original file

Step 1: Prepare Data: For every $0 \le l \le q-1$, divide the symbol vector $\widetilde{\mathbf{Y}}_{i,l}$ from the i^{th} node into A/α_l equal row vectors: $[\tilde{\mathbf{y}}_{i,l,1}, \tilde{\mathbf{y}}_{i,l,2}, \ldots, \tilde{\mathbf{y}}_{i,l,A/\alpha_l}]$. Without loss of generality, we assume $0 \le i \le k_l - 1$.

Step 2: Reconstruct data matrices: For every $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$, DC reconstructs the matrices related to the original file:

Step 2.1: Let $R_{l,t} = [\tilde{\mathbf{y}}^T_{0,l,t}, \tilde{\mathbf{y}}^T_{1,l,t}, \ldots, \tilde{\mathbf{y}}^T_{k_l-1,l,t}]^T$, we have the equation: $\Phi_{0\to k_l-1,l}M_{l,t} = R_{l,t}$ according to the encoding algorithm.

Step 2.2: DC reconstructs $M_{l,t}$ using techniques similar to that of [15].

Step 3: Reconstruct original file: DC reconstructs the original file from all the matrices $M_{l,t}$, $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$.

B. Reconstruction in Hostile Networks

Similar to the H-MSR code, the reconstruction algorithms for H-MBR code in error-free networks do not work in hostile networks. Even if the data collector can calculate the symbol matrices M using Algorithm 12, it cannot verify whether the result is correct or not. There are two modes for the original file to be reconstructed in hostile networks. 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 in the way similar to that of the error-free networks in Section VIII-A. 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 send symbols similar to that of the error-free networks in Section VIII-A. The reconstruction algorithm is described in Algorithm 13 with the detection probability described in Theorem 11.

Algorithm 13 (*Detection Mode):* DC detects the bogus symbols to reconstruct the original file in hostile networks

Step 1: Prepare data: For every $0 \le l \le q-1$, we can divide the symbol vector $\widetilde{\mathbf{Y}}'_{i,l}$ into A/α_l equal row vectors: $[\tilde{\mathbf{y}}'_{i,l,1}, \tilde{\mathbf{y}}'_{i,l,2}, \ldots, \tilde{\mathbf{y}}'_{i,l,A/\alpha_l}]$. $\widetilde{\mathbf{Y}}'_{i,l} = \widetilde{\mathbf{Y}}_{i,l} + \mathbf{e}_{i,l}$ is the response from the i^{th} storage node. If $\widetilde{\mathbf{Y}}_{i,l}$ has been modified by the malicious node i, we have $\mathbf{e}_{i,l} \in \mathbb{F}_{q^2}^A \backslash \{\mathbf{0}\}$. To detect whether there are $i's$ such that $\mathbf{e}_{i,l} \ne \mathbf{0}$, we will compare reconstructed symbols calculated from two sets of storage nodes. Without loss of generality, we assume $0 \le i \le k_l$.

Step 2: Reconstruct data matrices: For every $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$, DC can reconstruct the matrices related to the original file:

Step 2.1: Let $R'_{1,l,t} = [\tilde{\mathbf{y}}'^T_{0,l,t}, \ldots, \tilde{\mathbf{y}}'^T_{k_l-1,l,t}]^T$, which are the symbols collected from node 0 to node k_l-1, then we have $\Phi_{0\to k_l-1,l}M_1 = R'_{1,l,t}$. Solve M_1 using the method same to algorithm 12.

Step 2.2: Let $R'_{2,l,t} = [\tilde{\mathbf{y}}'^T_{1,l,t}, \ldots, \tilde{\mathbf{y}}'^T_{k_l,l,t}]^T$, which are the symbols collected from node 1 to node k_l, then we have $\Phi_{1\to k_l,l}M_2 = R'_{2,l,t}$. Solve M_2 using the method same to algorithm 12.

Step 2.3: If $M_1 = M_2$, let $M_{l,t} = M_1$. Otherwise, errors are detected in the received symbols. Exit the algorithm and switch to recovery reconstruction mode.

Step 3: Reconstruct original file: DC reconstructs the original file from all the matrices $M_{l,t}$, $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$.

Theorem 11 (*H-MBR Reconstruction–Detection Mode):* When the number of malicious nodes in the k_l+1 nodes of Algorithm 13 is less than k_l+1, the probability for the bogus symbols sent from the malicious nodes to be detected is at least $1-1/q^{2\alpha_l}$.

The proof of this theorem is given in the Appendix.

2) Recovery Mode: Once DC detects errors using Algorithm 13, it will send integer $j=q-1$ to all the q^2 nodes in the network requesting symbols. Storage node i will use the way similar to that of the error-free networks in Section VIII-A to send symbols. The reconstruct procedures are described in Algorithm 14.

Algorithm 14 (*Recovery Mode):* DC reconstructs the original file in hostile networks

Step 1: Prepare data: For every $0 \le l \le q-1$, divide the symbol vector $\widetilde{\mathbf{Y}}'_{i,l}$ into A/α_l equal row vectors: $[\tilde{\mathbf{y}}'_{i,l,1}, \tilde{\mathbf{y}}'_{i,l,2}, \ldots, \tilde{\mathbf{y}}'_{i,l,A/\alpha_l}]$. Without loss of generality, we assume $0 \le i \le q^2-1$.

Step 2: Reconstruct data matrices: For $l=q-1$ to 0 and $t=1$ to A/α_l, DC reconstructs the matrices related to the original file when the errors in the received symbol vectors $\tilde{\mathbf{y}}'_{i,l,t}$ from q^2 storage nodes can be corrected:

Step 2.1: Let $R'_{l,t} = [\tilde{\mathbf{y}}'^T_{0,l,t}, \tilde{\mathbf{y}}'^T_{1,l,t}, \ldots, \tilde{\mathbf{y}}'^T_{q^2-1,l,t}]^T$. substitute $\tilde{\mathbf{y}}'_{i,l,t}$ in $R'_{l,t}$ with the symbol $\otimes$ representing an erasure vector if node i has been detected to be corrupted.

Step 2.2: Solve $M_{l,t}$ using the method in section VIII-C. If symbols from node i are detected to be erroneous during the calculation, mark node i as corrupted. If the number of corrupted nodes detected is larger than q^2-k_{l-1} for $l>0$, flag the decoding failure and exit the algorithm.

Step 3: Reconstruct original file: DC reconstructs the original file from all the matrices $M_{l,t}$, $0 \le l \le q-1$ and $1 \le t \le A/\alpha_l$.

For data reconstruction described in Algorithm 14, since the structures of the underlying Hermitian codes of H-MSR code and H-MBR code with the same code rates are the same, we have a similar result as that in Theorem 6.

Theorem 12 (*H-MBR Reconstruction–Recovery Mode):* For data reconstruction, the number of errors that the H-MBR code can correct is

$$\tau_{H-MBR-REC} = q\lfloor (q^2-k_{q-1})/2 \rfloor. \tag{36}$$

C. Recover Matrices $M_{l,t}$ From q^2 Storage Nodes

Reconstruction of data matrices $M_{l,t}$ using symbols collected from k storage nodes was first proposed in [15], which is also employed in our proposed Algorithms 12 and 13. When there are bogus symbols $\tilde{p}'_{i,l,t}$ sent by the corrupted nodes for certain l,t, we can recover the matrices $M_{l,t}$ as follows (For convenience, we write $R'_{l,t}$ in Algorithm 14 as R'):

For R' in Algorithm 14, we have $\Phi_l M' = R'$, where $\Phi_l = [\Phi_l^{(1)}, \Phi_l^{(2)}]$, $R' = [R'_1, R'_2]$. $\Phi_l^{(1)}, R'_1$ are $q^2 \times k_l$ submatrices and $\Phi_l^{(2)}, R'_2$ are $q^2 \times (\alpha_l - k_l)$ submatrices.

According to equation (31), we have

$$\Phi_l M' = [\Phi_l^{(1)} S' + \Phi_l^{(2)} T'^T, \Phi_l^{(1)} T'] = [R'_1, R'_2]. \tag{37}$$

For $R'_2 = \Phi_l^{(1)} T'$, we can view each column of R'_2 as a (q^2, k_l, q^2-k_l+1) MDS code because Φ_l is a Vandermonde matrix. If the number of erasures σ (corresponding to the previously detected corrupted nodes) and the number of corrupted nodes τ that have not been detected satisfy:

$$\sigma + 2\tau \le q^2 - k_l, \tag{38}$$

then all the columns of T' can be recovered and the error locations (corresponding to the corrupted nodes) can be pinpointed. After T' has been recovered, we can recover S' following the same process because $\Phi_l^{(1)} S' = R'_1 - \Phi_l^{(2)} T'^T$. So DC can reconstruct $M_{l,t}$.

IX. Performance Analysis

In this section, we analyze the performance of the H-MSR code and compare it with the performance of the RS-MSR code. The comparison between the H-MBR code and the RS-MBR code is the same. Therefore, it will be skipped here.

A. Scalable Error Correction

1) Error Correction for Data Regeneration: The RS-MSR code introduced in [24] can correct up to τ errors by downloading symbols from $d+2\tau$ nodes. However, the number of errors contained in the received help symbols may vary. When there is no error or the number of errors is far less than τ, downloading help symbols from too many nodes is a waste of bandwidth. On the other hand, 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 3. If no error is detected, regeneration of the H-MSR only needs to download help symbols from one more node than the regeneration in error-free networks, while the extra cost for the RS-MSR code is 2τ. If errors are detected in the help symbols, the H-MSR code can correct the errors using Algorithm 4. 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 6. The errors can then be corrected using Algorithm 7. The algorithm can also determine whether the decoding is successful.

B. Error Correction Performance

For data regeneration described in Algorithm 4, according to Theorem 4 and equation (22), the H-MSR code can correct $\tau_{H-MSR} = q\lfloor (q^2 - d_{q-1} - 1)/2 \rfloor$ errors, 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 13: For 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

$$\begin{aligned} \tau_{RS-MSR} &= \left\lfloor \left(q^3 - q - \sum_{l=0}^{q-1} d_l \right) /2 \right\rfloor \qquad (39) \\ &\leq \left\lfloor (q^3 - q - q d_{q-1} - \frac{q}{2}(q-1))/2 \right\rfloor \\ &= \left\lfloor q(q^2 - d_{q-1} - 1)/2 - \frac{q(q-1)}{4} \right\rfloor \\ &\leq q(q^2 - d_{q-1} - 1)/2 - \frac{q(q-1)}{4}. \end{aligned}$$

For τ_{H-MSR}, we have

$$\tau_{H-MSR} = q\lfloor (q^2 - d_{q-1} - 1)/2 \rfloor. \qquad (40)$$

When $q = 3$, it is easy to verify that $\tau_{H-MSR} > \tau_{RS-MSR}$. When $q > 3$, We can rewrite equation (40) as

$$\tau_{H-MSR} \geq q(q^2 - d_{q-1} - 1)/2 - q/2. \qquad (41)$$

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

$$\frac{q(q-1)}{4} - \frac{q}{2} = \frac{q^2 - 3q}{4} > 0, \ q > 3, \qquad (42)$$

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

Example 3: Suppose $q = 4$ and $m = 37$, the Hermitian curve is defined by $y^4 + y = x^5$ over $\mathbb{F}_{4^2}$. 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 $\tau_{H-MSR} = 4\tau_3 = 4\lfloor (15 - 6)/2 \rfloor = 16$, which is larger than $\tau_{RS-MSR} = \lfloor (60 - 36)/2 \rfloor = 12$.

We also show that the maximum number of malicious nodes from which errors can be corrected by the H-MSR code in Fig. 4. The parameter q of the Hermitian code increases from 4 to 16. Take $q = 5$ as an example. We choose $d_0 = 14, d_1 = 12, d_2 = 10, d_3 = 8, d_4 = 6$. According to the decoding process in Algorithm 4, the 4^{th} layer with $d_4 = 6$ will be decoded first. The equivalent MDS code in this layer is a (24, 6, 19) code and can correct 9 errors. Next we decode the 3^{rd} layer with $d_3 = 8$ and equivalent MDS code (24, 8, 17). For this layer, the symbols corresponding to the errors detected in the 4^{th} layer will be viewed as erasures. Thus the 3^{rd} layer code can correct up to 3 more errors. For the first layer with $d_0 = 14$, the equivalent MDS code is (24, 14, 11). It can correct at most 10 erasures. Therefore, the 3^{rd} layer code can correct at most 1 more error. In the decodings of subsequent layers, symbols from the 10 nodes will be regarded as erasures. Thus for $q = 5$ in the figure, the maximum number of nodes from which errors can be corrected is 10.

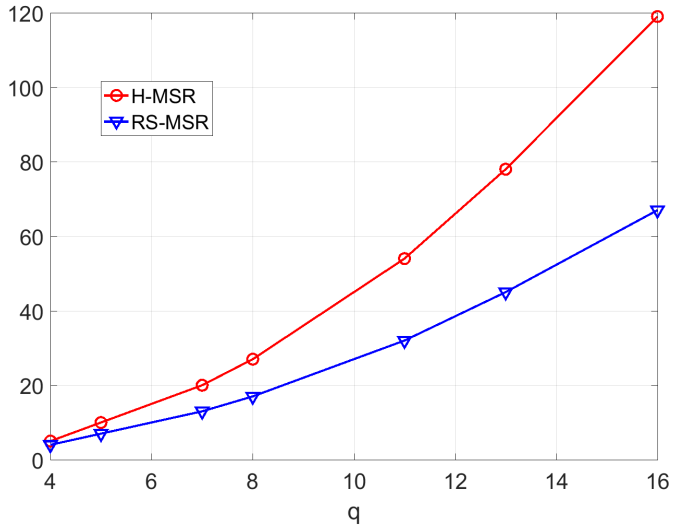

Fig. 4. Comparison of the H-MSR code and the RS-MSR code for the number of corrupted/manipulated nodes from which errors can be corrected.

For this code, the overall code dimension for the 5 layers is $\sum_{i=0}^{4} d_i = 50$. For RS code with comparable performance, the code would be (24, 10, 15). This code can correct only 7 errors, which is less than the manipulated/malicious helper nodes that the H-MSR code can correct.

For data reconstruction described in Algorithm 7, according to Theorem 6 and equation (23), the number of errors that H-MSR code can correct is $\tau_{H-MSR-REC} = q\lfloor (q^2 - k_{q-1})/2 \rfloor$. Similarly, we can conclude that for data reconstruction the H-MSR code has better error correction capability than the RS-MSR code under the same code rate.

C. Complexity Discussion

For the complexity of the H-MSR code, we only consider the recovery mode, which needs the most amount of computation. For the H-MSR regeneration, compared with the 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 $\mathcal{O}(q^2)$. Similar to [29], for a replacement node to correct possible errors and regenerate original symbols using Algorithm 4, we can derive that the complexity to regenerate symbols for the H-MSR code is $\mathcal{O}(q^5)$. For the RS-MSR code with the same code dimension as stated in Theorem 13, the complexity would be $\mathcal{O}((q^3)^2) = \mathcal{O}(q^6)$. For the reconstruction, compared with the RS-MSR code, the additional complexity of the H-MSR code for each storage node is also $\mathcal{O}(q^2)$. To correct possible errors and recover original symbols, the computational complexity for DC to reconstruct the data is $\mathcal{O}(q^5)$ for the H-MSR code and $\mathcal{O}(q^6)$ for the RS-MSR code.

X. Conclusion

In this paper, we developed a Hermitian code based minimum storage regeneration (H-MSR) code and a Hermitian code based minimum bandwidth regeneration (H-MBR) code

for distributed storage under the product-matrix framework. Due to the structure of Hermitian code, our proposed codes can significantly improve the performance of the regenerating code under malicious attacks. In particular, these codes can deal with errors beyond the maximum distance separable (MDS) code. Our theoretical analyses demonstrate that the H-MSR/ H-MBR codes have lower complexity than the RS based minimum storage regeneration (RS-MSR) code and the RS based minimum bandwidth regeneration (RS-MBR) 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.

APPENDIX

A. Proof of Lemma 1

Since $\Psi_{0\to d_l-1,l}$ and $\Psi_{1\to d_l,l}$ are full rank matrices, we can get their corresponding inverse matrices. $\hat{\mathbf{x}}_1=\hat{\mathbf{x}}_2$ is equivalent to $\Psi_{0\to d_l-1,l}\hat{\mathbf{x}}_1=\Psi_{0\to d_l-1,l}\hat{\mathbf{x}}_2$.

First, we have

$$\Psi_{0\to d_l-1,l}\hat{\mathbf{x}}_1=[e_0,e_1,\ldots,e_{d_l-1}]^T. \tag{43}$$

Suppose $\Psi^{-1}_{1\to d_l,l}=[\eta_0,\eta_1,\ldots,\eta_{d_l-1}]$, then we have:

$$\Psi_{r,l}\cdot\eta_s=\begin{cases}1, & r=s+1\\ 0, & r\neq s+1\end{cases},\ 1\le r\le d_l, 0\le s\le d_l-1. \tag{44}$$

$$\begin{aligned}\Psi_{0\to d_l-1,l}\hat{\mathbf{x}}_2=&\Psi_{0\to d_l-1,l}\Psi^{-1}_{1\to d_l,l}[e_1,e_2,\ldots,e_{d_l}]^T\\ =&\Psi_{0\to d_l-1,l}[\eta_0,\ldots,\eta_{d_l-1}][e_1,\ldots,e_{d_l}]^T\\ =&[x_{2,0},e_1,\ldots,e_{d_l-1}]^T.\end{aligned} \tag{45}$$

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

$$\Psi_{0,l}=\sum_{r=1}^{d_l}\zeta_r\Psi_{r,l}, \tag{46}$$

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

$$\begin{aligned}x_{2,0}=&\left(\sum_{r=1}^{d_l}\zeta_r\Psi_{r,l}\right)[\eta_0,\eta_1,\ldots,\eta_{d_l-1}][e_1,e_2,\ldots,e_{d_l}]^T\\ =&\sum_{r=1}^{d_l}\zeta_r e_r.\end{aligned} \tag{47}$$

If

$$e_0=\sum_{r=1}^{d_l}\zeta_r e_r, \tag{48}$$

then $\Psi_{0\to d_l-1,l}\hat{\mathbf{x}}_1=\Psi_{0\to d_l-1,l}\hat{\mathbf{x}}_2$ and $\hat{\mathbf{x}}_1=\hat{\mathbf{x}}_2$.

The number of errors corresponds to the number of malicious nodes. When only one element of $e_0,e_1,\ldots,e_{d_l}$ is nonzero, since $\zeta_1,\ldots,\zeta_{d_l}$ are all nonzero, equation (48) will never hold. In this case, the probability that $\hat{\mathbf{x}}_1=\hat{\mathbf{x}}_2$ is 0. When there are more than one nonzero elements, it means there are more than one malicious nodes. If the number of malicious nodes is less than d_l+1, they will not be able to collude to solve the coefficients ζ_r in (46). In this case, the probability that $\hat{\mathbf{x}}_1=\hat{\mathbf{x}}_2$ holds will be $1/q^2$. In summary, the probability that $\hat{\mathbf{x}}_1=\hat{\mathbf{x}}_2$ is at most $1/q^2$. □

B. Proof of Theorem 3

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

$$\begin{aligned}\mathbf{x}_1&=\Psi^{-1}_{0\to d_l-1,l}\left[\tilde{p}_{0,l,t}+e_0,\ldots,\tilde{p}_{d_l-1,l,t}+e_{d_l-1}\right]^T\\ &=\mathbf{x}+\Psi^{-1}_{0\to d_l-1,l}[e_0,e_1,\ldots,e_{d_l-1}]^T\\ &=\mathbf{x}+\hat{\mathbf{x}}_1.\end{aligned} \tag{49}$$

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

$$\mathbf{x}_2=\mathbf{x}+\Psi^{-1}_{1\to d_l,l}[e_1,e_2,\ldots,e_{d_l}]^T=\mathbf{x}+\hat{\mathbf{x}}_2. \tag{50}$$

If $\hat{\mathbf{x}}_1=\hat{\mathbf{x}}_2$, Algorithm 3 will fail to detect the errors. So we will focus on the relationship between $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$. According to Lemma 1, when the number of malicious nodes in the d_l+1 helper nodes is less than d_l+1, the probability that $\hat{\mathbf{x}}_1=\hat{\mathbf{x}}_2$ is at most $1/q^2$. So the probability that $\mathbf{x}_1\neq\mathbf{x}_2$, equivalent to the detection probability, is at least $1-1/q^2$. □

C. Proof of Theorem 5

We arrange this proof as follows. We will first study the requirements for $S_1=S_2,T_1=T_2$ in Algorithm 6 which will lead to the failure of the Algorithm when there are bogus symbols. Then we will study the corresponding failure probabilities depending on different values of λ_i of the matrix Δ defined in section III.

For convenience we write $\mathbf{e}_{i,l,t}$ as $\mathbf{e}_i$ in the proof, $\mathbf{e}_i\in\mathbb{F}_{q^2}^{\alpha_l}$ for $0\le i\le\alpha_l+1$, $R'_{1,l,t}$ as R'_1 and $R'_{2,l,t}$ as R'_2. We also write $\Psi_{0\to\alpha_l,l}=[\Phi_{0\to\alpha_l,l},\Delta_{\alpha_l}\Phi_{0\to\alpha_l,l}]$, where $\Delta_{\alpha_l}=\mathrm{diag}(\lambda_0,\lambda_1,\ldots\lambda_{\alpha_l})$.

Step 1. Derive the requirements: For $R'_1=R_1+E_1$, where $E_1=\begin{bmatrix}\mathbf{e}_0\\ \mathbf{e}_1\\ \vdots\\ \mathbf{e}_{\alpha_l}\end{bmatrix}$ as defined in Algorithm 6, we have:

$$\begin{aligned}&\Phi_{0\to\alpha_l,l}S_1\Phi^T_{0\to\alpha_l,l}+\Delta_{\alpha_l}\Phi_{0\to\alpha_l,l}T_1\Phi^T_{0\to\alpha_l,l}\\ =&R_1\Phi^T_{0\to\alpha_l,l}+E_1\Phi^T_{0\to\alpha_l,l}.\end{aligned} \tag{51}$$

Suppose $C_1=\Phi_{0\to\alpha_l,l}S_1\Phi^T_{0\to\alpha_l,l}, D_1=\Phi_{0\to\alpha_l,l}T_1\Phi^T_{0\to\alpha_l,l}$, we can write equation (51) as:

$$C_1+\Delta_{\alpha_l}D_1=R_1\Phi^T_{0\to\alpha_l,l}+E_1\Phi^T_{0\to\alpha_l,l}=\widehat{R}_1+\widehat{E}_1. \tag{52}$$

It is easy to see that C_1 and D_1 are symmetric, so we have

$$\begin{cases}C_{1,i,j}+\lambda_i D_{1,i,j}=\widehat{R}_{1,i,j}+\widehat{E}_{1,i,j}\\ C_{1,i,j}+\lambda_j D_{1,i,j}=\widehat{R}_{1,j,i}+\widehat{E}_{1,j,i}\end{cases}, \tag{53}$$

where $C_{1,i,j},D_{1,i,j},\widehat{R}_{1,i,j},\widehat{E}_{1,i,j}$ are the elements in the i^{th} row, j^{th} column of $C_1,D_1,\widehat{R}_1,\widehat{E}_1$ respectively. Solve equation (53) for all the i,j ($i\neq j, 0\le i\le\alpha_l, 0\le j\le\alpha_l-1$), we can get the corresponding $C_{1,i,j},D_{1,i,j}$. Because the structure of C_1 and D_1 are the same, we will only focus

on C_1 (corresponding to S_1) in the proof. The calculation for D_1 (corresponding to T_1) is the same.

$$\Phi_{0\to\alpha_l,l}S_1\Phi^T_{0\to\alpha_l,l}=\begin{bmatrix}\Phi_{0,l}\\ \Phi_{1,l}\\ \vdots\\ \Phi_{\alpha_l,l}\end{bmatrix}S_1[\Phi^T_{0,l},\Phi^T_{1,l},\ldots,\Phi^T_{\alpha_l,l}]=C_1. \tag{54}$$

So the elements of the i^{th} row of C_1 (except the element in the diagonal position) can be written as:

$$\begin{aligned}&\Phi_{i,l}S_1[\Phi^T_{0,l},\ldots,\Phi^T_{i-1,l},\Phi^T_{i+1,l},\ldots,\Phi^T_{\alpha_l,l}]\\ =&[C_{1,i,0},\ldots,C_{1,i,i-1},C_{1,i,i+1},\ldots,C_{1,i,\alpha_l}].\end{aligned} \tag{55}$$

Let $\Omega=\begin{bmatrix}\Phi_{0,l}\\ \Phi_{1,l}\\ \vdots\\ \Phi_{\alpha_l-1,l}\end{bmatrix}$, then Ω is an $\alpha_l\times\alpha_l$ full rank matrix, and we can derive S_1 from

$$\Omega S_1=\begin{bmatrix}\Theta_1\\ \vdots\\ \Theta_{\alpha_l}\end{bmatrix}, \tag{56}$$

where $\Theta_i=[C_{1,i-1,0},\ldots,C_{1,i-1,i-2},C_{1,i-1,i},\ldots,C_{1,i-1,\alpha_l}]\cdot[\Phi^T_{0,l},\ldots,\Phi^T_{i-2,l},\Phi^T_{i,l},\ldots,\Phi^T_{\alpha_l,l}]^{-1},\ i=1,\ldots,\alpha_l$.

For $R'_2=R_2+E_2$ in Algorithm 6, we can get ΩS_2 the same way. If $\Omega S_1=\Omega S_2$, Algorithm 6 will fail to detect the errors. This will happen if all the rows of ΩS_1 and ΩS_2 are the same. So we will focus on the i^{th} row of ΩS_1 and ΩS_2.

Step 2. Calculate the failure probabilities: Depending on the values of λ_i, we discuss two cases:

(**a**) If none of the λ_i ($0\le i\le\alpha_l$) equals to 0, we can solve $C_{1,i,j}$ in equation (53):

$$\begin{aligned}C_{1,i,j}=&\frac{\lambda_j\widehat{R}_{1,i,j}-\lambda_i\widehat{R}_{1,j,i}}{\lambda_i\lambda_j}+\frac{\mathbf{e}_i\Phi^T_{j,l}}{\lambda_i}-\frac{\mathbf{e}_j\Phi^T_{i,l}}{\lambda_j}\\ =&N_{1,i,j}+Q_{1,i,j}.\end{aligned} \tag{57}$$

In equation (57), $N_{1,i,j}$ represents the original solution without errors, while $Q_{1,i,j}$ represents the impact of the errors. So the i^{th} row of ΩS_1 can be written as:

$$\begin{aligned}&[C_{1,i,0},\ldots,C_{1,i,i-1},C_{1,i,i+1},\ldots,C_{1,i,\alpha_l}]\cdot\Pi^{-1}_{1,i}\\ &=[N_{1,i,0},\ldots,N_{1,i,i-1},N_{1,i,i+1},\ldots,N_{1,i,\alpha_l}]\cdot\Pi^{-1}_{1,i}\\ &+[Q_{1,i,0},\ldots,Q_{1,i,i-1},Q_{1,i,i+1},\ldots,Q_{1,i,\alpha_l}]\cdot\Pi^{-1}_{1,i}\\ &=\xi_i+\delta_{1,i},\end{aligned} \tag{58}$$

where $\Pi_{1,i}=[\Phi^T_{0,l},\ldots,\Phi^T_{i-1,l},\Phi^T_{i+1,l},\ldots,\Phi^T_{\alpha_l,l}]$. ξ_i corresponds to the part independent of the errors. $\delta_{1,i}$ is the error part and can be further expanded as:

$$\begin{aligned}\delta_{1,i}=&\left[\frac{\mathbf{e}_i\Phi^T_{0,l}}{\lambda_i},\ldots,\frac{\mathbf{e}_i\Phi^T_{i-1,l}}{\lambda_i},\frac{\mathbf{e}_i\Phi^T_{i+1,l}}{\lambda_i},\ldots,\frac{\mathbf{e}_i\Phi^T_{\alpha_l,l}}{\lambda_i}\right]\Pi^{-1}_{1,i}\\ &-\left[\frac{\mathbf{e}_0\Phi^T_{i,l}}{\lambda_0},\ldots,\frac{\mathbf{e}_{i-1}\Phi^T_{i,l}}{\lambda_{i-1}},\frac{\mathbf{e}_{i+1}\Phi^T_{i,l}}{\lambda_{i+1}},\ldots,\frac{\mathbf{e}_{\alpha_l}\Phi^T_{i,l}}{\lambda_{\alpha_l}}\right]\Pi^{-1}_{1,i}.\end{aligned} \tag{59}$$

The first part of equation (59) can be reduced as follows:

$$\begin{aligned}&\left[\frac{\mathbf{e}_i\Phi^T_{0,l}}{\lambda_i},\ldots,\frac{\mathbf{e}_i\Phi^T_{i-1,l}}{\lambda_i},\frac{\mathbf{e}_i\Phi^T_{i+1,l}}{\lambda_i},\ldots,\frac{\mathbf{e}_i\Phi^T_{\alpha_l,l}}{\lambda_i}\right]\Pi^{-1}_{1,i}\\ =&\frac{\mathbf{e}_i}{\lambda_i}\left[\Phi^T_{0,l},\ldots,\Phi^T_{i-1,l},\Phi^T_{i+1,l},\ldots,\Phi^T_{\alpha_l,l}\right]\Pi^{-1}_{1,i}\\ =&\frac{\mathbf{e}_i}{\lambda_i}.\end{aligned} \tag{60}$$

So we have:

$$\begin{aligned}\delta_{1,i}=&\frac{\mathbf{e}_i\Phi^T_{i,l}}{\lambda_i}-\left[\frac{\mathbf{e}_0\Phi^T_{i,l}}{\lambda_0},\ldots,\frac{\mathbf{e}_{i-1}\Phi^T_{i,l}}{\lambda_{i-1}},\frac{\mathbf{e}_{i+1}\Phi^T_{i,l}}{\lambda_{i+1}},\right.\\ &\left.\ldots,\frac{\mathbf{e}_{\alpha_l}\Phi^T_{i,l}}{\lambda_{\alpha_l}}\right]\Pi^{-1}_{1,i}\\ =&\frac{\mathbf{e}_i\Phi^T_{i,l}}{\lambda_i}-\rho_{1,i}.\end{aligned} \tag{61}$$

For $R'_2=R_2+E_2$ in Algorithm 6 where $E_2=\begin{bmatrix}\mathbf{e}_0\\ \vdots\\ \mathbf{e}_{\alpha_l-1}\\ \mathbf{e}_{\alpha_l+1}\end{bmatrix}$, we can derive $C_{2,i,j}$, then ΩS_2 the same way. The i^{th} row of ΩS_2 can be written as:

$$\xi_i+\delta_{2,i}=\xi_i+\frac{\mathbf{e}_i}{\lambda_i}-\rho_{2,i}, \tag{62}$$

where $\rho_{2,i}=\left[\frac{\mathbf{e}_0\Phi^T_{i,l}}{\lambda_0},\ldots,\frac{\mathbf{e}_{i-1}\Phi^T_{i,l}}{\lambda_{i-1}},\frac{\mathbf{e}_{i+1}\Phi^T_{i,l}}{\lambda_{i+1}},\ldots,\frac{\mathbf{e}_{\alpha_l-1}\Phi^T_{i,l}}{\lambda_{\alpha_l-1}},\frac{\mathbf{e}_{\alpha_l+1}\Phi^T_{i,l}}{\lambda_{\alpha_l+1}}\right]\Pi^{-1}_{2,i}$, $\Pi_{2,i}=[\Phi^T_{0,l},\ldots,\Phi^T_{i-1,l},\Phi^T_{i+1,l},\ldots,\Phi^T_{\alpha_l-1,l},\Phi^T_{\alpha_l+1,l}]$.

Because $\Pi_{1,i}$ is a full rank matrix, $\rho_{1,i}=\rho_{2,i}$ is equivalent to $\rho_{1,i}\Pi_{1,i}=\rho_{2,i}\Pi_{1,i}$. Similar to the proof of Lemma 1, suppose $\Pi^{-1}_{2,i}=\begin{bmatrix}\eta_0\\ \vdots\\ \eta_{\alpha_l-1}\\ \eta_{\alpha_l+1}\end{bmatrix}$, we have $\eta_s\Phi^T_{r,l}=\begin{cases}1 & r=s\\ 0 & r\ne s\end{cases}$. So

$$\begin{aligned}\rho_{1,i}\Pi_{1,i}=&\left[\ldots,\frac{\mathbf{e}_{i-1}\Phi^T_{i,l}}{\lambda_{i-1}},\frac{\mathbf{e}_{i+1}\Phi^T_{i,l}}{\lambda_{i+1}},\right.\\ &\left.\ldots,\frac{\mathbf{e}_{\alpha_l-1}\Phi^T_{i,l}}{\lambda_{\alpha_l-1}},\frac{\mathbf{e}_{\alpha_l}\Phi^T_{i,l}}{\lambda_{\alpha_l}}\right],\end{aligned} \tag{63}$$

$$\begin{aligned}\rho_{2,i}\Pi_{1,i}=&\left[\ldots,\frac{\mathbf{e}_{i-1}\Phi^T_{i,l}}{\lambda_{i-1}},\frac{\mathbf{e}_{i+1}\Phi^T_{i,l}}{\lambda_{i+1}},\right.\\ &\left.\ldots,\frac{\mathbf{e}_{\alpha_l-1}\Phi^T_{i,l}}{\lambda_{\alpha_l-1}},x_{2,\alpha_l}\right].\end{aligned} \tag{64}$$

Because $\Phi^T_{0,l},\ldots,\Phi^T_{i-1,l},\Phi^T_{i+1,l},\ldots,\Phi^T_{\alpha_l-1,l},\Phi^T_{\alpha_l+1,l}$ are linearly independent, they can be viewed as a set of bases of the α_l dimensional linear space. So we have

$$\Phi^T_{\alpha_l,l}=\sum_{r=0,r\ne i,\alpha_l}^{\alpha_l+1}\zeta_r\Phi^T_{r,l}. \tag{65}$$

Thus

$$x_{2,\alpha_l} = \left[\ldots, \frac{\mathbf{e}_{i-1}\Phi_{i,l}^T}{\lambda_{i-1}}, \frac{\mathbf{e}_{i+1}\Phi_{i,l}^T}{\lambda_{i+1}}, \ldots, \frac{\mathbf{e}_{\alpha_l-1}\Phi_{i,l}^T}{\lambda_{\alpha_l-1}}, \frac{\mathbf{e}_{\alpha_l+1}\Phi_{i,l}^T}{\lambda_{\alpha_l+1}}\right] \cdot \Pi_{2,i}^{-1} \cdot \left(\sum_{r=0,r\neq i,\alpha_l}^{\alpha_l+1} \zeta_r \cdot \Phi_{r,l}^T\right) = \left(\sum_{r=0,r\neq i,\alpha_l}^{\alpha_l+1} \zeta_r \cdot \frac{\mathbf{e}_r\Phi_{i,l}^T}{\lambda_r}\right). \quad (66)$$

If

$$\frac{\mathbf{e}_{\alpha_l}\Phi_{i,l}^T}{\lambda_{\alpha_l}} = \sum_{r=0,r\neq i,\alpha_l}^{\alpha_l+1} \zeta_r \frac{\mathbf{e}_r\Phi_{i,l}^T}{\lambda_r} \ (0 \le i \le \alpha_l - 1), \quad (67)$$

$\rho_{1,i}$ and $\rho_{2,i}$ will be equal, so are ΩS_1 and ΩS_2. Therefore, Algorithm 6 will fail.

For the error $\mathbf{e}_i$ $(0 \le i \le \alpha_l + 1)$, the following equation holds:

$$\mathbf{e}_i[\Phi_{0,l}^T, \Phi_{1,l}^T, \ldots, \Phi_{\alpha_l-1,l}^T] = [\hat{e}_{i,0}, \hat{e}_{i,1}, \ldots, \hat{e}_{i,\alpha_l-1}] = \hat{\mathbf{e}}_i. \quad (68)$$

Because $[\Phi_{0,l}^T, \Phi_{1,l}^T, \ldots, \Phi_{\alpha_l-1,l}^T]$ is a full rank matrix, there is a one-to-one mapping between $\mathbf{e}_i$ and $\hat{\mathbf{e}}_i$. Equation (67) can be written as:

$$\frac{\hat{e}_{\alpha_l,i}}{\lambda_{\alpha_l}} = \sum_{r=0,r\neq i,\alpha_l}^{\alpha_l+1} \zeta_r \frac{\hat{e}_{r,i}}{\lambda_r} \ (0 \le i \le \alpha_l - 1). \quad (69)$$

When the number of malicious nodes in the $k_l + 1$ nodes is less than $k_l + 1$, the malicious nodes can collude to satisfy equation (69) for at most one particular i. So the probability that equation (69) holds is $1/q^2$ for at least $\alpha_l - 1$ out of α_l $i's$ between 0 and $\alpha_l - 1$. If we consider equation (69) for all the $i's$ simultaneously, the probability will be at most $(1/q^2)^{\alpha_l-1}$. As discussed before, the probability for $T_1 = T_2$ will be at most $(1/q^2)^{\alpha_l-1}$. In this case, the detection probability is at least $1 - (1/q^2)^{2(\alpha_l-1)}$.

(**b**) If one of the λ_i $(0 \le i \le \alpha_l)$ equals to 0, we can assume $\lambda_0 = 0$ without loss of generality. When $i = 0$, the solution for equation (53) is:

$$C_{1,0,j} = \widehat{R}_{1,0,j} + \mathbf{e}_0\Phi_{j,l}^T = N_{1,0,j} + Q_{1,0,j}. \quad (70)$$

Similar to equations (58), (59) and (60), we have $\delta_{1,0} = \mathbf{e}_0$. For $R_2' = R_2 + E_2$, it is easy to see that $\delta_{2,0} = \mathbf{e}_0$. So the first rows of ΩS_1 and ΩS_2 are the same no matter what the error vector $\mathbf{e}_0$ is.

When $i > 0, j = 0$, the solution for equation (53) is:

$$C_{1,i,0} = \widehat{R}_{1,i,0} + \mathbf{0} \cdot \Phi_{0,l}^T + \mathbf{e}_0\Phi_{i,l}^T = N_{1,i,0} + Q_{1,i,0}, \quad (71)$$

where $\mathbf{0}$ is a zero row vector. When $i > 0, j > 0$, the solution has the same expression as equation (57). In this case, for the i^{th} $(i > 0)$ row of ΩS_1, equation (59) can be written as:

$$\delta_{1,i} = \left[\mathbf{0}, \ldots, \frac{\mathbf{e}_i\Phi_{i-1,l}^T}{\lambda_i}, \frac{\mathbf{e}_i\Phi_{i+1,l}^T}{\lambda_i}, \ldots, \frac{\mathbf{e}_i\Phi_{\alpha_l,l}^T}{\lambda_i}\right] \Pi_{1,i}^{-1} - \left[-\mathbf{e}_0\Phi_{i,l}^T, \ldots, \frac{\mathbf{e}_{i-1}\Phi_{i,l}^T}{\lambda_{i-1}}, \frac{\mathbf{e}_{i+1}\Phi_{i,l}^T}{\lambda_{i+1}}, \ldots, \frac{\mathbf{e}_{\alpha_l}\Phi_{i,l}^T}{\lambda_{\alpha_l}}\right] \Pi_{1,i}^{-1}. \quad (72)$$

The first part of equation (72) can be divided into two parts:

$$\left[\frac{\mathbf{e}_i\Phi_{0,l}^T}{\lambda_i}, \ldots, \frac{\mathbf{e}_i\Phi_{i-1,l}^T}{\lambda_i}, \frac{\mathbf{e}_i\Phi_{i+1,l}^T}{\lambda_i}, \ldots, \frac{\mathbf{e}_i\Phi_{\alpha_l,l}^T}{\lambda_i}\right] \Pi_{1,i}^{-1} - \left[\frac{\mathbf{e}_i\Phi_{0,l}^T}{\lambda_i}, \mathbf{0}, \ldots, \mathbf{0}\right] \Pi_{1,i}^{-1} = \frac{\mathbf{e}_i}{\lambda_i} - \frac{\mathbf{e}_i}{\lambda_i}[\Phi_{0,l}^T, \mathbf{0}, \ldots, \mathbf{0}]\Pi_{1,i}^{-1}. \quad (73)$$

So equation (72) can be further written as:

$$\delta_{1,i} = \frac{\mathbf{e}_i}{\lambda_i} - \left[\frac{\mathbf{e}_i\Phi_{0,l}^T}{\lambda_i} - \mathbf{e}_0\Phi_{i,l}^T, \ldots, \frac{\mathbf{e}_{i-1}\Phi_{i,l}^T}{\lambda_{i-1}}, \frac{\mathbf{e}_{i+1}\Phi_{i,l}^T}{\lambda_{i+1}}, \ldots, \frac{\mathbf{e}_{\alpha_l}\Phi_{i,l}^T}{\lambda_{\alpha_l}}\right] \Pi_{1,i}^{-1} = \frac{\mathbf{e}_i}{\lambda_i} - \rho_{1,i}. \quad (74)$$

By employing the same derivation in case (a), for $1 \le i \le \alpha_l - 1$, $\rho_{1,i}$ and $\rho_{2,i}$ will be equal if

$$\frac{\mathbf{e}_{\alpha_l}\Phi_{i,l}^T}{\lambda_{\alpha_l}} = \sum_{r=1,r\neq i,\alpha_l}^{\alpha_l+1} \zeta_r \frac{\mathbf{e}_r\Phi_{i,l}^T}{\lambda_r} - \zeta_0\mathbf{e}_0\Phi_{i,l}^T + \zeta_0\frac{\mathbf{e}_i\Phi_{0,l}^T}{\lambda_i}, \quad (75)$$

$$\frac{\hat{e}_{\alpha_l,i}}{\lambda_{\alpha_l}} = \sum_{r=1,r\neq i,\alpha_l}^{\alpha_l+1} \zeta_r \frac{\hat{e}_{r,i}}{\lambda_r} - \zeta_0\hat{e}_{0,i} + \zeta_0\frac{\hat{e}_{i,0}}{\lambda_i}. \quad (76)$$

When the number of malicious nodes in the $k_l + 1$ nodes is less than $k_l + 1$, similar to case (a), the probability that equation (76) holds is $1/q^2$ for at least $\alpha_l - 2$ out of $\alpha_l - 1$ $i's$ between 1 and $\alpha_l - 1$. If we consider equation (76) for all the $i's$ simultaneously, the probability will be at most $(1/q^2)^{\alpha_l-2}$. Here the probability for $T_1 = T_2$ will be at most $(1/q^2)^{\alpha_l-2}$. In this case, the detection probability is $1 - (1/q^2)^{2(\alpha_l-2)}$.

Combining both cases, the detection probability is at least $1 - (1/q^2)^{2(\alpha_l-2)}$. □

D. Proof of Theorem 9

Similar to the proof of Theorem 3, we can write

$$\mathbf{x}_1 = \mathbf{x} + \Phi_{0\to d_l-1,l}^{-1}[e_0, \ldots, e_{d_l-1}]^T = \mathbf{x} + \hat{\mathbf{x}}_1, \quad (77)$$

$$\mathbf{x}_2 = \mathbf{x} + \Phi_{1\to d_l,l}^{-1}[e_1, \ldots, e_{d_l}]^T = \mathbf{x} + \hat{\mathbf{x}}_2. \quad (78)$$

Since $\Phi_{0\to d_l-1,l}, \Phi_{1\to d_l,l}$ are full rank matrices like the matrices $\Psi_{0\to d_l-1,l}, \Psi_{1\to d_l,l}$ in the proof of Lemma 1 and any d_l vectors out of $\Phi_{0,l}, \Phi_{1,l}, \ldots, \Phi_{d_l,l}$ are linearly independent,

the rest of this proof is similar to that of Lemma 1. When the number of malicious nodes in the d_l+1 helper nodes is less than d_l+1, the probability for $\hat{\mathbf{x}}_1=\hat{\mathbf{x}}_2$ is at most $1/q^2$. Therefore, the detection probability is at least $1-1/q^2$. □

E. Proof of Theorem 11

For convenience, we write $\mathbf{e}_{i,l,t}$ as $\mathbf{e}_i$ in the proof, $\mathbf{e}_i \in [\mathbb{F}_{q^2}]^{\alpha_l}$ for $0 \le i \le k_l$, $R'_{1,l,t}$ as R'_1 and $R'_{2,l,t}$ as R'_2. In Algorithm 13, ${R_1}' = R_1 + Q_1$ where $Q_1 = \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_{k_l-1} \end{bmatrix}$. Let $\Phi_{0\to k_l-1,l} = [\Phi^{(1)}_{0\to k_l-1,l}, \Phi^{(2)}_{0\to k_l-1,l}]$, $R_1 = [R_{1,1}, R_{1,2}]$ and $Q_1 = [Q_{1,1}, Q_{1,2}]$, where $\Phi^{(1)}_{0\to k_l-1,l}, R_{1,1}, Q_{1,1}$ are $k_l \times k_l$ submatrices and $\Phi^{(2)}_{0\to k_l-1,l}, R_{1,2}, Q_{1,2}$ are $k_l \times (\alpha_l - k_l)$ submatrices.

According to equation (31), we have

$$\begin{aligned}\Phi_{0\to k_l-1,l}M_1 &= [\Phi^{(1)}_{0\to k_l-1,l}S_1 + \Phi^{(2)}_{0\to k_l-1,l}T_1^T, \Phi^{(1)}_{0\to k_l-1,l}T_1] \\ &= [R_{1,1}+Q_{1,1}, R_{1,2}+Q_{1,2}]. \end{aligned} \tag{79}$$

Since $\Phi^{(1)}_{0\to k_l-1,l}$ is a submatrix of a Vandermonde matrix, it is a full rank matrix. So we have

$$T_1 = [\Phi^{(1)}_{0\to k_l-1,l}]^{-1}R_{1,2} + [\Phi^{(1)}_{0\to k_l-1,l}]^{-1}Q_{1,2} = T + \widehat{T}_1, \tag{80}$$

$$\begin{aligned} S_1 &= [\Phi^{(1)}_{0\to k_l-1,l}]^{-1}(R_{1,1}+Q_{1,1}-\Phi^{(2)}_{0\to k_l-1,l}T_1^T) \\ &= [\Phi^{(1)}_{0\to k_l-1,l}]^{-1}(R_{1,1}-\Phi^{(2)}_{0\to k_l-1,l}T^T) \\ &\quad + [\Phi^{(1)}_{0\to k_l-1,l}]^{-1}(Q_{1,1}-\Phi^{(2)}_{0\to k_l-1,l}\widehat{T}_1^T) \\ &= S + [\Phi^{(1)}_{0\to k_l-1,l}]^{-1}(Q_{1,1}-\Phi^{(2)}_{0\to k_l-1,l}\widehat{T}_1^T) = S+\widehat{S}_1. \end{aligned} \tag{81}$$

For $R'_2 = R_2 + Q_2$ in Algorithm 13, let $R_2 = [R_{2,1}, R_{2,2}], Q_2 = [Q_{2,1}, Q_{2,2}]$ and $\Phi_{1\to k_l,l} = [\Phi^{(1)}_{1\to k_l,l}, \Phi^{(2)}_{1\to k_l,l}]$, where $R_{2,1}, Q_{2,1}, \Phi^{(1)}_{1\to k_l,l}$ are $k_l \times k_l$ submatrices and $R_{2,2}, Q_{2,2}, \Phi^{(2)}_{1\to k_l,l}$ are $k_l \times (\alpha_l - k_l)$ submatrices. Similarly, we have

$$T_2 = [\Phi^{(1)}_{1\to k_l,l}]^{-1}R_{2,2} + [\Phi^{(1)}_{1\to k_l,l}]^{-1}Q_{2,2} = T + \widehat{T}_2, \tag{82}$$

$$S_2 = S + [\Phi^{(1)}_{1\to k_l,l}]^{-1}(Q_{2,1} - \Phi^{(2)}_{1\to k_l,l}\widehat{T}_2^T) = S + \widehat{S}_2. \tag{83}$$

If $\widehat{T}_1 = \widehat{T}_2$ and $\widehat{S}_1 = \widehat{S}_2$, Algorithm 13 will fail to detect the bogus symbols. So we will focus on $\widehat{T}_1, \widehat{T}_2$ and $\widehat{S}_1, \widehat{S}_2$.

Suppose $\Pi_{1,j} = [e_0, \ldots, e_{k_l-1}]^T, \Pi_{2,j} = [e_1, \ldots, e_{k_l}]^T$ are the $j^{th}, 1 \le j \le \alpha_l - k_l$, columns of $Q_{1,2}$ and $Q_{2,2}$ respectively, where $e_i \in \mathbb{F}_{q^2}$. Since $\Phi^{(1)}_{0\to k_l-1,l}$ and $\Phi^{(1)}_{1\to k_l,l}$ are Vandermonde matrices and have the same relationship as that of between $\Psi_{0\to d_l-1,l}$ and $\Psi_{1\to d_l,l}$, similar to the proof of Lemma 1, we can prove that when the number of malicious nodes in the k_l+1 nodes is less than k_l+1, the probability of $[\Phi^{(1)}_{0\to k_l-1,l}]^{-1}\Pi_{1,j} = [\Phi^{(1)}_{1\to k_l,l}]^{-1}\Pi_{2,j}$ is at most $1/q^2$. Thus the probability for $\hat{T}_1 = \hat{T}_2$ is at most $1/q^{2(\alpha_l-k_l)}$. Through the same procedure, we can derive that the probability of $\hat{S}_1 = \hat{S}_2$ is at most $1/q^{2k_l}$. The probability for both $\hat{S}_1 = \hat{S}_2$ and $\hat{T}_1 = \hat{T}_2$ is at most $1/q^{2\alpha_l}$. So the detection probability is at least $1-1/q^{2\alpha_l}$. □

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor Prof. A. G. Dimakis and the anonymous reviewers for their careful reading of their article and their many insightful comments and suggestions.

REFERENCES

[1] S. Rhea *et al.*, "Maintenance-free global data storage," *IEEE Internet Comput.*, vol. 5, no. 5, pp. 40–49, Sep./Oct. 2001.
[2] 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.
[3] 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.
[4] 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.
[5] 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.
[6] D. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, "Repair optimal erasure codes through Hadamard designs," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3021–3037, May 2013.
[7] 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.
[8] C. Tian and T. Liu, "Multilevel diversity coding with regeneration," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 4833–4847, Sep. 2016.
[9] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. 45th Annu. Allerton Conf. Control, Comput., Commun.*, Sep. 2007, pp. 1–5.
[10] 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.*, Jun. 2009, pp. 376–384.
[11] K. W. Shum, "Cooperative regenerating codes for distributed storage systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.
[12] 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.
[13] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2009, pp. 2276–2280.
[14] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2134–2158, Apr. 2012.
[15] 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 Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.
[16] V. Guruswami and M. Wootters, "Repairing Reed–Solomon codes," 2015, *arXiv:1509.04764*. [Online]. Available: http://arxiv.org/abs/1509.04764
[17] 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.
[18] M. Abd-El-Malek, G. R. Ganger, M. K. Reiter, J. J. Wylie, and G. R. Goodson, "Lazy verification in fault-tolerant distributed storage systems," in *Proc. 24th IEEE Symp. Reliable Distrib. Syst. (SRDS)*, Oct. 2005, pp. 179–190.
[19] C. Cachin and S. Tessaro, "Optimal resilience for erasure-coded Byzantine distributed storage," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2006, pp. 115–124.
[20] Y. S. Han, R. Zheng, and W. Ho Mow, "Exact regenerating codes for byzantine fault tolerance in distributed storage," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2498–2506.
[21] H. C. H. Chen and P. P. C. Lee, "Enabling data integrity protection in regenerating-coding-based cloud storage," in *Proc. IEEE 31st Symp. Reliable Distrib. Syst.*, Oct. 2012, pp. 51–60.
[22] F. Oggier and A. Datta, "Byzantine fault tolerance of regenerating codes," in *Proc. IEEE Int. Conf. Peer-to-Peer Comput.*, Aug. 2011, pp. 112–121.

[23] 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.
[24] K. V. Rashmi, N. B. Shah, K. Ramchandran, and P. V. Kumar, "Regenerating codes for errors and erasures in distributed storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 1202–1206.
[25] 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.
[26] K. Huang, U. Parampalli, and M. Xian, "On secrecy capacity of minimum storage regenerating codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 3, pp. 1510–1524, Mar. 2017.
[27] A. S. Rawat, "Secrecy capacity of minimum storage regenerating codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 1406–1410.
[28] S. Shao, T. Liu, C. Tian, and C. Shen, "On the tradeoff region of secure exact-repair regenerating codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7253–7266, Nov. 2017.
[29] 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.

Jian Li received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, China, in 2005 and 2008, respectively, and the Ph.D. degree in electrical engineering from Michigan State University, East Lansing, in 2015. He is currently an Associate Professor with 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.

Tongtong Li (Senior Member, IEEE) received the Ph.D. degree in math from Sun Yat-sen University, China, in 1995, and the Ph.D. degree in electrical engineering from Auburn University, USA, in 2000. From 2000 to 2002, she was with Bell Labs, and had been working on the design and implementation of 3G and 4G systems. Since 2002, she has been with Michigan State University, where she is currently an Associate Professor. Her research interests fall into the areas of wireless and wired communications, wireless security, information theory, statistical signal processing, and computational brain analysis. She was a recipient of the National Science Foundation (NSF) CAREER Award in 2008, for her research on efficient and reliable wireless communications. She served as an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS from 2007 to 2009 and the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2012 to 2016, and an Editorial Board Member of *EURASIP Journal on Wireless Communications and Networking* from 2004 to 2011.

Jian Ren (Senior Member, IEEE) received the B.S. and M.S. degrees in mathematics from Shaanxi Normal University and the Ph.D. degree in EE from Xidian University, China. He is currently an Associate Professor with the Department of ECE, Michigan State University. His current research interests include network security, cloud computing security, privacy-preserving communications, distributed network storage, and Internet of Things. He was a recipient of the U.S. National Science Foundation Faculty Early Career Development (CAREER) Award in 2009. He served as the TPC Chair of IEEE ICNC'17, a General Chair of ICNC'18, and an Executive Chair of ICNC'19 and ICNC'20. He currently serves as an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE INTERNET OF THINGS JOURNAL, *ACM Transactions on Sensor Networks (TOSN),* and a Senior Associate Editor for *IET Communications*.