Proxy-Assisted Regenerating Codes With Uncoded Repair for Distributed Storage Systems

Yuchong Hu, *Member, IEEE*, Patrick P. C. Lee, *Senior Member, IEEE*, Kenneth W. Shum, *Senior Member, IEEE*, and Pan Zhou, *Member, IEEE*

***Abstract*— Distributed storage systems can store data with erasure coding to maintain data availability with low storage redundancy. One class of erasure coding is based on *regenerating codes*, which provably minimize the amount of data transferred for failure repair and realize the optimal tradeoff between the storage redundancy and the amount of traffic transferred for repair. Typical regenerating codes often require surviving storage nodes to encode their stored data for repair. In this paper, we study a framework called proxy-assisted regeneration, which offloads the repair process to a centralized proxy. We extend the previous applied work on proxy-assisted regeneration by providing theoretical validation. Specifically, we study a special class of regenerating codes called *proxy-assisted minimum storage regenerating (PMSR)* codes, which enable *uncoded repair* without the need of encoding in surviving nodes, while preserving the minimum storage redundancy and minimum amount of traffic transferred for repair. We formally prove the existence of PMSR codes for two configurations: 1) repairing single-node failures under double fault tolerance and 2) repairing double-node failures under triple fault tolerance. We also provide a semi-deterministic PMSR code construction for repairing single-node failures under double fault tolerance.**

***Index Terms*— Distributed storage, regenerating codes, network coding.**

I. INTRODUCTION

WE HAVE witnessed the wide deployment of storage systems in Internet-wide distributed settings, such as peer-to-peer storage (e.g., [3], [10], [24], [48]), data-center storage (e.g., [7], [15]), or multi-cloud storage (e.g., [1], [9]). Such storage systems stripe data over multiple *nodes* that span across a networked environment, such that each node can represent a storage server (for peer-to-peer and data-center storage) or a cloud storage provider (for multi-cloud storage). For data availability, a storage system must keep user data for a long period of time and allow users to access their data if necessary. However, storage systems are prone to node failures [14], [15]; there are even real-world cases suggesting that cloud storage providers experience failures that incur permanent data loss [9]. It is thus important for a storage system to ensure data availability in practical deployment.

One way to ensure data availability is to store redundant data over multiple nodes. Redundancy can be generated via *erasure coding*, which incurs much less redundancy overhead than replication under the same fault tolerance [35], [49]. *Maximum distance separable (MDS)* codes are one popular family of erasure coding. An MDS code can be defined by two parameters n and k (where $k < n$). It first divides an original file of size M into k fragments of size M/k each, and then encodes them into n fragments also of size M/k each. It has the property (which we call the *MDS property*) that any k out of n encoded fragments suffice to recover the original file, while the storage redundancy is shown to be minimum. By storing the n encoded fragments over n nodes, a storage system can tolerate at most $n - k$ node failures. Examples of MDS codes are Reed-Solomon codes [34] and Cauchy Reed-Solomon codes [5].

When a node fails, it is necessary to recover the lost data of the failed node to preserve fault tolerance. Since bandwidth resources are limited in a distributed networked environment, it is critical to minimize the bandwidth usage in the repair process and hence improve the overall repair performance. *Regenerating codes* [13] are one special class of erasure coding that minimizes the *repair bandwidth*, defined as the amount of data traffic transferred in the repair process. The repair process of regenerating codes builds on network coding [2], such that to repair a failed node, existing surviving (i.e., non-failed) nodes encode their own stored data and send the encoded data to the new node, which then reconstructs the lost data. It is proven that regenerating codes achieve the optimal trade-off between storage cost and repair bandwidth, and incur much less bandwidth than conventional repair under the same storage redundancy and fault tolerance settings.

Typical regenerating code constructions (e.g., [6], [12], [32], [39], [41], [42], [52]) require storage nodes encode stored data for repair. However, this may not be feasible for some storage devices (e.g., tapes, optical disks, raw disks) or

Manuscript received July 20, 2016; revised January 29, 2017; accepted April 27, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61502191, Grant 61401169, and Grant 61502190, in part by the Hubei Provincial Natural Science Foundation of China under Grant 2016CFB226, in part by the University Grants Committee of Hong Kong under Grant AoE/E-02/08, and in part by the Research Grants Council of Hong Kong under Grant CRF-C7036-15. This paper was presented at the 2013 INFOCOM.

Y. Hu is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yuchonghu@hust.edu.cn).

P. P. C. Lee is with The Chinese University of Hong Kong, Hong Kong (e-mail: pclee@cse.cuhk.edu.hk).

K. W. Shum is with the Institute of Network Coding, The Chinese University of Hong Kong, Hong Kong (e-mail: wkshum@inc.cuhk.edu.hk).

P. Zhou is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: panzhou@hust.edu.cn).

Communicated by P. Sadeghi, Associate Editor for Coding Techniques.

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

Digital Object Identifier 10.1109/TIT.2017.2705092

thin-cloud storage services [44] (e.g., Amazon S3) that merely provide the basic I/O functionalities without being equipped with any computational capability for the encoding operation. Some studies [36], [38], [40] use a *repair-by-transfer* approach to eliminate the need of encoding in surviving nodes for repair, but the new node still needs to be equipped with the computational capability to decode the lost data.

This motivates us to study *proxy-assisted regeneration*, which offloads the repair process to a centralized proxy to realize *uncoded repair*, meaning that all storage nodes (including the surviving nodes and new nodes) no longer need to perform encoding or decoding operations for repair; in the meantime, we still achieve the same optimality as regenerating codes in terms of minimizing the amount of data transferred for repair. Our previous work NCCloud [9] provides a starting point to this problem from an applied perspective, in which it runs as a proxy and implements *functional minimum storage regenerating (FMSR)* codes to realize uncoded repair for multi-cloud storage. Through analysis and experiments, NCCloud can achieve up to 50% of repair bandwidth saving compared to double-fault tolerant RAID-6 codes [22] for a single-node failure repair. Previous studies (e.g., RACS [1], BlueSky [45], etc.) also consider a proxy-based design to simplify the cloud storage deployment, and the design can be feasibly generalized for a distributed proxy to address any single-point-of-failure concern [1]. In addition, for traditional RAID (Redundant Arrays of Independent Disks) [28], since raw disks do not have computational capability for the encoding/decoding operations, the RAID controller can be viewed as a centralized proxy; this idea has also been realized in previous studies (e.g., [21], [50]) to perform RAID-like encoding/decoding operations in distributed storage systems. Thus, we believe that proxy-assisted regeneration is of practical interest for real-world distributed storage systems.

Note that FMSR codes are designed as *non-systematic* codes as they do not keep the original uncoded data; instead, they store only linear combinations of original data called *parity chunks*. Each round of repair regenerates new parity chunks for the new nodes and ensures that the fault tolerance level is maintained. A trade-off of FMSR codes is that the whole encoded file must be decoded first if parts of a file are accessed. Nevertheless, FMSR codes are more suited to persistence-critical applications rather than performance-critical ones. One example is long-term archival applications, in which data backups are rarely read and it is common to restore the whole file rather than the partial content of the file.

While proxy-assisted regeneration has been experimented in cloud testbed environments, the original NCCloud work [9] does not provide any formal theoretical analysis to prove whether FMSR codes exist and whether they can be deterministically constructed. In particular, given that each round of repair regenerates new parity chunks, there is no guarantee that the MDS property is maintained after multiple rounds of repair. In addition, NCCloud only focuses on single node failures, yet concurrent multi-node failures are also commonly found in practical distributed storage systems and lead to data loss (e.g., power-on restart) [8]. Thus, the key motivation of this work is to provide theoretical foundation for the practicality of proxy-assisted regeneration.

A. Contributions

In this paper, we conduct formal theoretical analysis on the optimality of proxy-assisted regeneration. We propose a family of *proxy-assisted minimal storage regenerating (PMSR) codes*, which support optimal uncoded repair for both single-node and multi-node failures, by minimizing the total amount of data read from all surviving nodes for failure repair. To summarize, this paper makes the following contributions.

- We propose a family of PMSR codes with uncoded repair and polynomial subpacketization for a general number of node failures (including single-node and multi-node failures). We formally establish a necessary condition of PMSR codes in terms of the lower bound of the amount of data read from each surviving node.
- We formally prove the existence of PMSR codes for two configurations: (i) repairing single-node failures under double fault tolerance and (ii) repairing double-node failures under triple fault tolerance.
- We provide a *semi-deterministic* PMSR code construction for repairing single-node failures under double fault tolerance, such that the chunk selection from each surviving node is deterministic and the encoding coefficients used to regenerate new chunks can be determined based on a set of rules rather than completely at random. We show that our semi-deterministic PMSR code construction significantly speeds up the repair time compared to the non-deterministic approach in NCCloud [9].

B. Paper Organization

The rest of the paper proceeds as follows. Section II reviews related work. Section III-A states the proxy-assisted regeneration problem. Section III gives an information flow model of the the proxy-assisted regeneration problem and derives a bound of the minimum repair bandwidth. Section IV characterizes the system model of PMSR codes. Section V formally proves the existence of PMSR codes. Section VI provides a family of deterministic PMSR code construction. Section VII presents evaluation results. Section VIII concludes the paper.

II. Background and Related Work

Dimakis *et al.* [13] first propose *regenerating codes* based on *network coding* [2] for distributed storage systems, and prove that when repairing a single failed storage node, regenerating codes achieve the optimal trade-off between storage cost and repair bandwidth. There are two extreme points on the optimal trade-off spectrum: *minimum storage regenerating (MSR)* codes, which incur the minimum storage redundancy as MDS codes, and *minimum bandwidth regenerating (MBR)* codes, which incur higher storage redundancy than MDS codes to further minimize the repair bandwidth. In this work, we focus on the MSR codes.

Previous studies (e.g., [13], [18], [51]) show that the MSR point is achievable through the construction of *functional-repair* MSR (FMSR) codes, meaning that the repaired data may not be the same as the lost data, while the same fault tolerance is maintained. However, the corresponding coding schemes [13], [18], [51] are based on random linear codes and do not provide explicit construction. A number of studies (e.g., [6], [12], [32], [39], [42], [52]) have proposed *exact-repair* MSR (EMSR) codes, in which the repaired data is identical to the originally lost data.

Most regenerating codes require surviving storage nodes encode stored data for repair, implying that storage nodes need to possess the computational capability. We examine code constructions that achieve *uncoded repair*, meaning that the encoding requirement of surviving storage nodes is eliminated. It is known that we can construct MBR codes with uncoded repair [33], [36], [38]. For EMSR codes, Tamo *et al.* [43] propose codes that have the uncoded repair property for *systematic nodes* (i.e., nodes that store original data chunks) but not for the parity nodes that store encoded chunks. Wang *et al.* [47] propose codes that achieve uncoded repair for both systematic and parity nodes, but the codes require the total number of data chunks being stored increase exponentially with the number of systematic nodes. Rashmi *et al.* [30] propose product-matrix (PM) MSR codes that support uncoded repair, but the storage redundancy $\frac{n}{k}$ of PM-MSR codes needs to be at least $(2 - \frac{1}{k})\times$. Pamies-Juarez *et al.* [27] present Butterfly codes, which are MSR codes that support uncoded repair and have storage redundancy below $2\times$, but the number of chunks per node is 2^{k-1} (i.e., exponential with k). Our PMSR codes are MSR codes that support uncoded repair, while achieving storage redundancy below $2\times$ (i.e., low storage overhead)) and having a polynomial number of chunks per node (i.e., small subpacketization overhead). Furthermore, we analyze concurrent node failures and design a family of MSR codes for repairing two node failures with the minimum repair bandwidth.

Aside regenerating codes, some studies (e.g., [23], [46], [54], [55]) propose uncoded repair schemes that minimize the amount of disk reads for existing XOR-based erasure codes (e.g., RDP [11], EVENODD [4], and STAR [20]). Some studies [19], [37], [53] propose implementations of locally repairable codes, which support uncoded repair, and deploy them in practical distributed storage systems. However, the codes that they consider do not achieve the optimal storage-bandwidth trade-off as regenerating codes.

Our recent applied work NCCloud [9] (extended from the conference version [17]) builds a network-coding-based cloud storage system, which implements FMSR codes to minimize the repair bandwidth for repairing a single node failure with uncoded repair. Shum and Hu [40] analyze the correctness of FMSR codes for a special case of two systematic nodes (i.e., $k = 2$). In this paper, we generalize the analysis for more systematic nodes (i.e., $k > 2$) and address the repair of concurrent multi-node failures.

III. Problem Formulation for Proxy-Assisted Regeneration

In this section, we formulate the problem, derive the lower bound of β, and establish a *necessary condition* of preserving the (n, k) MDS property in proxy-assisted regeneration. We do not treat our derivations as a contribution of this paper, since we mainly extend the information flow analysis of Dimakis *et al.* [13]. Also, the previous work [25] gives the identical result, although it only provides a sketch of proof. Here, we only present a rigorous proof for completeness, and use it as a starting point for our later existence proof (see Section V) and semi-deterministic code construction (see Section VI).

A. Proxy-Assisted Regeneration

We formulate the problem of *proxy-assisted regeneration*, whose core idea is to coordinate the repair process through a centralized proxy.

We first define the notation. We encode an original file of size M via an (n, k) MDS code into n encoded fragments, which will be distributed and stored at n distinct nodes for storage. Let $X_1, X_2, \cdots, X_n$ be the n nodes. Suppose that r (where $1 \leq r \leq n - k$) nodes fail and their stored fragments are lost. Our goal is to repair the lost fragments and store the repaired fragments in r new nodes to preserve fault tolerance. Without loss of generality, we assume that nodes $X_1, X_2, \cdots, X_r$ fail, and let $X'_1, X'_2, \cdots, X'_r$ be the corresponding new nodes.

Proxy-assisted regeneration can be decomposed into three steps:

1) The proxy downloads data from all surviving nodes $X_{r+1}, X_{r+2}, \cdots, X_n$.
2) The proxy encodes the collected data into repaired fragments of size $\frac{M}{k}$ each.
3) The proxy uploads the repaired fragments to the new nodes $X'_1, X'_2, \cdots, X'_r$.

Our analysis makes the following key assumptions. First, we require that $k > r$; otherwise (i.e., when $r \geq k$), the proxy can download k fragments to first reconstruct the original file and hence the lost fragments. The condition $k > r$ is commonly found in real-life distributed storage systems. For example, Facebook's warehouse cluster [31] sets $(n, k) = (14, 10)$. Second, we consider a homogeneous setting, in which all nodes have the same storage and bandwidth capacities. Finally, we assume that the proxy is always available, which can be enforced through a distributed proxy setting in practice [1].

Based on our assumptions, let β be the amount of data that a proxy downloads from each surviving node for repair (i.e., step (1)). Our primary goal is to minimize the amount of traffic transferred during repair. To achieve this, *our analysis minimizes β, while preserving (n, k) MDS property after repair*. Note that we do not need to consider the amount of repaired fragments that a proxy uploads to the new nodes, since the amount is always fixed at $\frac{rM}{k}$. In addition, we can prove that the amount of data that the proxy downloads from surviving nodes must be larger than that it uploads to the new nodes (see Section III). Thus, if we pipeline the download and upload processes, the download process will be the bottleneck. Thus, under proxy-assisted regeneration, we re-define the *repair bandwidth* as the amount of data that the proxy downloads from all surviving nodes.

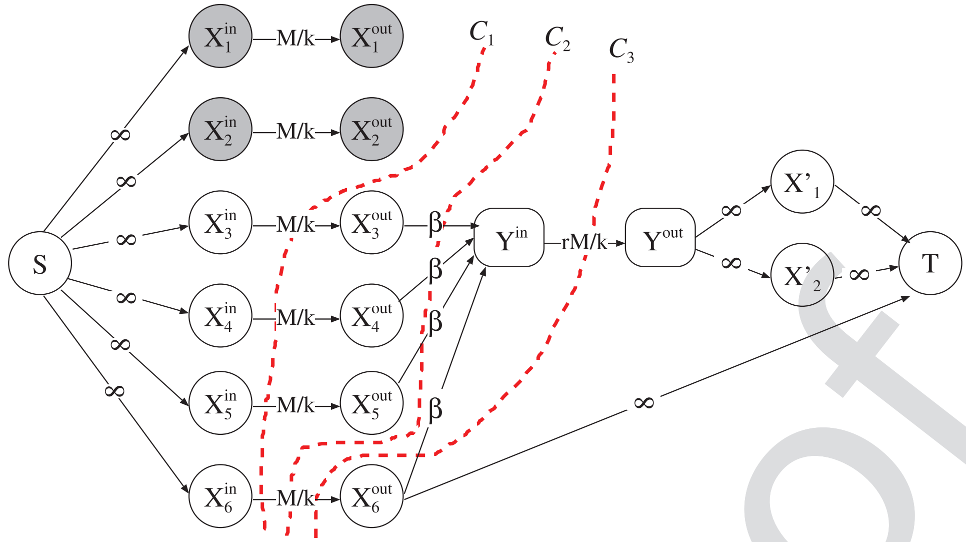

Fig. 1. Information flow graph $\mathcal{G}$ for $(n, k, r) = (6, 3, 2)$.

B. Information Flow Graph

We construct an information flow graph $\mathcal{G}$ and derive the lower bound of β through information flow analysis. Figure 1 illustrates an example of $\mathcal{G}$ for $(n, k, r) = (6, 3, 2)$. Our analysis extends the one by Dimakis *et al.* [13] to include a new proxy and address the repair of multiple node failures.

1) Nodes in $\mathcal{G}$:

- We add a *virtual source* S and a *data collector* T as the source and destination nodes of information flow to $\mathcal{G}$, respectively.
- For each node X_i (where $1 \le i \le n$), we expand it into an input/output node pair (X_i^{in}, X_i^{out}) and add the pair to $\mathcal{G}$.
- For the proxy, we add an input/output node pair (Y^{in}, Y^{out}) in $\mathcal{G}$.
- We keep each new node X'_i (where $1 \le i \le r$) in $\mathcal{G}$. As we show later, the new nodes only receive the repaired fragments from the proxy and are not involved in the information flow analysis.

2) Edges in $\mathcal{G}$:

- We add an edge from S to each X_i^{in} (where $1 \le i \le n$) with infinite capacity.
- We add an edge from each X_i^{in} to X_i^{out} (where $1 \le i \le n$) with capacity $\frac{M}{k}$, which represents the amount of data stored in node X_i.
- We add an edge from each surviving node X_i^{out} (where $r+1 \le i \le n$) to Y^{in} with capacity β, which represents the amount of data transferred from a surviving node to the proxy.
- We add an edge from Y^{in} to Y^{out} with capacity $\frac{r \times M}{K}$, which represents the amount of repaired data for the r new nodes.
- We add an edge from Y^{out} to each new node X'_i (where $1 \le i \le r$) with infinite capacity.
- We select k non-failed nodes (i.e., surviving or new nodes) for reconstructing the original file. We then add an edge from each of the k selected nodes to T with infinite capacity.

C. Lower Bound

We derive the lower bound of β by studying the min-cut capacity of $\mathcal{G}$. We define a *cut* as a set of edges, such that removing them from $\mathcal{G}$ will disconnect S and T, and we define a *min-cut* as the cut that has the minimum capacity among all cuts in $\mathcal{G}$. Since the data collector T can reconstruct the original file by connecting to any k out of n nodes, there are $\binom{n}{k}$ connection choices of T. Each choice leads to a different $\mathcal{G}$, and hence a different min-cut. To preserve (n, k) MDS property, the capacity of each possible min-cut must be at least the original file size M; otherwise, the maximum flow from S to T is less than M, and T cannot retrieve enough information to reconstruct the original file. This leads to Lemma 1.

Lemma 1: To preserve (n, k) MDS property in proxy-assisted regeneration, the capacity of each possible min-cut must be at least M.

By Lemma 1, we specify the lower bound of β.

Lemma 2: If the capacity of each possible min-cut of $\mathcal{G}$ is at least M, then $\beta \ge \frac{rM}{k(n-k)}$.

The proof of Lemma 2 is in Appendix A.

Note that Lemma 2 specifies the *necessary condition* of preserving (n, k) MDS property, because if $\beta < \frac{rM}{k(n-k)}$, then there exists some possible min-cut of $\mathcal{G}$ whose capacity is less than M, which makes it impossible for (n, k) MDS property to be maintained by Lemma 1.

In Section IV, we will propose a code construction that matches the lower bound. Therefore, the lower bound in Lemma 2 is tight, and our code construction can minimize β.

IV. PMSR Codes

A. Basics

We first present the basics of PMSR codes, which have three design properties.

- **Property 1: PMSR codes satisfy the MDS property.** PMSR codes satisfy the MDS property, such that for any

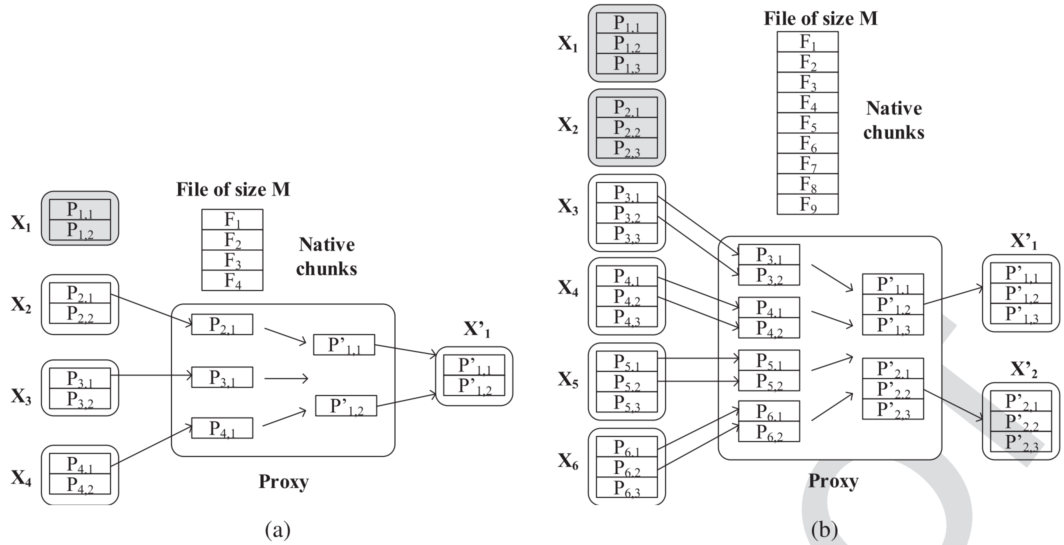

Fig. 2. Two examples of PMSR codes. (a) $n = 4$, $k = 2$ and $r = 1$. (b) $n = 6$, $k = 3$ and $r = 2$.

subset of k out of n nodes, the $k(n-k)$ parity chunks from the k nodes can be decoded to the $k(n-k)$ native chunks of the original file, while incurring the minimum storage redundancy.

- **Property 2: PMSR codes minimize the repair bandwidth.** If $r \geq 1$ nodes fail, we must repair the lost data of r failed nodes to preserve fault tolerance. PMSR codes match the lower bound of β (see Section III). That is, to repair r failed nodes, the proxy only needs to download $\frac{rM}{k(n-k)}$ units of data from each surviving node (see Lemma 2), or equivalently, a size of r parity chunks. Note that for $r = 1$, the lower bound is equivalent to the classical MSR point [13].
- **Property 3: PMSR codes use *uncoded* repair.** During repair, each surviving node under PMSR codes directly sends parity chunks to the proxy without the need of performing any encoding operation.

Based on the above properties, we now provide a basic construction of PMSR codes. Figure 2 shows two examples of PMSR codes.

1) File Distribution (Property 1): To store a file of size M units, a PMSR code splits the file evenly into $k(n-k)$ *native chunks*, say $F_1, F_2, \ldots, F_{k(n-k)}$, and encodes them into $n(n-k)$ *parity chunks* of size $\frac{M}{k(n-k)}$ each. Each l^{th} parity chunk is formed by a linear combination of the $k(n-k)$ native chunks, i.e, $\sum_{m=1}^{k(n-k)} \alpha_{l,m} F_m$ for some encoding coefficients $\alpha_{l,m}$. All encoding coefficients and arithmetic are operated over a finite field $\mathbb{F}_q$ of size q. We store the $n(n-k)$ parity chunks on n nodes $X_1, X_2, \cdots, X_n$, each keeping $n-k$ parity chunks. The original file can be reconstructed by decoding $k(n-k)$ parity chunks of any k nodes, where decoding can be done by inverting an encoding matrix [29]. Let $P_{i,j}$ be the j^{th} parity chunk stored on node i, where $i = 1, 2, \ldots, n$ and $j = 1, \ldots, n-k$.

2) Repair Process (Properties 2 and 3): To preserve the MDS property over multiple rounds of repair, our prior work NCCloud [9] uses random chunk selection in a way that verifies whether the selection can ensure that there is no linear dependence in chunk regeneration that can lead to the loss of the MDS property. This way is used and implemented in NCCloud. Specifically, the proxy performs the m^{th} (where $m \geq 1$) round of repair as follows (suppose that we have r failed nodes $X_1, X_2, \cdots, X_r$):

(i) The proxy directly downloads r parity chunks from each surviving node i ($r+1 \leq i \leq n$). The proxy then generates random encoding coefficients and encodes the $r(n-r)$ downloaded parity chunks into a set of $r(n-k)$ linearly independent parity chunks $P'_{i',j'}$ ($1 \leq i' \leq r$ and $1 \leq j' \leq n-k$).

(ii) The proxy then performs *two-phase checking*. In the first phase, it checks if the MDS property is satisfied with the new chunks generated (i.e., the chunks of any k out of n nodes remain decodable) after the current m^{th} round of repair. In the second phase, it further checks if the MDS property is still satisfied after the $(m+1)^{th}$ round of repair for any possible node failure.

(iii) If both phases are passed, the proxy uploads the generated chunks $P'_{i',1}, P'_{i',2}, \cdots, P'_{i',n-k}$ to each new node $X'_{i'}$ ($1 \leq i' \leq r$); otherwise, it repeats (i) and (ii) with another collection of random chunks and random encoding coefficients.

We explain why two-phase checking is required. Since PMSR codes regenerate different chunks in each repair, one major challenge of PMSR codes is to preserve the MDS property after multiple rounds of repair. We illustrate it with an example in Figure 2(a). Suppose that X_1 fails, and we construct new chunks $P'_{1,1}$ and $P'_{1,2}$ using $P_{2,1}$, $P_{3,1}$, and $P_{4,1}$ as in Figure 2(a). Now, suppose that X_2 fails afterwards. If we construct new chunks $P'_{2,1}$ and $P'_{2,2}$ using $P'_{1,1}$, $P_{3,1}$, and $P_{4,1}$, then in the two new nodes, the chunks $\{P'_{1,1}, P'_{1,2}, P'_{2,1}, P'_{2,2}\}$ are the linear combinations of only three chunks $P_{2,1}$, $P_{3,1}$, and $P_{4,1}$ instead of four. Hence, the chunks in these two new nodes are *not* decodable, and the MDS property is lost. Therefore, the two-phase checking is to preserve the MDS property over multiple rounds of repair.

3) Examples: We show via examples the repair bandwidth saving of PMSR codes compared to the conventional repair (e.g., as used by Reed-Solomon codes [34]). In conventional repair, the proxy needs to read k fragments from any k surviving nodes to first reconstruct the original file and then the lost data for all failed nodes. Clearly, the repair bandwidth is the file size M. For PMSR codes, we consider the two settings in Figure 2. For $n = 4$, $k = 2$, and $r = 1$ (see Figure 2(a)), the repair bandwidth of PMSR codes is $0.75M$, i.e., 25% less than that of conventional repair. For $n = 6$, $k = 3$, and $r = 2$ (see Figure 2(b)), the repair bandwidth of PMSR codes is $8M/9$, i.e., 11.11% less than that of conventional repair. In general, the repair bandwidth saving of PMSR codes increases with n. For example, for $k = n - 2$ and $r = 1$, the repair bandwidth of PMSR codes is $\frac{M(n-1)}{2(n-2)}$. The saving compared to standard RAID-6 codes [22], which are also double-fault tolerant, is up to 50% if n is large. For $k = n - 3$ and $r = 2$, the repair bandwidth of PMSR codes is $\frac{2M(n-2)}{3(n-3)}$. The saving compared to STAR codes [20], which are also triple-fault tolerant, is up to 33.3% if n is large.

B. Formulation of Repair Process of PMSR Codes

We provide a theoretical framework for the repair process of PMSR codes so as to formally define PMSR codes which is based on three preliminary definitions.

Definition 1 (Decodability): We say that a collection of $k(n-k)$ parity chunks is decodable if the parity chunks can be decoded to the original file, which can be verified by checking if the associated $k(n-k)$ vectors of encoding coefficients are linearly independent. Note that these $k(n-k)$ parity chunks may be scattered among n nodes, and need not reside in exactly k nodes. □

Recall that PMSR codes are non-systematic codes (see Section I). It means that PMSR codes operate on parity chunks. For simplicity, when we use the term "chunk" in our discussion, we actually refer to a parity chunk.

Definition 2 (Repair-Based Collection (RBC)): An RBC of the m^{th} round of repair is a collection of $k(n-k)$ chunks that exist after the m^{th} round of repair as follows:

(i) *We select any $n - r$ out of n nodes.*

(ii) *We select $k - r$ out of the $n - r$ nodes in Step (i) and choose $n - k$ chunks from each selected node.*

(iii) *We select the remaining $n - k$ out of the $n - r$ nodes in Step (i) and choose r chunks from each selected node. Clearly, the number of chunks of an RBC is $(k - r)(n - k) + (n - k)r = k(n - k)$.* □

The physical meaning of an RBC after the m^{th} round of repair is as follows. As stated in Section IV-A, the repair process of PMSR codes needs to perform two-phase checking; that is, it not only checks whether the $k(n-k)$ chunks of any k nodes after the m^{th} round of repair are decodable, but also checks whether the $k(n-k)$ chunks of any k nodes after the $(m+1)^{th}$ round of repair are still decodable. For example, after repairing X_1 in Figure 2(a), PMSR codes first check whether four chunks of any two nodes out of X'_1, X_2, X_3, X_4 are decodable. Then let us consider the next round of repair for a failed node, say X_2. A new node X'_2 is repaired in the same way as X'_1, with its chunks reconstructed by the proxy and generated from three random chunks from the surviving nodes, say $P'_{1,1}$, $P_{3,1}$, and $P_{4,1}$. Finally, PMSR codes further check whether the four chunks of any two nodes out of X'_1, X'_2, X_3, X_4 are decodable. If we select X'_2 and X_3, then we can see that the four chunks of X'_2 and X_3 are obviously the linear combinations of the collection $\{P'_{1,1}, P_{3,1}, P_{3,2}, P_{4,1}\}$, which happens to be an RBC after repairing X'_1. Thus, if this RBC is not decodable, then the MDS property is not maintained after repairing X'_2. In fact, based on proxy-assisted regeneration (see Section III), for the $k(n-k)$ chunks of any k nodes after the $(m+1)^{th}$ round of repair, we can always find an RBC of the m^{th} round of repair such that these $k(n-k)$ chunks are linear combinations of the RBC (we will provide the reason in Section V). Therefore, to maintain the MDS property after the $(m+1)^{th}$ round of repair, we must ensure that the corresponding RBCs of the m^{th} round of repair are decodable.

Note that there exist some provably non-decodable RBCs (i.e., they are linear combinations of a set of fewer than $k(n-k)$ chunks). For example, in Figure 2(a), the RBCs $\{P'_{1,1}, P'_{1,2}, P_{2,1}, P_{3,1}\}$, $\{P'_{1,1}, P'_{1,2}, P_{2,1}, P_{4,1}\}$, and $\{P'_{1,1}, P'_{1,2}, P_{3,1}, P_{4,1}\}$ are non-decodable, since $P'_{1,1}$ and $P'_{1,2}$ are linear combinations of $P_{2,1}$, $P_{3,1}$, $P_{4,1}$. In other words, all the chunks of each of the above three RBCs are linear combinations of $P_{2,1}$, $P_{3,1}$, $P_{4,1}$, which have fewer than $k(n-k)$ chunks in total. Accordingly, we define the following:

Definition 3 (Linear Dependent Collection (LDC)): Consider an RBC of the m^{th} round of repair. If and only if all the chunks of this RBC are linear combinations of a set of fewer than $k(n-k)$ chunks from all the surviving nodes, then it is called an LDC of the m^{th} round of repair. □

Definition 4 (Repair MDS (rMDS) Property): If all RBCs, after excluding the LDCs, of the m^{th} round of repair are decodable, then we say the rMDS property is satisfied. □

Note that even though some RBCs are not classified as LDCs (i.e., they are linear combinations of a set of $k(n-k)$ chunks from all the surviving nodes), they may still be non-decodable due to some "bad" linear combinations that cause linear dependency as a result of the selection of wrong encoding coefficients. Thus, the rMDS property is to handle this case and ensures that only if the rMDS property is maintained, all RBCs except LDCs are decodable.

Definition 5 ((n,k,r)-PMSR Codes): An original file is stored in n nodes in the form of $n(n-k)$ chunks. If these $n(n-k)$ chunks satisfy both the MDS and rMDS properties after repairing r failed nodes, then we say that this file is PMSR-encoded.

Previous Results: Our previous work NCCloud [9] shows via simulations that by checking both the MDS and rMDS properties in each round of single node failures (i.e., $r = 1$), PMSR codes can preserve the MDS property after hundreds of rounds of repair. Also, if we only check the MDS property but not the rMDS property, then after some rounds of repair, we cannot regenerate the chunks that preserve the MDS property within a fixed number of iterations (this is called the *bad* repair [9]). In the following, we present formal theoretical analysis that justifies the need of two-phase checking to preserve the MDS property after *any* number of rounds of repair. Our analysis also addresses the repair of a general number $r \geq 1$ of concurrent node failures.

V. Existence of PMSR Codes

We now prove the existence of PMSR codes. In this work, we focus on two cases: (i) $r = 1$ when $k = n-2$, implying that PMSR codes are double-fault tolerant as traditional RAID-6 codes [22]; (ii) $r = 2$ when $k = n - 3$, implying that PMSR codes are triple-fault tolerant as STAR codes [20].

A. PMSR Codes With $k = n - 2$ and $r = 1$

Double-fault tolerance has been assumed in real cloud storage systems (e.g., GFS [15] and Azure [7]). Our goal is to show that PMSR codes always maintain double-fault tolerance (i.e., the MDS property is always satisfied with $k = n - 2$) after *any* number of rounds of uncoded repair, while the repair bandwidth is equal to $\frac{M}{k(n-k)}$ units (or equivalently, a size of *one* parity chunk) according to Property 2 in Section IV-A. Note that each node stores $n - k = 2$ parity chunks.

We first give three lemmas. Lemma 3 and Lemma 4 provide a guideline of how to choose $n-1$ chunks from $n-1$ surviving nodes (one chunk from each node) to repair a failed node. Lemma 5 implies that if the finite field size is large enough, we can always find a set of encoding coefficients to regenerate new chunks for a repaired node so as to maintain the MDS and rMDS properties after each round of repair. Finally, we prove Theorem 1 for the existence of PMSR codes with $k = n - 2$ and $r = 1$.

Lemma 3: In single-node failure repair, let $\mathcal{F}$ be the set of $n - 1$ chunks selected from $n - 1$ surviving nodes to regenerate the $n - k$ chunks of the repaired node. For the RBC of this repair, let $\mathcal{Q}$ be the set of chunks chosen Step 3 of RBC construction (see Definition 2) excluding those from the repaired node. If an RBC (denoted by $\mathcal{R}$) of this repair is an LDC, then $\mathcal{F}$ and $\mathcal{Q}$ have two or more identical common chunks of the surviving nodes.

Proof: Without loss of generality, let node 1 be the failed node. There are two cases to construct an RBC which contains chunks of the repaired node 1: (1) the RBC contains the $n - k = 2$ chunks of the repaired node chosen in Step 2 of RBC construction; (2) the RBC contains one chunk of the repaired node chosen in Step 3 of RBC construction. Note that we do not consider the RBCs which do not contain any chunks of the repaired node 1, because they have already been checked before the m^{th} round of single-node failure repair.

Case 1: Let $\mathcal{P}$ be the set of chunks chosen in Step 2 of Definition 2 excluding those from the repaired node 1. Thus, $\mathcal{R} = \{P'_{1,1}, P'_{1,2}\} \cup \mathcal{P} \cup \mathcal{Q}$. As $\{P'_{1,1}, P'_{1,2}\}$ are obtained by linearly combining the chunks in $\mathcal{F}$, we infer that all the chunks of $\mathcal{R}$ of Case 1 are linear combinations of chunks in $\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}$, which only contain chunks from surviving nodes.

Since $\mathcal{F}$ selects $r = 1$ chunk from each surviving nodes and $\mathcal{P}$ has all the chunks from $k-r-1 = k-2$ out of all the surviving nodes, $\mathcal{F}$ and $\mathcal{P}$ have $k-2$ identical common chunks of the surviving nodes, i.e., $|\mathcal{F} \cap \mathcal{P}| = k-2$. $\mathcal{Q}$ contains $r = 1$ chunk from each of $n - k = 2$ surviving nodes, i.e., $|\mathcal{Q}| = 2$.

According to the given conditions, we can easily have the following equalities: $|\mathcal{F}| = n - 1$, $|\mathcal{P}| = 2(k - 2)$, $|\mathcal{P} \cap \mathcal{Q}| = |\mathcal{F} \cap \mathcal{P} \cap \mathcal{Q}| = 0$. Finally, we can have $|\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}| = |\mathcal{F}| + |\mathcal{P}| + |\mathcal{Q}| - |\mathcal{F} \cap \mathcal{P}| - |\mathcal{F} \cap \mathcal{Q}| - |\mathcal{P} \cap \mathcal{Q}| + |\mathcal{F} \cap \mathcal{P} \cap \mathcal{Q}| = 2k+1-|\mathcal{F} \cap \mathcal{Q}|$.

If an RBC of Case 1 is an LDC, which means $\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}$ are linear combinations of less than $k(n-k)$ chunks from the surviving nodes, then $|\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}| < 2k$. Hence, $|\mathcal{F} \cap \mathcal{Q}| \geq 2$.

Case 2: Similar to Case 1, we infer that all the chunks of $\mathcal{R}$ of Case 2 are linear combinations of chunks in $\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}$.

Since $\mathcal{F}$ selects $r = 1$ chunk from each surviving nodes and $\mathcal{P}$ has all the chunks from $k-r = k-1$ out of all the surviving nodes, $\mathcal{F}$ and $\mathcal{P}$ have $k - 1$ identical common chunks of the surviving nodes, i.e., $|\mathcal{F} \cap \mathcal{P}| = k-1$. $\mathcal{Q}$ contains $r = 1$ chunk from each of $n - k - 1 = 1$ surviving node, i.e., $|\mathcal{Q}| = 1$.

According to the given conditions, we can easily have the following equalities: $|\mathcal{F}| = n - 1$, $|\mathcal{P}| = 2(k - 1)$, $|\mathcal{P} \cap \mathcal{Q}| = |\mathcal{F} \cap \mathcal{P} \cap \mathcal{Q}| = 0$. Finally we can have $|\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}| = |\mathcal{F}| + |\mathcal{P}| + |\mathcal{Q}| - |\mathcal{F} \cap \mathcal{P}| - |\mathcal{F} \cap \mathcal{Q}| - |\mathcal{P} \cap \mathcal{Q}| + |\mathcal{F} \cap \mathcal{P} \cap \mathcal{Q}| = 2k+1-|\mathcal{F} \cap \mathcal{Q}|$. Because $|\mathcal{Q}| = 1$, $|\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}| \geq 2k$, which means the RBC of case 2 is never an LDC.

Therefore, Lemma 3 holds. □

Lemma 4: Suppose that the rMDS property is satisfied after every m^{th} round of single-node failure repair. Then for any $n - 1$ out of n nodes, we can always select one chunk from these $n - 1$ nodes (i.e., a total of $n - 1$ chunks) such that any RBC containing the selected $n - 1$ chunks is decodable.

Proof: Without loss of generality, suppose that we construct an RBC $\mathcal{R}$ by selecting the chunks from nodes $2, \ldots, n$ (see Step 1 of Definition 2), and that $\mathcal{H}$ be the set of $n - 1$ chunks selected from nodes $2, \ldots, n$ (one chunk from each node). We prove the existence of $\mathcal{H}$ such that if $\mathcal{R}$ contains $\mathcal{H}$ (i.e., $\mathcal{H} \subset \mathcal{R}$), then $\mathcal{R}$ is decodable.

If node 1 is the repaired node in the m^{th} round of repair, then all the $k(n-k)$ chunks of each possible $\mathcal{R}$ must come from the surviving nodes. Thus, $\mathcal{R}$ is never an LDC (by Definition 3). Since the rMDS property is satisfied by our assumption, $\mathcal{R}$ is decodable (by Definition 4).

If node 1 is not the repaired node in the m^{th} round of repair, then without loss of generality, let node 2 be the repaired node and the new parity chunks are $P'_{2,1}$ and $P'_{2,2}$. By the PMSR code design, the chunks of node 2 are linearly combined by one chunk in each of nodes $1, 3, \ldots, n$. We denote these chunks by $\mathcal{F} = \{P_{1,f(1)}, P_{3,f(3)}, \ldots, P_{n,f(n)}\}$. Since each node has $n - k = 2$ chunks, we can construct $\mathcal{H} = \{P'_{2,g(2)}, P_{3,g(3)}, \ldots, P_{n,g(n)}\}$ such that $g(i) \neq f(i)$ for $i = 3, \ldots, n$ (while $g(2)$ can be randomly picked). Let $\mathcal{Q}$ be the set of chunks chosen in Step 3 of Definition 2 excluding those from the repaired node. If $\mathcal{R}$ contains $\mathcal{H}$, then $\mathcal{Q}$ and $\mathcal{F}$ have no identical common chunks of the surviving nodes. By Lemma 3, $\mathcal{R}$ is not an LDC. Since the rMDS property is satisfied, $\mathcal{R}$ is decodable. □

Based on Lemma 4, we have the following claim.

Claim 1: Consider an RBC that selects $n-1$ nodes out of n nodes except node 1. There exists a set of $n-1$ chunks, denoted by $\mathcal{F} = \{P_{2,f(2)}, \ldots, P_{k+2,f(k+2)}\}$ (i.e., one chunk is retrieved from each of nodes $2, \ldots, n$), such that the RBC containing

$\mathcal{F}$ must be decodable. Here, $f(i)$, where $2 \leq i \leq k+2$, denotes a function that specifies the index of the chunk to be retrieved from surviving node i to the proxy. For example, if the retrieved chunk of the surviving node 4 is its second chunk, then $f(4) = 2$.

Lemma 5 (Schwartz-Zippel Theorem [26]): Consider a multivariate non-zero polynomial $h(x_1, \ldots, x_t)$ of total degree ρ over a finite field $\mathbb{F}$. Let $\mathbb{S}$ be a finite subset of $\mathbb{F}$, and $\tilde{x}_1, \ldots, \tilde{x}_t$ be the values randomly selected from $\mathbb{S}$. Then the probability $\Pr[h(\tilde{x}_1, \ldots, \tilde{x}_t) = 0] \leq \frac{\rho}{|\mathbb{S}|}$.

Theorem 1: Consider a file encoded using PMSR codes with $k = n-2$. In the m^{th} ($m \geq 1$) round of uncoded repair of some failed node j, the lost chunks are reconstructed by the random linear combination of $n-1$ chunks selected from $n-1$ surviving nodes (one chunk from each node). Then after the repair, the distributed storage system still satisfies both the MDS and rMDS properties with probability that can be driven arbitrarily to 1 by increasing the size of $\mathbb{F}_q$.

Proof: We prove by induction on m. Initially, we use Reed-Solomon codes to encode a file into $n(n-k) = 2n$ chunks that satisfy both the MDS and rMDS properties. Suppose that after the m^{th} round of repair, both the MDS and rMDS properties are satisfied (this is our induction hypothesis).

Let $\mathcal{U}_m = \{P_{1,1}, P_{1,2}; \ldots; P_{k+2,1}, P_{k+2,2}\}$ be the current set of chunks after the m^{th} round of repair. In the $(m+1)^{th}$ round of repair, without loss of generality, let node 1 be the failed node to repair.

Since $\mathcal{U}_m$ satisfies the rMDS property, we use $\mathcal{F}$ to repair node 1 with the help of Claim 1. Suppose that the repaired node 1 has the new chunks $\{P'_{1,1}, P'_{1,2}\}$. Then

$$P'_{i',j'} = \sum_{i=2}^{k+2} \gamma^{i}_{i',j'} P_{i,f(i)}, \quad \text{for } i' = 1, j' = 1, 2. \tag{1}$$

Here $\gamma^{i}_{i',j'}$ denotes the encoding coefficient for the single retrieved chunk of the surviving node i to generate the the j'^{th} chunk of the new node i'. Next we prove that we can always tune $\gamma^{i}_{i',j'}$ in $\mathbb{F}_q$ in such a way that the set of chunks in the $(m+1)^{th}$ round of repair $\mathcal{U}_{m+1} = \{P'_{1,1}, P'_{1,2}; \ldots; P_{k+2,1}, P_{k+2,2}\}$ still satisfies both MDS and rMDS properties. The proof consists of two parts.

Part I $\mathcal{U}_{m+1}$ Satisfies the MDS Property: Since $\mathcal{U}_m$ satisfies the MDS property, we only need to ensure that for any $k-1$ surviving nodes, say for any subset $\{s_1, \ldots, s_{k-1}\} \subseteq \{2, \ldots, n\}$, all the $k(n-k)$ chunks (denoted by $\mathcal{V}$) of nodes $s_1, \ldots, s_{k-1}$ and the repaired node 1 are decodable.

First, we prove that every chunk of $\mathcal{V}$ is a linear combination of a certain decodable RBC (denoted by $\mathcal{R}$) and let $\mathbf{A}$ be the encoding matrix which shows the linear combination, i.e., $\mathcal{V} = \mathbf{A} \times \mathcal{R}$. Without loss of generality, let $(s_1, \ldots, s_{k-1}) = (2, \ldots, k)$, and the other cases are symmetric. In this case, $\mathcal{V} = \{P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}; P'_{1,1}, P'_{1,2}\}$, i.e., the set of chunks of nodes 1 to k. By Equation (1), each chunk of $\mathcal{V}$ is a linear combination given by $\mathcal{R} = \{P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}; P_{k+1,f(k+1)}, P_{k+2,f(k+2)}\}$. Mathematically, we express as:

$$\begin{bmatrix} P_{2,1} \\ P_{2,2} \\ \cdots \\ P_{k,1} \\ P_{k,2} \\ P'_{1,1} \\ P'_{1,2} \end{bmatrix} = \mathbf{A} \times \begin{bmatrix} P_{2,1} \\ P_{2,2} \\ \cdots \\ P_{k,1} \\ P_{k,2} \\ P_{k+1,f(k+1)} \\ P_{k+2,f(k+2)} \end{bmatrix},$$

where $\mathbf{A}$ is a $k(n-k) \times k(n-k)$ (i.e., $2k \times 2k$) encoding matrix given by

$$\mathbf{A} = \begin{pmatrix} 1, 0, & \cdots, & 0, 0, & 0, 0 \\ 0, 1, & \cdots, & 0, 0, & 0, 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0, 0, & \cdots, & 1, 0, & 0, 0 \\ 0, 0, & \cdots, & 0, 1, & 0, 0 \\ \delta_1^2 \gamma_{1,1}^2, \delta_2^2 \gamma_{1,1}^2, & \cdots, & \delta_1^k \gamma_{1,1}^k, \delta_2^k \gamma_{1,1}^k, & \gamma_{1,1}^{k+1}, \gamma_{1,1}^{k+2} \\ \delta_1^2 \gamma_{1,2}^2, \delta_2^2 \gamma_{1,2}^2, & \cdots, & \delta_1^k \gamma_{1,2}^k, \delta_2^k \gamma_{1,2}^k, & \gamma_{1,2}^{k+1}, \gamma_{1,2}^{k+2} \end{pmatrix}$$

where $\delta_1^i = 1$, $\delta_2^i = 0$ when $f(i) = 1$; and $\delta_1^i = 0$, $\delta_2^i = 1$ when $f(i) = 2$. Since $\mathcal{R}$ is an RBC consisting of set $\mathcal{F}$ (i.e., the $n-1$ chunks from $n-1$ nodes), it is decodable due to Lemma 4.

In addition, the determinant $det(\mathbf{A})$ is a multivariate polynomial in terms of variables $\gamma^{i}_{i',j'}$. By Lemma 5, the value of $det(\mathbf{A})$ is non-zero, with probability driven to 1 if we increase the finite field size. Now since $\mathcal{R}$ is decodable and $\mathbf{A}$ has a full rank, $\mathcal{V}$ is decodable.

Similarly, for any $\{s_1, \ldots, s_{k-1}\} \subseteq \{2, \ldots, n\}$, all the $k(n-k)$ chunks of node $s_1, \ldots, s_{k-1}$ and the repaired node 1 are linear combinations of a certain decodable RBC. In addition, $\prod_{\{s_1,\ldots,s_{k-1}\} \subseteq \{2,\ldots,n\}} det(\mathbf{A})$ is also a multivariate polynomial in terms of variables $\gamma^{i}_{i',j'}$ and the value of it is also non-zero with probability driven to 1 if we increase the finite field size by Lemma 5. Therefore, for any $s_1, \ldots, s_{k-1}$, the corresponding $\mathcal{V}$ is decodable. This implies that $\mathcal{U}_{m+1}$ satisfies the MDS property.

Part II $\mathcal{U}_{m+1}$ Satisfies the rMDS Property: By Definition 4, we need to prove that all the RBCs of $\mathcal{U}_{m+1}$ except the LDCs are decodable. By Definition 2, we only consider two cases of RBCs which contain the chunks of the repaired node 1 due to the same reason stated by Lemma (3).

Case 1: The repaired node 1 is selected in Step 2. Suppose in Step 1, an RBC needs to select $(n-r)-1 = k$ additional surviving nodes, say $\{s_1, \ldots, s_k\} \subseteq \{2, \ldots, n\}$. Then in Step 2, the RBC further selects any subset of $(k-r)-1 = k-2$ nodes if $k > 2$, say nodes $s_1, \ldots, s_{k-2}$. If $k \leq 2$, the RBC does not have to select additional nodes. Finally, in Step 3, the RBC chooses two chunks, denoted by $P_{s_{k-1},g(s_{k-1})}$ and $P_{s_k,g(s_k)}$ from the remaining two nodes s_{k-1} and s_k, respectively. Without loss of generality, let $(s_1, \ldots, s_{k-2}) = (2, \ldots, k-1)$ and $(s_{k-1}, s_k) = (k, k+1)$.

Denote the RBC by $\mathcal{R}_1 = \{P_{2,1}, P_{2,2}; \ldots; P_{k-1,1}, P_{k-1,2}; P'_{1,1}, P'_{1,2}; P_{k,g(k)}, P_{k+1,g(k+1)}\}$. In addition, by Equation (1), the chunks of $\mathcal{R}_1$ are linear combinations of a set of chunks denoted by $\mathcal{X} = \{P_{2,1}, P_{2,2}; \ldots; P_{k-1,1}, P_{k-1,2}; P_{k,g(k)}, P_{k,f(k)}; P_{k+1,g(k+1)}, P_{k+1,f(k+1)}; P_{k+2,f(k+2)}\}$.

Our goal is to show that if $\mathcal{R}_1$ is not an LDC, then it is decodable. Clearly, if and only if $g(k) = f(k)$ and $g(k+1) = f(k+1)$, $\mathcal{X}$ have less than $k(n-k)$ chunks of the surviving nodes after the m^{th} repair such that $\mathcal{R}_1$ becomes an LDC. Thus, to prove that $\mathcal{R}_1$ except the LDCs is decodable, it is equivalent to prove that $\mathcal{R}_1$ is decodable when (a) $g(k) \neq f(k)$ and $g(k+1) = f(k+1)$, (b) $g(k) = f(k)$ and $g(k+1) \neq f(k+1)$, or (c) $g(k) \neq f(k)$ and $g(k+1) \neq f(k+1)$.

First consider (a). We can reduce $\mathcal{X}$ to $\{P_{2,1}, P_{2,2}; \ldots; P_{k-1,1}, P_{k-1,2}; P_{k,1}, P_{k,2}; P_{k+1,f(k+1)}, P_{k+2,f(k+2)}\}$. The above collection is an RBC containing $\mathcal{F}$. By Claim 1, the collection is decodable. Therefore, $\mathcal{R}_1$ is linear combination of a decodable collection. We can also prove that $\mathcal{R}_1$ is a linear combination of a decodable collection which is similar to (a) and thus omit the proof.

Lastly, let us consider (c). Now, $\mathcal{X}$ can be written as $\{P_{2,1}, P_{2,2}; \ldots; P_{k-1,1}, P_{k-1,2}; P_{k,1}, P_{k,2}; P_{k+1,1}, P_{k+1,2}; P_{k+2,f(k+2)}\}$. Define $\overline{\mathcal{X}} = \mathcal{X} - \{P_{k+2,f_{k+2}}\}$. Note that the MDS property of $\overline{\mathcal{X}}$ is satisfied by induction hypothesis. Thus, $\overline{\mathcal{X}}$ is decodable, implying that $P_{k+2,f(k+2)}$ can be seen as a linear combination of $\overline{\mathcal{X}}$. Obviously, we can also say that $\mathcal{X}$ is a linear combination of $\overline{\mathcal{X}}$. Therefore, $\mathcal{R}_1$ is also a linear combination of the decodable collection $\overline{\mathcal{X}}$.

Case 2: The repaired node 1 is selected in Step 3. Suppose in Step 1, the RBC selects any $n-2 = k$ surviving nodes, say $\{s_1, \ldots, s_k\} \subseteq \{2, \ldots, n\}$. Then in Step 2, the RBC further selects any subset of $k-1$ nodes, say $s_1, \ldots, s_{k-1}$ to choose all the chunks of nodes $s_1, \ldots, s_{k-1}$. Finally, in Step 3, the RBC chooses two chunks $P'_{1,g(1)}$ and $P_{s_k,g(s_k)}$ from the repaired node 1 and the last selected node s_k, respectively. Without loss of generality, let $(s_1, \ldots, s_{k-1}) = (2, \ldots, k)$ and $s_k = k+1$.

Denote the RBC by $\mathcal{R}_2 = \{P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}; P'_{1,g(1)}, P_{k+1,g(k+1)}\}$. We need to show that if $\mathcal{R}_2$ is not a LDC, it is decodable. Based on Lemma 3, there is no more than one identical chunk between $\mathcal{F}$ and the RBC's chunks chosen in Step 3, so $\mathcal{R}_2$ is never an LDC. We just prove that every possible $\mathcal{R}_2$ is decodable.

By Equation (1), the chunks of $\mathcal{R}_2$ are linear combinations of a set of chunks denoted by $\mathcal{Y} = \{P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}; P_{k+1,g(k+1)}, P_{k+1,f(k+1)}; P_{k+2,f(k+2)}\}$. Suppose $g(k+1) \neq f(k+1)$. Define $\overline{\mathcal{Y}} = \mathcal{Y} - \{P_{k+1,g(k+1)}\}$. Since $\overline{\mathcal{Y}}$ is an RBC containing $\mathcal{F}$, by Claim 1, $\overline{\mathcal{Y}}$ is decodable. Therefore, $P_{k+1,g(k+1)}$ can be seen as a linear combination of $\overline{\mathcal{Y}}$. Obviously, we can also say $\mathcal{Y}$ is a linear combination of $\overline{\mathcal{Y}}$. Therefore, $\mathcal{R}_2$ is also linear combination of the decodable collection $\overline{\mathcal{Y}}$.

Combining Case 1 and Case 2, we deduce that each RBC excluding the LDCs is linear combination of a decodable collection, and let $\mathbf{B}$ be the encoding matrix which shows the linear combination for a certain set $\{s_1, \ldots, s_k\}$. Similar to Part I, for all possible $\{s_1, \ldots, s_k\}$ of Case 1 and Case 2, $\prod_{\{s_1,\ldots,s_k\}\subseteq\{2,\ldots,n\}} det(\mathbf{B})$ is also a multivariate polynomial in terms of variables $\gamma^i_{i',j'}$, and by Lemma 5 there always exists an assignment of $\gamma^i_{i',j'}$ in a sufficiently large field such that $\mathcal{R}_1$ and $\mathcal{R}_2$ are also decodable. This implies $\mathcal{U}_{m+1}$ satisfies the rMDS Property.

Therefore, for Part I and II, there always exists an assignment of $\gamma^i_{i',j'}$ in a sufficiently large field such that

$$\prod_{\{s_1,\ldots,s_{k-1}\}\subseteq\{2,\ldots,n\}} det(\mathbf{A}) \times \prod_{\{s_1,\ldots,s_k\}\subseteq\{2,\ldots,n\}\ of\ Case\ 1,2} det(\mathbf{B}) \neq 0. \tag{2}$$

This concludes the proof of Theorem 1. □

B. PMSR Codes With $k = n-3$ and $r = 2$

We now extend the analysis for PMSR codes for a more complicated case $k = n-3$ and $r = 2$. In Section V-A, we have analyzed the case of optimally repairing a single node failure under double fault tolerance. We can readily generalize the analytical result to the case of optimally repairing a single node failure under triple fault tolerance. Thus, we here only focus on the case of optimally repairing a double-node failure. Our goal is to show that PMSR codes always maintain triple-fault tolerance (i.e., the MDS property is always satisfied with $k = n-3$) after *any* number of rounds of uncoded double-node repair, while the repair bandwidth is equal to $\frac{2M}{k(n-k)}$ units (or equivalently, a size of *two* parity chunks) according to Property 2 in Section IV. Note that each node stores $n-k = 3$ parity chunks. We will give two new lemmas and a new theorem as in Section V-A. While the proof steps are similar to those for Lemma 3, Lemma 4, and Theorem 1, the proof details become more complicated and cannot be directly obtained since we need to address more cases. To make the paper more concise, we present the proofs in Appendix B, C and D.

Lemma 6: In double-node failure repair, let $\mathcal{F}$ be the set of $2(n-2)$ chunks selected from $n-2$ surviving nodes to regenerate the six chunks of two repaired nodes. For the RBC of this double-node failure repair, let $\mathcal{Q}$ be the set of chunks chosen in Step 3 of Definition 2 excluding those from all the repaired nodes. If an RBC (denoted by $\mathcal{R}$) of this repair is an LDC, then $\mathcal{F}$ and $\mathcal{Q}$ have four or more identical common chunks of the surviving nodes.

Lemma 7: Suppose that the rMDS property is satisfied after every m^{th} round of double-node failure repair. Then for any $n-2$ out of n nodes, we can always select two chunks from these $n-2$ nodes (i.e., a total of $2(n-2)$ chunks) such that any RBC containing the selected $2(n-2)$ chunks is decodable.

Based on Lemma 7, we have the following claim.

Claim 2: Consider an RBC that selects $n-2$ nodes out of n nodes except node 1 and node 2. There exists a set of $2(n-2)$ chunks, denoted by $\mathcal{F} = \{P_{3,f_1(3)}, P_{3,f_2(3)}, \ldots, P_{k+3,f_1(k+3)}, P_{k+3,f_2(k+3)}\}$ selected from nodes $3, \ldots, n$, such that the RBC containing $\mathcal{F}$ must be decodable. Here, $f_{i'}(i)$ (where $3 \leq i \leq k+3$ and $1 \leq i' \leq 2$) denotes a function that specifies the index of the i'^{th} retrieved chunk of surviving node i to the proxy. For example, if the second retrieved chunk of the surviving node 4 is its third chunk, then $f_2(4) = 3$.

Theorem 2: Consider a file encoded using PMSR codes with $k = n-3$. In the m^{th} $(m \geq 1)$ round of uncoded *repair of two failed node j_1 and node j_2, the lost chunks are reconstructed by the random linear combination of $2(n-2)$ chunks selected from $n-2$ surviving nodes (two chunks from each node). Then after the repair, the distributed storage system still satisfies both the MDS and rMDS properties with probability that can be driven arbitrarily to 1 by increasing the size of $\mathbb{F}_q$.*

C. Discussion on Arbitrary (n, k, r)

We observe that PMSR codes are constructed based on Lemmas 3 and 4. However, it is much more difficult to generalize both lemmas with arbitrary (n, k, r). For example, in the proof of Lemma 3, when $r > 1$, we need to consider more cases of how an RBC includes the chunks of more than one repaired node. How to generalize the construction of PMSR codes for arbitrary (n, k, r) is posed as future work.

VI. Semi-Deterministic PMSR Codes

In NCCloud [9], the repair operation under PMSR codes is accomplished based on two random processes: (i) using random chunk selection to read chunks from the surviving nodes and (ii) applying random linear combinations of the selected chunks to generate new chunks for the repaired node. Section V has proved the correctness of the random-based repair operation by virtue of existence of PMSR codes. On the other hand, a drawback of the random approach is that it may need to try many iterations to generate the correct set of chunks that satisfies both the MDS and rMDS properties.

In this section, we propose a repair scheme under PMSR codes ($k = n-2, r = 1$), such that the chunk selection is deterministic and the linear combination operations are still random but have some inequality constraints. This enables us to significantly speed up the repair operation. Note that the randomness lies in the fact that $\gamma^i_{1,1}$ and $\gamma^i_{1,2}$ (see Equation (1)) are randomly chosen. Thus, we call this family of PMSR codes *semi-deterministic*. In our semi-deterministic construction, we specify which particular chunk should be read from each surviving node in each round of repair. We also derive the sufficient conditions that the encoding coefficients should satisfy. Here, we will use Lemma 4 to design the semi-deterministic construction with $k = n-2, r = 1$, so we can also design a similar semi-deterministic repair scheme with $k = n-3, r = 2$ with the help of Lemma 7. Thus, in this paper we only consider the case of $k = n-2, r = 1$ as a representative case. First, we introduce an evolved repair MDS property.

Definition 6 (Evolved Repair MDS (erMDS) Property): Let $k = n-2$. For any $k+1$ out of n nodes, if we can always select one specific chunk from each of the $k+1$ nodes such that any RBC which consists of these selected $k+1$ chunks is decodable, then we say the code scheme has the erMDS property. □

We see that if the erMDS property is satisfied, then Lemma 4 is ensured, so it suffices for our codes to satisfy both MDS and erMDS properties, and hence Theorem 1 can be satisfied. Thus, we use the erMDS property to construct semi-deterministic PMSR codes.

A. Construction of Semi-Deterministic PMSR Codes

To construct semi-deterministic PMSR codes for $k = n-2, r = 1$, we describe how we store a file and how we trigger the m^{th} ($m \geq 1$) round of repair for a node failure.

1) Storing a File: We divide a file into $k(n-k) = 2k$ equal-size native chunks, and encode them into $n(n-k) = 2(k+2)$ parity chunks denoted by $P_{1,1}, P_{1,2}; \ldots; P_{k+2,1}, P_{k+2,2}$ using Reed-Solomon codes, such that any $2k$ out of $2(k+2)$ chunks are decodable to the original file. Note that the number of chunks per file is polynomial with k; compared to some state-of-the-art MSR codes [27], this causes small sub-packetization which can reduce the access overhead to chunks. Each node i (where $i = 1, 2, \ldots, k+2$) stores two chunks $P_{i,1}$ and $P_{i,2}$. Clearly, the generated parity chunks satisfy the MDS property, i.e., for any k out of n nodes $\{s_1, \ldots, s_k\} \subset \{1, \ldots, k+2\}$, the $2k$ parity chunks $\{P_{s_1,1}, P_{s_1,2}; \ldots; P_{s_k,1}, P_{s_k,2}\}$ are decodable. In addition, the generated parity chunks also satisfy the erMDS property (see Definition 6), i.e., for any $k+1$ nodes $s_1, \ldots, s_{k+1}$, we can always select some specific chunks $P_{s_1, f(s_1)}, \ldots, P_{s_{k+1}, f(s_{k+1})}$ such that any RBC consisting of them is decodable. Here, we need to find and record such $k+1$ specific chunks for any $k+1$ nodes. For illustrative purposes, we let $f(s_i) = 1$, where $i = 1, 2, \ldots, k+1$, so we record the chunks $\{P_{s_1,1}, \ldots, P_{s_{k+1},1}\}$.

2) The First Round of Repair: Suppose without loss of generality that node 1 fails and then is repaired by the following two steps.

Step 1 (Chunk Selection): We select $k+1$ chunks $P_{2,1}, \ldots, P_{k+2,1}$ that are recorded when the file is stored.

Step 2 (Coefficient Construction): For each selected chunk $P_{i^*,1}$ ($i^* = 2, \ldots, k+2$), we compute $2k$ encoding coefficients $\lambda^{(i^*)}_{i,j}$ ($i = 2, \ldots, k+2,\ i \neq i^*,\ j = 1, 2$) which satisfy

$$P_{i^*,1} = \sum_{i=2, i \neq i^*}^{k+2} \sum_{j=1}^{2} \lambda^{(i^*)}_{i,j} P_{i,j}. \tag{3}$$

Each parity chunk is a linear combination of $k(n-k) = 2k$ native chunks (see Section III). By equating the coefficients that are multiplied with the $2k$ native chunks on both left and right sides of Equation (3), we obtain $2k$ equations, which allow us to solve for $\lambda^{(i^*)}_{i,j}$.

Next we need to construct the encoding coefficients $\gamma^i_{1,1}$ and $\gamma^i_{1,2}$ (See Equation (1)) by satisfying the following inequalities (4), (5), and (6):

$$\gamma^i_{1,1}\gamma^j_{1,2} \neq \gamma^i_{1,2}\gamma^j_{1,1}, \tag{4}$$

where $i \neq j$ and $i, j = 2, 3, \ldots, k+2$;

$$\gamma^i_{1,2} + \gamma^{i^*}_{1,2}\lambda^{(i^*)}_{i,1} \neq 0, \tag{5}$$

where $i \neq i^*$ and $i, i^* \in \{2, \ldots, k+2\}$; and

$$\begin{aligned}(\gamma^i_{1,1} + \gamma^{i^{**}}_{1,1}\lambda^{(i^{**})}_{i,1})(\gamma^{i^*}_{1,2} + \gamma^{i^{**}}_{1,2}\lambda^{(i^{**})}_{i^*,1}) \\ \neq (\gamma^{i^*}_{1,1} + \gamma^{i^{**}}_{1,1}\lambda^{(i^{**})}_{i^*,1})(\gamma^i_{1,2} + \gamma^{i^{**}}_{1,2}\lambda^{(i^{**})}_{i,1}),\end{aligned} \tag{6}$$

where i, i^* and i^{**} are distinct, $i, i^*, i^{**} \in \{2, \ldots, k+2\}$. Lastly, we regenerate new chunks $P'_{1,1}$ and $P'_{1,2}$

as follows:

$$P'_{1,1} = \gamma_{1,1}^{2} P_{2,1} + \gamma_{1,1}^{3} P_{3,1} + \ldots + \gamma_{1,1}^{k+2} P_{k+2,1}, \qquad (7)$$

$$P'_{1,2} = \gamma_{1,2}^{2} P_{2,1} + \gamma_{1,2}^{3} P_{3,1} + \ldots + \gamma_{1,2}^{k+2} P_{k+2,1}. \qquad (8)$$

Note that Equation (4) is used for maintaining the MDS property, while Equations (5) and (6) are used for maintaining the erMDS property.

3) The m^{th} Round of Repair ($m > 1$): If the failed node in the m^{th} round of repair is the repaired node in the $(m-1)^{th}$ round of repair, then we just repeat the $(m-1)^{th}$ repair. Otherwise, we first select the $k+1$ chunks such that they are *different* from those selected in the $(m-1)^{th}$ round of repair. Then similar to the first round of repair, we generate the coefficients that satisfy inequalities in (4), 5), and (6). Finally, we regenerate the new chunks accordingly as (7) and (8).

B. Proof of Correctness of Semi-Deterministic PMSR Codes

We now prove the correctness of the semi-deterministic PMSR codes in Section VI. Since the file is stored by Reed-Solomon codes, any $2k$ out of $2(k+2)$ (parity) chunks are decodable to the original file. Therefore, the MDS and erMDS properties are satisfied. Now, we show that the MDS and erMDS properties are always satisfied after each round of repair, based on our chunk selection and coefficient construction.

1) The First Round of Repair: Let $\mathcal{U}_1 = \{P'_{1,1}, P'_{1,2}; P_{2,1}, P_{2,2}; \ldots; P_{n,1}, P_{n,2}\}$ be the set of all chunks after the first round of repair (for failed node 1). Next we prove that $\mathcal{U}_1$ still satisfies both the MDS and erMDS properties.

($\mathcal{U}_1$ *satisfies the MDS property*) Since the file is stored with Reed-Solomon Codes, all the chunks of any k out of nodes $2, \ldots, k+2$ are obviously decodable. Thus, we only need to check whether the chunks of the repaired node 1 and any $k-1$ of nodes $2, \ldots, k+2$ are decodable. Take the repaired node 1 and nodes $2, \ldots, k$ for instance. Denote the $2k$ chunks of them by $\mathcal{V} = \{P'_{1,1}, P'_{1,2}; P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}\}$. Due to Equations (7) and (8), span($\mathcal{V}$) = span($\gamma_{1,1}^{k+1} P_{k+1,1} + \gamma_{1,1}^{k+2} P_{k+2,1}, \gamma_{1,2}^{k+1} P_{k+1,1} + \gamma_{1,2}^{k+2} P_{k+2,1}; P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}$). Due to inequality (4), we can find coefficients that satisfy $\gamma_{1,1}^{k+1}\gamma_{1,2}^{k+2} \neq \gamma_{1,2}^{k+1}\gamma_{1,1}^{k+2}$. So span($\mathcal{V}$) = span($P_{k+1,1}, P_{k+2,1}; P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}$). Based on the erMDS property, the right hand side of the above equation is decodable because it contains $P_{2,1}, P_{3,1}, \ldots, P_{k+2,1}$ for nodes $2, \ldots, k+2$. So $\mathcal{V}$ is also decodable.

($\mathcal{U}_1$ *satisfies the erMDS Property*) Since the file is stored with Reed-Solomon Codes, there already exist $k+1$ chunks $P_{2,1}, \ldots, P_{k+2,1}$ such that any RBC consisting of them is decodable due to the erMDS property that we enforce when we store the file. Thus, we only need to check whether for the repaired node 1 and any k of nodes $2, \ldots, k+2$, there always exist $k+1$ chunks such that by choosing one chunk from each such node, any RBC consisting of them is decodable. Without loss of generality, we just consider the case for the repaired node 1 and nodes $2, \ldots, k+1$ for simplicity.

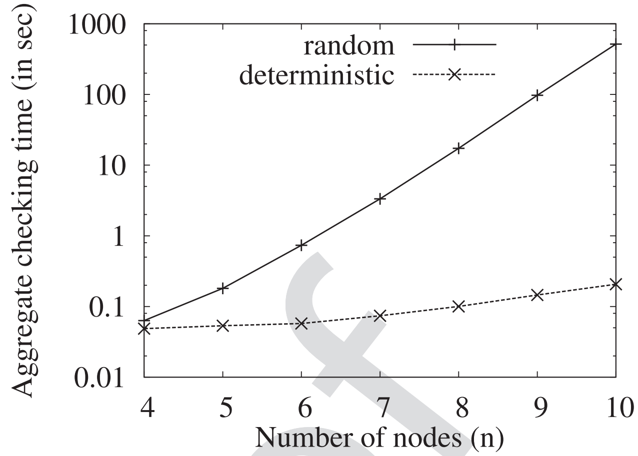

Fig. 3. Aggregate checking time of 50 rounds of repair (y-axis is in log scale).

Here, we select the $k+1$ chunks in the way that they are *distinct* from those selected for the first round of repair. In this case, we collect $\mathcal{F}_1 = \{P'_{1,2}, P_{2,2}, \ldots, P_{k+1,2}\}$ (note: either $P'_{1,1}$ or $P'_{1,2}$ is fine). Next we show the constructed $\gamma_{1,1}^{i}$ and $\gamma_{1,2}^{i}$ make any RBC consisting of $\mathcal{F}_1$ decodable. Since the repaired node 1 may offer one or two chunks to an RBC, we consider two cases.

Case 1: The repaired node 1 only offers one chunk. Then the RBC needs another $k-1$ nodes (e.g., nodes $2, \ldots, k$) to offer all their chunks and another one node (e.g., node $k+1$) to offer one chunk. To make the RBC include $\mathcal{F}_1$, we have the repaired node 1 offering $P'_{1,2}$ and node $k+1$ offering $P_{k+1,2}$. Then the RBC is $\mathcal{R}_1 = \{P'_{1,2}; P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}; P_{k+1,2}\}$. By Equation (8), span($\mathcal{R}_1$) = span ($\{\gamma_{1,2}^{k+1} P_{k+1,1} + \gamma_{1,2}^{k+2} P_{k+2,1}; P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}; P_{k+1,2}\}$).

Based on the MDS property, we consider a decodable collection $\mathcal{Z} = \{P_{2,1}, P_{2,2}; \ldots; P_{k+1,1}, P_{k+1,2}\}$. Then $P_{k+2,1}$ is a linear combination of $\mathcal{Z}$, and can be expressed as

$$P_{k+2,1} = \sum_{i=2}^{k+1} \sum_{j=1}^{2} \lambda_{i,j}^{k+2} P_{i,j}, \qquad (9)$$

based on Equation (3). Thus, span($\mathcal{R}_1$) = span ($\{(\gamma_{1,2}^{k+1} + \gamma_{1,2}^{k+2}\lambda_{k+1,1}^{k+2}) P_{k+1,1}; P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}; P_{k+1,2}\}$). Due to inequality (5), we can find coefficients that satisfy $\gamma_{1,2}^{k+1} + \gamma_{1,2}^{k+2}\lambda_{k+1,1}^{k+2} \neq 0$. Thus, span($\mathcal{R}_1$) = span ($\{P_{2,1}, P_{2,2}; \ldots; P_{k,1}, P_{k,2}; P_{k+1,1}, P_{k+1,2}\}$). The right hand side of the above equation is decodable due to the MDS property. So $\mathcal{R}_1$ is also decodable.

Case 2: The repaired node 1 offers two chunks. Then the RBC needs another $k-2$ nodes (e.g., nodes $2, \ldots, k-1$) to offer all their chunks and another two nodes (e.g., nodes k and $k+1$) to offer one chunk. To make the RBC include $\mathcal{F}_1$, we have nodes k and $k+1$ offering $P_{k,2}$ and $P_{k+1,2}$, respectively. Then the RBC is $\mathcal{R}_2 = \{P'_{1,1}, P'_{1,2}; P_{2,1}, P_{2,2}; \ldots; P_{k-1,1}, P_{k-1,2}; P_{k,2}; P_{k+1,2}\}$. Due to Equations (7) and (8), span ($\mathcal{R}_2$) = span ($\{\gamma_{1,1}^{k} P_{k,1} + \gamma_{1,1}^{k+1} P_{k+1,1} + \gamma_{1,1}^{k+2} P_{k+2,1}, \gamma_{1,1}^{k} P_{k,1} + \gamma_{1,1}^{k+1} P_{k+1,1} + \gamma_{1,1}^{k+2} P_{k+2,1}, P_{2,1}, P_{2,2}; \ldots; P_{k-1,1}, P_{k-1,2}; P_{k,2}; P_{k+1,2}\}$).

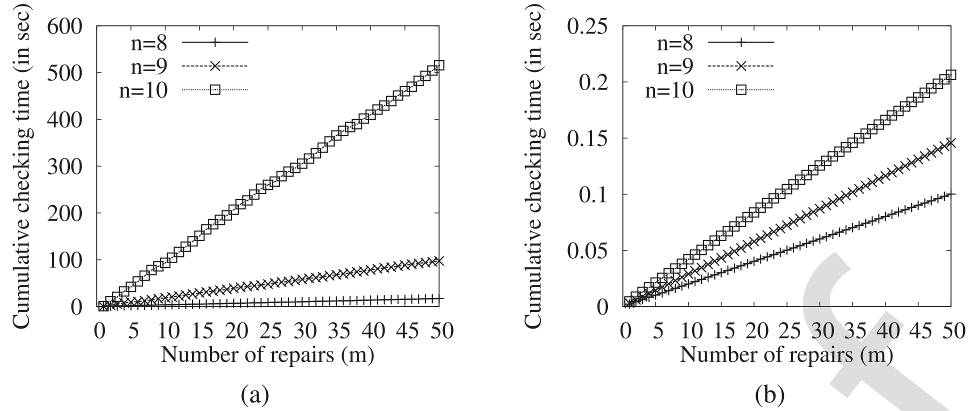

Fig. 4. Cumulative checking time of m rounds of repair. (a) random PMSR codes. (b) semi-deterministic PMSR codes.

By Equation (9), span $(\mathcal{R}_2)$ = span $(\{(\gamma_{1,1}^{k} + \gamma_{1,1}^{k+2}\lambda_{k,1}^{k+2})P_{k,1} + (\gamma_{1,1}^{k+1} + \gamma_{1,1}^{k+2}\lambda_{k+1,1}^{k+2})P_{k+1,1},\ (\gamma_{1,2}^{k} + \gamma_{1,2}^{k+2}\lambda_{k,1}^{k+2})P_{k,1} + (\gamma_{1,2}^{k+1} + \gamma_{1,2}^{k+2}\lambda_{k+1,1}^{k+2})P_{k+1,1};\ P_{2,1}, P_{2,2}; \ldots; P_{k-1,1}, P_{k-1,2}; P_{k,2}; P_{k+1,2}\})$.

Due to inequality (6), we can find coefficients which satisfy $(\gamma_{1,1}^{k} + \gamma_{1,1}^{k+2}\lambda_{k,1}^{k+2})(\gamma_{1,2}^{k+1} + \gamma_{1,2}^{k+2}\lambda_{k+1,1}^{k+2}) \neq (\gamma_{1,1}^{k+1} + \gamma_{1,1}^{k+2}\lambda_{k+1,1}^{k+2})(\gamma_{1,2}^{k} + \gamma_{1,2}^{k+2}\lambda_{k,1}^{k+2})$, then span$(\mathcal{R}_2)$ = span $(\{P_{2,1}, P_{2,2}; \ldots; P_{k+1,1}, P_{k+1,2}\})$.

The right hand side of the above equation is decodable due to the MDS property. So $\mathcal{R}_2$ is also decodable.

2) The m^{th} Repair ($m > 1$): Take $m = 2$ for instance. Suppose without loss of generality that node $k+2$ fails, then we select $\{P'_{1,2}, P_{2,2}, \ldots, P_{k+1,2}\}$ which are distinct from those in the first round of repair. We can observe that in fact this set is $\mathcal{F}_1$ in the first round of repair. As mentioned above, any RBC consisting of $\mathcal{F}_1$ is decodable. So $\mathcal{F}_1$ can be used for the second round of repair. Then we can generate the coefficients that satisfy the similar inequalities as (4), (5), and (6). The proof of correctness is similar to $m = 1$ and thus omitted.

VII. Evaluation

In this section, we evaluate the repair performance of two implementations of PMSR codes with $k = n - 2, r = 1$ in a real multiple cloud storage system: (i) *random PMSR codes*, which use random chunk selection in repair and is used in NCCloud [9] and (ii) *semi-deterministic PMSR codes*, which use deterministic chunk selection proposed in Section VI. We show that our proposed semi-deterministic PMSR codes can significantly reduce the time required to regenerate parity chunks in repair.

We implement both versions of PMSR codes in C. We implement finite-field arithmetic operations over a Galois Field GF(2^8) based on the standard table lookup approach [16]. We conduct our evaluation on a server running on an Intel CPU at 2.4GHz. We consider different values of n (i.e., the number of nodes). For each n, we first apply Reed-Solomon codes to generate the encoding coefficients that will be used to encode a file into parity chunks before uploading. In each round of repair, we randomly pick a node to fail. We then repair the failed node using two-phase checking, based on either random or semi-deterministic PMSR code implementations. The failed node that we choose is different from that of the previous round of repair, so as to ensure a different chunk selection in each round of repair. We conduct 50 rounds of repair in each evaluation run. We conduct a total of 30 runs over different seeds for each n.

The metric we are interested in is the checking time spent on determining if the chunks selected from surviving nodes can be used to regenerate the lost chunks. We do not measure the times of reading or writing chunks, as they are the same for both random and semi-deterministic PMSR codes. Instead, we focus on measuring the processing time of two-phase checking in each round of repair. It is important to note that two-phase checking only operates on encoding coefficients, and is independent of the size of the file being encoded. Note that we do not specifically optimize our encoding operations, but we believe our results provide fair comparison of both random and semi-deterministic PMSR codes using our baseline implementations.

Figure 3 first depicts the aggregate checking times for a total of 50 rounds of repair versus the number of nodes when using random and semi-deterministic PMSR codes. The aggregate checking time of random PMSR codes is small when n is small (e.g., less than 1 second for $n \leq 6$), but exponentially increases as n is large. On the other hand, the aggregate checking time of semi-deterministic PMSR codes is significantly small (e.g., within 0.2 seconds for $n \leq 10$).

Our investigation finds that the checking time of random PMSR codes increases dramatically as the value of n increases. For example, when $n = 12$ (not shown in our figures), we find that the repair operation of our random PMSR code implementation still cannot return a right set of regenerated chunks after running for two hours. In contrast, our semi-deterministic PMSR codes can return a solution within 0.5 seconds.

To further examine the significant performance overhead of random PMSR codes, Figures 4 and 5 show the cumulative checking time and number of two-phase checking operations performed for m rounds of repair, respectively, for $n = 8, 9, 10$. We observe that the checking time in each round of repair remains almost the same regardless of the number of repairs that have been performed; in other words, the repair performance remains stable after a number of rounds

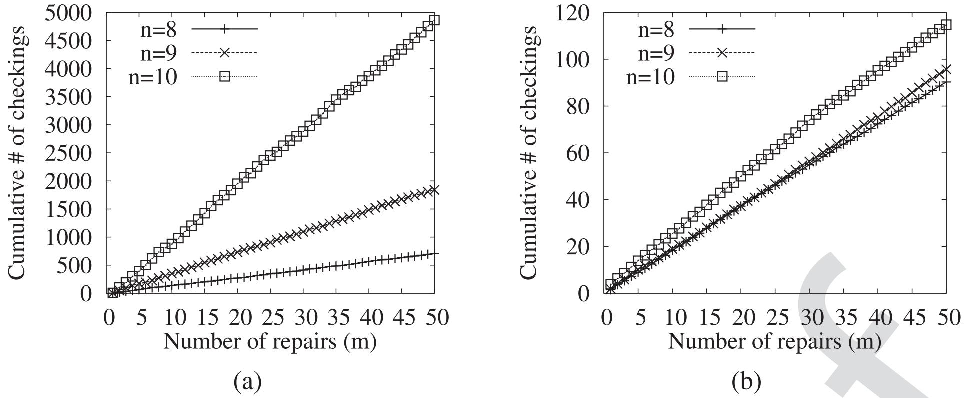

Fig. 5. Cumulative number of two-phase checkings of m rounds of repair. (a) random PMSR codes. (b) semi-deterministic PMSR codes.

of repairs. We note that random PMSR codes incur a fairly large but constant number of two-phase checking operations in each round of repair. For example, for $n = 10$, each round of repair takes around 100 iterations of two-phase checking (see Figure 5(a)). On the other hand, semi-deterministic PMSR codes significantly reduce the number of iterations of two-phase checking (e.g., less than 2.5 on average for $n = 10$). *In summary, our evaluation results show that semi-deterministic PMSR codes significantly reduce the two-phase checking overhead of ensuring that the MDS property is preserved during repair.*

VIII. Conclusions

This paper formulates an uncoded repair problem based on proxy-assisted minimum storage regenerating (PMSR) codes. We formally prove the existence of PMSR codes with uncoded repair against both single and concurrent failures matching the lower bound of repair bandwidth, and provide a semi-deterministic family of PMSR code construction. We also show via our evaluation that our semi-deterministic PMSR codes significantly reduce the repair time overhead of random PMSR codes. Our theoretical results validate the correctness of the NCCloud implementation [9] and design a more generic family of PMSR codes for repairing concurrent node failures. We also demonstrate the feasibility of preserving the benefits of network coding in minimizing the repair bandwidth with uncoded repair.

Appendix A
Proof of Lemma 2

Proof: Suppose that T is connected to t (where $t \leq r$) new nodes $X'_1, \cdots, X'_t$ and $k - t$ surviving nodes $X_{n-k+t+1}, \cdots, X_n$ for reconstructing the original file. By excluding edges with infinite capacities, a possible cut can fall into one of three cases:

- Cut $\mathcal{C}_1$: It spans across all surviving nodes, i.e., it contains all edges from X_i^{in} to X_i^{out}, where $r + 1 \leq i \leq n$.
- Cut $\mathcal{C}_2$: It spans across some surviving nodes and the connections between the surviving nodes and the proxy, i.e., it contains some edges from X_i^{in} to X_i^{out} and some edges from X_i^{out} to Y^{in}, where $r + 1 \leq i \leq n$.
- Cut $\mathcal{C}_3$: It spans across some surviving nodes and the proxy, i.e., it contains some edges from X_i^{in} to X_i^{out}, where $r + 1 \leq i \leq n$ and the edge from Y^{in} to Y^{out}.

Figure 1 shows the cuts $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ in $\mathcal{G}$. Let Λ_1, Λ_2, and Λ_3 denote the capacities of $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, respectively. We now analyze the capacity of each cut as follows.

A. Derivations of Λ_1

Since $n - k \geq r$ (otherwise, there is data loss when r nodes fail), we have:

$$\begin{aligned} \Lambda_1 &= (n-r) \cdot M/k \\ &\geq k \cdot M/k \\ &= M. \end{aligned}$$

B. Derivations of Λ_2

Let w be the number of edges from some X_i^{out} (where $r + 1 \leq i \leq n$) to Y^{in}.

$$\Lambda_2 = w \cdot \beta + (n - r - w) \cdot M/k. \tag{10}$$

Since Λ_2 of every possible min-cut is at least M, Equation (10) implies that for all variants of $\mathcal{G}$,

$$\beta \geq M/k \cdot (1 - \frac{n-k-r}{w}). \tag{11}$$

Let β' be the right side of Equation (11). Then for all variants of $\mathcal{G}$, Equation (11) can be reduced to:

$$\beta \geq \max\{\beta'\}. \tag{12}$$

We now derive $\max\{\beta'\}$. Since T is connected to at most r new nodes, we have $t \leq r$. Also, since T is connected to $k - t$ surviving nodes, we have $w \leq (n - r) - (k - t)$. Thus, we have $w \leq n - k$. When $w = n - k$, $\max\{\beta'\}$ is achieved. That is,

$$\max\{\beta'\} = \frac{rM}{k(n-k)}. \tag{13}$$

By Equations (12) and (13), we have

$$\beta \geq \frac{rM}{k(n-k)}.$$

C. Derivations of Λ_3

Since T is connected to $k-t$ surviving nodes, we have

$$\Lambda_3 \geq r \cdot M/k + (k-t) \cdot M/k. \tag{14}$$

Since $t \leq r$, Equation (14) implies that

$$\Lambda_3 \geq M.$$

D. Summary

Clearly, both Λ_1 and Λ_3 are always at least M, independent of β. Also, when $\Lambda_2 \geq M$, we have a lower bound of β equal to $\frac{rM}{k(n-k)}$. This concludes the proof of Lemma 2. □

Appendix B
Proof of Lemma 6

Proof: Without loss of generality, let node 1 and node 2 be the failed nodes. Suppose that there are x repaired nodes selected in Step 2 of RBC construction and y repaired nodes selected in Step 3 of RBC construction.

There are five cases to construct an RBC which contains chunks of the repaired node 1 and node 2: (1) $x = 2, y = 0$; (2) $x = 1, y = 1$;(3) $x = 0, y = 2$ (4) $x = 1, y = 0$; (5) $x = 0, y = 1$. For the same reason as stated by Lemma 3, we do not consider the RBCs which do not contain any chunks of the repaired nodes.

Let $\mathcal{P}$ be the set of chunks chosen in Step 2 of Definition 2 excluding those from the repaired nodes. Similar to the proof of Lemma 3, we infer that all the chunks of $\mathcal{R}$ are linear combinations of chunks in $\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}$, which only contain chunks from surviving nodes.

Since $\mathcal{P}$ contains $n-k = 3$ chunks from each of $k-r-x = k-2-x$ surviving nodes, $\mathcal{P}$ has $3(k-2-x)$ chunks of the surviving nodes, i.e., $|\mathcal{P}| = 3(k-2-x)$;

Since $\mathcal{F}$ selects $r = 2$ chunks from each surviving nodes and $\mathcal{P}$ has all the chunks from $k-r-x = k-2-x$ out of all the surviving nodes, $\mathcal{F}$ and $\mathcal{P}$ have $2(k-2-x)$ identical common chunks of the surviving nodes, i.e., $|\mathcal{F} \cap \mathcal{P}| = 2(k-2-x)$.

Since $\mathcal{Q}$ contains $r = 2$ chunks from each of $n-k-y = 3-y$ surviving nodes, $\mathcal{Q}$ has $2(3-y)$ chunks of the surviving nodes, i.e., $|\mathcal{Q}| = 2(3-y)$.

According to the given conditions, we can easily have the following equalities: $|\mathcal{F}| = 2(n-2)$, $|\mathcal{P} \cap \mathcal{Q}| = |\mathcal{F} \cap \mathcal{P} \cap \mathcal{Q}| = 0$. Thus, we have

$$\begin{aligned} |\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}| &= |\mathcal{F}| + |\mathcal{P}| + |\mathcal{Q}| - |\mathcal{F} \cap \mathcal{P}| \\ &\quad - |\mathcal{F} \cap \mathcal{Q}| - |\mathcal{P} \cap \mathcal{Q}| + |\mathcal{F} \cap \mathcal{P} \cap \mathcal{Q}| \\ &= 3k + (6 - x - 2y) - |\mathcal{F} \cap \mathcal{Q}|. \end{aligned} \tag{15}$$

If an RBC is an LDC, which means $\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}$ are linear combinations of less than $k(n-k)$ chunks from the surviving nodes, then $|\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}| < 3k$. There are five cases of different values of x and y as follows:

Case 1: When $x = 2, y = 0$, we have $6 - x - 2y = 4$. Hence, by Equation (15), when an RBC of Case 1 is an LDC, we can obtain: $|\mathcal{F} \cap \mathcal{Q}| \geq 5$.

Case 2: When $x = 1, y = 1$, we have $6 - x - 2y = 3$. Hence, by Equation (15), when an RBC of Case 2 is an LDC, we can obtain: $|\mathcal{F} \cap \mathcal{Q}| \geq 4$.

Case 3: When $x = 0, y = 2$, we have $6 - x - 2y = 2$, $|\mathcal{Q}| = 2(3-y) = 2$. Hence, by Equation (15) and $|\mathcal{F} \cap \mathcal{Q}| \leq |\mathcal{Q}|$, we can obtain $|\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}| \geq 3k$, which means the RBC of case 3 is never an LDC.

Case 4: When $x = 1, y = 0$, we have $6 - x - 2y = 5$. Hence, by Equation (15), when an RBC of Case 4 is an LDC, we can obtain: $|\mathcal{F} \cap \mathcal{Q}| \geq 5$.

Case 5: When $x = 0, y = 1$, we have $6 - x - 2y = 4$, $|\mathcal{Q}| = 2(3-y) = 4$. Hence, by Equation (15) and $|\mathcal{F} \cap \mathcal{Q}| \leq |\mathcal{Q}|$, we can obtain $|\mathcal{F} \cup \mathcal{P} \cup \mathcal{Q}| \geq 3k$, which means the RBC of case 5 is never an LDC.

Therefore, Lemma 6 holds. □

Appendix C
Proof of Lemma 7

Proof: Without loss of generality, suppose that we construct an RBC $\mathcal{R}$ by selecting the chunks from nodes $3, \ldots, n$ (see Step 1 of Definition 2), and that $\mathcal{H}$ be the set of $2(n-2)$ chunks selected from nodes $3, \ldots, n$ (two chunks from each node). We prove the existence of $\mathcal{H}$ such that if $\mathcal{R}$ contains $\mathcal{H}$ (i.e., $\mathcal{H} \subset \mathcal{R}$), then $\mathcal{R}$ is decodable. There are three cases about node 1 and node 2: (1) node 1 and node 2 are the repaired nodes in the m^{th} round of repair; (2) node 1 and node 2 are not the repaired nodes in the m^{th} round of repair; (3) node 1 (or node 2) is the repaired node while node 2 (node 1) is not the repaired node in the m^{th} round of repair. We discuss them as follows:

Case 1: If node 1 and node 2 are the repaired nodes in the m^{th} round of repair, then $\mathcal{R}$ is never an LDC (by Definition 3) similar to that in Lemma 4. Since the rMDS property is satisfied by our assumption, $\mathcal{R}$ is decodable (by Definition 4).

Case 2: If node 1 and node 2 are not the repaired nodes in the m^{th} round of repair, then without loss of generality, let node 3 and node 4 be the repaired nodes and the new parity chunks are $P'_{3,1}$, $P'_{3,2}$, $P'_{3,3}$, $P'_{4,1}$, $P'_{4,2}$ and $P'_{4,3}$. By the PMSR code design, the chunks of node 3 and node 4 are linearly combined by two chunks in each of nodes $1, 2, 5, \ldots, n$. We denote these chunks by $\mathcal{F} = \{P_{1,f_1(1)}, P_{1,f_2(1)}, P_{2,f_1(2)}, P_{2,f_2(2)}, P_{5,f_1(5)}, P_{5,f_2(5)}, \ldots, P_{n,f_1(n)}, P_{n,f_2(n)}\}$. Since each node has $n-k = 3$ chunks, we can construct $\mathcal{H} = \{P'_{3,g_1(3)}, P'_{3,g_2(3)}, P'_{4,g_1(4)}, P'_{4,g_2(4)}, P_{5,g_1(5)}, P_{5,g_2(5)}, \ldots, P_{n,g_1(n)}, P_{n,g_2(n)}\}$ such that $g_1(i) \neq f_1(i)$ and $g_2(i) = f_2(i)$ for $i = 5, \ldots, n$ (while $g_3(1)$, $g_3(2)$,$g_4(1)$ and $g_4(2)$ can be randomly picked). Let $\mathcal{Q}$ be the set of chunks chosen in Step 3 of Definition 2 excluding those from the two repaired nodes. If $\mathcal{R}$ contains $\mathcal{H}$, then $\mathcal{Q}$ and $\mathcal{F}$ only contain $1 \cdot (n-k) = 3$ identical common chunks of the surviving nodes. By Lemma 6, $\mathcal{R}$ is not an LDC. Since the rMDS property is satisfied, $\mathcal{R}$ is decodable.

Case 3: The proof is similar to that of Case 2. So omitted. □

Appendix D
Proof of Theorem 2

Proof: We prove by induction on m. Initially, we use Reed-Solomon codes to encode a file into $n(n-k) = 3n$

chunks that satisfy both the MDS and rMDS properties. Suppose that after the m^{th} round of repair, both the MDS and rMDS properties are satisfied (this is our induction hypothesis).

Let $\mathcal{U}_m = \{P_{1,1}, P_{1,2}, P_{1,3}; \ldots; P_{k+3,1}, P_{k+3,2}, P_{k+3,3}\}$ be the current set of chunks after the m^{th} round of repair. In the $(m+1)^{th}$ round of repair, without loss of generality, let node 1 and node 2 be the failed nodes to be repaired.

Since $\mathcal{U}_m$ satisfies the rMDS property, we use $\mathcal{F}$ to repair node 1 and node 2. Suppose that the repaired node 1 and node 2 have the new chunks $\{P'_{1,1}, P'_{1,2}, P'_{1,3}\}$ and $\{P'_{2,1}, P'_{2,2}, P'_{2,3}\}$, respectively. Then

$$P'_{i',j'} = \sum_{i=3}^{k+3} \gamma^{i,1}_{i',j'} P_{i,f_1(i)} + \gamma^{i,2}_{i',j'} P_{i,f_2(i)}, \quad \text{for } i' = 1,2 \text{ and } j' = 1,2,3. \tag{16}$$

Here $\gamma^{i,1}_{i',j'}$ and $\gamma^{i,2}_{i',j'}$ denote the encoding coefficients for the two retrieved chunks of the surviving node i to generate the the j'^{th} chunk of the new node i'. Next we prove that we can always tune $\gamma^{i,1}_{i',j'}$ and $\gamma^{i,2}_{i',j'}$ in $\mathbb{F}_q$ in such a way that the set of chunks in the $(m+1)^{th}$ round of repair $\mathcal{U}_{m+1} = \{P'_{1,1}, P'_{1,2}, P'_{1,3}; P'_{2,1}, P'_{2,2}, P'_{2,3}; P_{3,1}, P_{3,2}, P_{3,3}; \ldots; P_{k+3,1}, P_{k+3,2}, P_{k+3,3}\}$ still satisfies both MDS and rMDS properties. The proof consists of two parts.

Part I $\mathcal{U}_{m+1}$ Satisfies the MDS Property:

The proof of Part I is similar to that in Theorem 1. So omitted.

Part II $\mathcal{U}_{m+1}$ Satisfies the rMDS Property:

By Definition 4, we need to prove that all the RBCs of $\mathcal{U}_{m+1}$ except the LDCs are decodable. By Definition 2, we only consider the following five cases of RBCs which contain the chunks of the repaired node 1 and node 2 due to the same reason stated by Lemma (3): (Case 1) The repaired node 1 and node 2 are selected in Step 2; (Case 2) The repaired node 1 is selected in Step 1 and node 2 is selected in Step 2; (Case 3) The repaired node 1 and node 2 are selected in Step 3; (Case 4) The repaired node 1 is selected in Step 2 while the repaired node 2 is selected neither in Step 2 nor in Step 3; (Case 5) The repaired node 1 is selected in Step 3 while the repaired node 2 is selected neither in Step 2 nor in Step 3. Due to the similarity of the proofs of all the Cases, we only prove Case 1 as follows:

Suppose the repaired node 1 and node 2 are selected in Step 2 (Case 1). In Step 1, an RBC needs to randomly select $(n-r)-2 = k-1$ additional surviving nodes, say $\{s_1, \ldots, s_{k-1}\} \subseteq \{3, \ldots, n\}$. Then in Step 2, the RBC needs to select $(k-r)-2 = k-4$ additional nodes if $k > 4$, say nodes $s_1, \ldots, s_{k-4}$. The cases with $k \leq 4$ are similar to but easier than those with $k > 4$ and thus are omitted. Finally, in Step 3, the RBC chooses six chunks, denoted by $\{P_{s_{k-3},g_1(s_{k-3})}, P_{s_{k-3},g_2(s_{k-3})}\}$, $\{P_{s_{k-2},g_1(s_{k-2})}, P_{s_{k-2},g_2(s_{k-2})}\}$ and $\{P_{s_{k-1},g_1(s_{k-1})}, P_{s_{k-1},g_2(s_{k-1})}\}$ from the remaining three nodes s_{k-3}, s_{k-2} and s_{k-1}, respectively. Without loss of generality, let $(s_1, \ldots, s_{k-4}) = (3, \ldots, k-2)$ and $(s_{k-3}, s_{k-2}, s_{k-1}) = (k-1, k, k+1)$.

Denote the RBC by

$$\begin{aligned}\mathcal{R}_1 = &\{P'_{1,1}, P'_{1,2}, P'_{1,3}; P'_{2,1}, P'_{2,2}, P'_{2,3}\}\\ &\cup \{P_{3,1}, P_{3,2}, P_{3,3}; \ldots; P_{k-2,1}, P_{k-2,2}, P_{k-2,3}\}\\ &\cup \{P_{k-1,g_1(k-1)}, P_{k-1,g_2(k-1)}\}\\ &\cup \{P_{k,g_1(k)}, P_{k,g_2(k)}\}\\ &\cup \{P_{k+1,g_1(k+1)}, P_{k+2,g_2(k+1)}\}.\end{aligned}$$

In addition, by Equation (16), the chunks of $\mathcal{R}_1$ are linear combinations of a set of chunks denoted by

$$\begin{aligned}\mathcal{X}_1 = &\{P_{3,1}, P_{3,2}, P_{3,3}; \ldots; P_{k-2,1}, P_{k-2,2}, P_{k-2,3}\}\\ &\cup \{P_{k-1,g_1(k-1)}, P_{k-1,g_2(k-1)}, P_{k-1,f_1(k-1)}, P_{k-1,f_2(k-1)}\}\\ &\cup \{P_{k,g_1(k)}, P_{k,g_2(k)}, P_{k,f_1(k)}, P_{k,f_2(k)}\}\\ &\cup \{P_{k+1,g_1(k+1)}, P_{k+1,g_2(k+1)}, P_{k+1,f_1(k+1)}, P_{k+1,f_2(k+1)}\}\\ &\cup \{P_{k+2,f_1(k+2)}, P_{k+2,f_2(k+2)}\}\\ &\cup \{P_{k+3,f_1(k+3)}, P_{k+3,f_2(k+3)}\}.\end{aligned}$$

Note that because there are $n-k = 3$ chunks in node k-1, there are at least one identical chunk between $\{P_{k-1,g_1(k-1)}, P_{k-1,g_2(k-1)}\}$ and $\{P_{k-1,f_1(k-1)}, P_{k-1,f_2(k-1)}\}$, so we suppose that $P_{k-1,g_2(k-1)} = P_{k-1,f_2(k-1)}$ without loss of generality. Thus, we can consider that node k-1 contains three chunks $\{P_{k-1,g_1(k-1)}, P_{k-1,f_1(k-1)}, P_{k-1,f_2(k-1)}\}$ as shown in Equation (17); and so do the nodes k and k+1.

$$\begin{aligned}\mathcal{X}_1 = &\{P_{3,1}, P_{3,2}, P_{3,3}; \ldots; P_{k-2,1}, P_{k-2,2}, P_{k-2,3}\}\\ &\cup \{P_{k-1,g_1(k-1)}, P_{k-1,f_1(k-1)}, P_{k-1,f_2(k-1)}\}\\ &\cup \{P_{k,g_1(k)}, P_{k,f_1(k)}, P_{k,f_2(k)}\}\\ &\cup \{P_{k+1,g_1(k+1)}, P_{k+1,f_1(k+1)}, P_{k+1,f_2(k+1)}\}\\ &\cup \{P_{k+2,f_1(k+2)}, P_{k+2,f_2(k+2)}\}\\ &\cup \{P_{k+3,f_1(k+3)}, P_{k+3,f_2(k+3)}\}.\end{aligned} \tag{17}$$

Our goal is to show that if $\mathcal{R}_1$ is not an LDC, then it is decodable. Clearly, there are $k(n-k)+1$ chunks in the right-hand side of Equation (17), so if and only if there exist at least two out of three nodes $k-1$, k and $k+1$ (let's say they are nodes $k-1$ and k without loss of generality) satisfying that $g_1(k-1) = f_1(k-1)$ and $g_1(k) = f_1(k)$, $\mathcal{R}_1$ becomes an LDC since $\mathcal{X}_1$ can be reduced to less than $k(n-k)$ chunks of the surviving nodes after the m^{th} repair. Thus, to prove that $\mathcal{R}_1$ except the LDCs is decodable, it is equivalent to prove that $\mathcal{R}_1$ is decodable when there exists at most one out of three nodes $k-1$, k and $k+1$ (let's say it is the node $k-1$ without loss of generality) satisfying that (a) $g_1(k-1) = f_1(k-1)$, $g_1(k) \neq f_1(k)$ and $g_1(k+1) \neq f_1(k+1)$; or (b) $g_1(k-1) \neq f_1(k-1)$, $g_1(k) \neq f_1(k)$ and $g_1(k+1) \neq f_1(k+1)$.

First consider (a) and define $\overline{\mathcal{X}_1} = \mathcal{X}_1 - \{P_{k,g_1(k)}, P_{k+1,g_1(k+1)}\}$. Since $\overline{\mathcal{X}_1}$ is an RBC containing $\mathcal{F}$, by Claim 2, $\overline{\mathcal{X}_1}$ is decodable. Therefore, $\{P_{k,g_1(k)}, P_{k+1,g_1(k+1)}\}$ can be seen as a linear combination of $\overline{\mathcal{X}_1}$. Obviously, we can also say $\mathcal{X}_1$ is a linear combination of $\overline{\mathcal{X}_1}$. Therefore, $\mathcal{R}_1$ is also linear combination of the decodable collection $\overline{\mathcal{X}_1}$.

Similarly, we can also prove that all the RBCs of Cases 2,3,4,5 can be reduced to linear combination of a decodable RBC containing $\mathcal{F}$ by Claim 2. Then we can use the similar method in Part I of Theorem (1) to prove that for all possible $\{s_1, \ldots, s_{k-1}\} \subseteq \{3, \ldots, n\}$, there always exists an assignment of $\gamma_{i',j'}^{i,1}$ and $\gamma_{i',j'}^{i,2}$ in a sufficiently large field such that all RBCs excluding the LDCs are decodable (by Lemma 5). Thus, $\mathcal{U}_{m+1}$ satisfies the rMDS Property.

Therefore, there always exists an assignment of $\gamma_{i',j'}^{i}$ in a sufficiently large field such that for for all possible $\{s_1, \ldots, s_{k-1}\} \subseteq \{3, \ldots, n\}$, both MDS and rMDS Property can be maintained. This concludes the proof of Theorem 2. $\square$

Acknowledgment

We thank the anonymous reviewers and our editor Parastoo Sadeghi for their feedback and comments that help us improve the paper. The corresponding author is Pan Zhou.

References

[1] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A case for cloud storage diversity," in *Proc. ACM SoCC*, 2010, pp. 229–240.

[2] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[3] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total recall: System support for automated availability management," in *Proc. NSDI*, 2004, p. 25.

[4] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 192–202, Feb. 1995.

[5] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-based erasure-resilient coding scheme," Int. Comput. Sci. Inst., Tech. Berkeley, CA, USA, Tech. Rep. ICSI TR-95-048, 1995.

[6] V. R. Cadambe, S. A. Jafar, and H. Maleki. (2010). "Distributed data storage with minimum storage regenerating codes—Exact and functional repair are asymptotically equally efficient." [Online]. Available: https://arxiv.org/abs/1004.4299

[7] B. Calder *et al.*, "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. ACM SOSP*, 2011, pp. 143–157.

[8] R. J. Chansler, "Data availability and durability with the hadoop distributed file system," *USENIX Assoc. Newslett.*, vol. 37, no. 1, pp. 16–22, 2012.

[9] H. C. H. Chen, Y. Hu, P. P. C. Lee, and Y. Tang, "NCCloud: A network-coding-based storage system in a cloud-of-clouds," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 31–44, Jan. 2014.

[10] B. Chun *et al.*, "Efficient replica maintenance for distributed storage systems," in *Proc. NSDI*, 2006, p. 4.

[11] P. Corbett *et al.*, "Row-diagonal parity for double disk failure correction," in *Proc. USENIX FAST*, 2004, p. 1.

[12] D. Cullina, A. G. Dimakis, and T. Ho, "Searching for minimum storage regenerating codes," presented at the 47th Annu. Allerton Conf. Commun., Control, Comput., Urbana-Champaign, IL, USA, Sep. 2009.

[13] 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.

[14] D. Ford *et al.*, "Availability in globally distributed storage systems," in *Proc. USENIX OSDI*, vol. 10. Oct. 2010, pp. 1–7.

[15] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. ACM SOSP*, 2003, pp. 29–43.

[16] K. M. Greenan, E. L. Miller, and T. J. E. Schwarz, "Optimizing Galois field arithmetic for diverse processor architectures and applications," in *Proc. IEEE MASCOTS*, Sep. 2008, pp. 1–10.

[17] Y. Hu, H. C. H. Chen, P. P. C. Lee, and Y. Tang, "NCCloud: Applying network coding for the storage repair in a cloud-of-clouds," in *Proc. FAST*, 2012, p. 21.

[18] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 268–276, Feb. 2010.

[19] C. Huang *et al.*, "Erasure coding in windows azure storage," presented at the USENIX Annu. Tech. Conf. (USENIX ATC), 2012, pp. 15–26.

[20] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 889–901, Jul. 2008.

[21] J. Huang, X. Liang, X. Qin, P. Xie, and C. Xie, "Scale-RS: An efficient scaling scheme for RS-coded storage clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1704–1717, Jun. 2015.

[22] *Intelligent RAID6 Theory Overview and Implementation*, Intel, Santa Clara, CA, USA, 2005.

[23] O. Khan, R. Burns, J. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads," in *Proc. USENIX FAST*, 2012, p. 20.

[24] J. Kubiatowicz *et al.*, "OceanStore: An architecture for global-scale persistent storage," in *Proc. ASPLOS*, 2000, pp. 1–12.

[25] R. Li, J. Lin, and P. P. C. Lee, "Enabling concurrent failure recovery for regenerating-coding-based storage systems: From theory to practice," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 1898–1911, Jul. 2015.

[26] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[27] L. Pamies-Juarez, F. Blagojevic, R. Mateescu, C. Guyot, E. E. Gad, and Z. Bandic, "Opening the chrysalis: On the real repair performance of MSR codes," in *Proc. USENIX FAST*, Feb. 2016, pp. 81–94.

[28] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proc. ACM SIGMOD*, 1988, pp. 109–116.

[29] J. S. Plank, "A tutorial on Reed–Solomon coding for fault-tolerance in RAID-like systems," *Softw.-Pract. Exper.*, vol. 27, no. 9, pp. 995–1012, Sep. 1997.

[30] K. V. Rashmi, P. Nakkiran, J. B. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage and network-bandwidth," in *Proc. 13th USENIX Conf. File Storage Technol. (FAST)*, 2015, pp. 81–94.

[31] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in *Proc. USENIX HotStorage*, 2013, pp. 1–5.

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

[33] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. Allerton Conf.*, Sep./Oct. 2009, pp. 1243–1249.

[34] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[35] R. Rodrigues and B. Liskov, "High availability in DHTs: Erasure coding vs. replication," in *Proc IPTPS*, 2005, pp. 226–239.

[36] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. Allerton*, 2010, pp. 1510–1517.

[37] M. Sathiamoorthy *et al.*, "XORing elephants: Novel erasure codes for big data," *Proc. VLDB Endowment*, vol. 6, no. 5, pp. 325–336, 2013.

[38] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1837–1852, Mar. 2012.

[39] 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.

[40] K. W. Shum and Y. Hu, "Functional-repair-by-transfer regenerating codes," in *Proc. ISIT*, 2012, pp. 1192–1196.

[41] K. W. Shum and Y. Hu, "Cooperative regenerating codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7229–7258, Nov. 2013.

[42] C. Suh and K. Ramchandran, "Exact-repair MDS code construction using interference alignment," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1425–1442, Mar. 2011.

[43] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2013.

[44] M. Vrable, S. Savage, and G. Voelker, "Cumulus: Filesystem backup to the cloud," in *Proc. USENIX FAST*, 2009, pp. 225–238.

[45] M. Vrable, S. Savage, and G. M. Voelker, "BlueSky: A cloud-backed file system for the enterprise," in *Proc. USENIX FAST*, 2012, p. 19.

[46] Z. Wang, A. G. Dimakis, and J. Bruck, "Rebuilding for array codes in distributed storage systems," in *Proc. IEEE GLOBECOM Workshops*, Dec. 2010, pp. 1905–1909.

[47] Z. Wang, I. Tamo, and J. Bruck, "On codes for optimal rebuilding access," in *Proc. Allerton*, 2011, pp. 1374–1381.

[48] H. Weatherspoon, P. Eaton, B.-G. Chun, and J. Kubiatowicz, "Antiquity: Exploiting a secure log for wide-area distributed storage," in *Proc. ACM SIGOPS/EuroSys*, 2007, pp. 371–384.

[49] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. IPTPS*, Mar. 2002, pp. 328–338.

[50] S. Wu, Y. Xu, Y. Li, and Z. Yang, "I/O-efficient scaling schemes for distributed storage systems with CRS codes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2639–2652, Sep. 2016.

[51] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 277–288, Feb. 2010.

[52] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. ISIT*, 2009, pp. 2276–2280.

[53] M. Xia, M. Saxena, M. Blaum, and D. A. Pease, "A tale of two erasure codes in HDFS," in *Proc. 13th USENIX Conf. File Storage Technol. (FAST)*, 2015, pp. 213–226.

[54] L. Xiang, Y. Xu, J. Lui, Q. Chang, Y. Pan, and R. Li, "A hybrid approach to failed disk recovery using RAID-6 codes: Algorithms and performance evaluation," *ACM Trans. Storage*, vol. 7, no. 3, 2011, Art. no. 11.

[55] Y. Zhu, P. P. C. Lee, Y. Hu, L. Xiang, and Y. Xu, "On the speedup of single-disk failure recovery in XOR-coded storage systems: Theory and practice," in *Proc. IEEE MSST*, Apr. 2012, pp. 1–12.

Yuchong Hu received the B.S. degree in Computer Science and Technology from the School of the Gifted Young, University of Science and Technology of China, Anhui, China, in 2005, and the Ph.D. degree in Computer Science and Technology from the School of Computer Science, University of Science and Technology of China, in 2010. He is currently an Associate Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests focus on improving the fault tolerance, repair and read/write performance of storage systems, which include cloud storage systems, distributed storage systems and NVM-based systems.

Patrick P. C. Lee received the B.Eng. degree (firstclass honors) in Information Engineering from the Chinese University of Hong Kong in 2001, the M.Phil. degree in Computer Science and Engineering from the Chinese University of Hong Kong in 2003, and the Ph.D. degree in Computer Science from Columbia University in 2008. He is now an Associate Professor of the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests are in various applied/systems topics including storage systems, distributed systems and networks, operating systems, dependability, and security.

Kenneth W. Shum received his B.Eng. degree in the Dept. of Information Engineering from The Chinese University of Hong Kong in 1993, and MSc and Ph.D. degrees in Dept. of Electrical Engineering from University of Southern California in 1995 and 2000, respectively. He is now a research Associate Professor with Institute of Network Coding in The Chinese University of Hong Kong. His research interests include network coding and coding for distributed storage systems.

Pan Zhou received the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2011. He was a Senior Technical Member with Oracle Inc., Boston, MA, USA, from 2011 to 2013, where he was involved in Hadoop and distributed storage systems for big data analytics at Oracle cloud Platform. He is currently an Associate Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China. His current research interests include communication and information networks, security and privacy, machine learning, and big data.