A Repair Framework for Scalar MDS Codes

Karthikeyan Shanmugam* *Student Member, IEEE*, Dimitris S. Papailiopoulos* *Student Member, IEEE*,
Alexandros G. Dimakis* *Member, IEEE*, and Giuseppe Caire† *Fellow, IEEE*
*Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX-78712
{karthiksh,dimitris}@utexas.edu,dimakis@austin.utexas.edu
†Department of Electrical Engineering
University of Southern California, Los Angeles, CA-90089
caire@usc.edu

***Abstract*—Several works have developed maximum-distance separable (MDS) storage codes that minimize the total *communication cost* required to repair a single coded symbol after an erasure, referred to as repair bandwidth (BW). A fundamental property of these codes is the fact that they are *vector-linear*: each coded symbol is treated as a collection of smaller sized sub-symbols. Communicating sub-symbols, instead of entire coded symbols, is the key to achieving low repair BW. In sharp contrast, off-the-shelf codes currently used in storage systems suffer from worst-case repair BW costs. Repairing classic codes efficiently has been an interesting open problem. The property of classic MDS codes that prevents non-trivial repair is that they are *scalar-linear*, i.e., symbols are treated as indivisible chunks.**

In this work, we present a simple framework that treats scalar codes, defined over extension fields, as vector-linear, enabling them to admit nontrivial repair schemes. The repair problem is viewed as a problem of designing algebraic repair field elements. Our framework can be seen as a finite field analogue of real interference alignment.

For the case of 2-parity MDS codes, we develop a scheme that we call *clique-repair* which identifies good repair strategies. It does so by using algebraic dependence properties hidden in the structure of the code. We use the above framework to repair specific $(5,3)$- and $(6,4)$-Reed-Solomon (RS) codes. Furthermore, we use the repair framework developed on the exact RS scalar code currently deployed in the Facebook Analytics Hadoop clusters. For this code,we present a repair strategy that saves approximately 20% of repair bandwidth.

***Index Terms*—Scalar MDS Codes; Reed Solomon; clique-repair; alignment.**

I. Introduction

Large scale distributed storage systems employ erasure coding to offer data reliability against physical node failures. Typically, erasure codes employed are (n,k) MDS (maximum distance separable) codes. An important property that ensures data reliability against failures is that encoded data from any k nodes suffice to recover the data stored. However, a central issue that arises in coded storage is the *Repair Problem*: how to maintain the encoded representation when a *single* node erasure occurs. To maintain the same redundancy posterior to an erasure, a new node has to join the storage array and regenerate the lost contents by downloading data from the remaining storage nodes.

During this repair process, there are several metrics that can be optimized. Currently, the most well understood one is the total number of bits communicated in the network. This metric, called repair bandwidth (BW), was characterized in [2] as a function of the storage per node. The codes with minimum storage that offer the optimal bandwidth are MDS, and are called minimum storage regenerating (MSR) codes. Building on the work in [2], a great volume of studies have developed MSR codes [3]–[12]. With these codes, in order to repair a failed node, a new node can access the remaining $n-1$ nodes participating in the coded storage. The key property of these codes is that they are vector codes, i.e., they are designed so that each coded symbol is considered as a collection of smaller sub-symbols. During repair, the BW savings are due to the fact that a small collection of sub-symbols can be communicated instead of entire coded symbols from each surviving node. Efficient codes, in terms of encoding and decoding schemes, have been found to meet the minimum BW bounds derived in [2] for rate $k/n \leq 1/2$. In the high rate regime, [5]–[9], [11] have presented constructions which have optimal repair efficiency. However, the amount of sub-packetization (sub-symbols) required is exponential in the parameters n and k. Code constructions in [10] rectify this problem by constructing high rate codes that have polynomial sub-packetization. However, these constructions are very specific to some rates. For more details on regenerating codes, we refer the reader to the surveys [13] [14] [3].

Another, more recently studied repair metric of interest is *locality*, i.e. the number of nodes accessed during repair. A code is said to have high locality if the number of nodes accessed during repair is large. Since, MDS codes possess high locality, near MDS codes were studied for optimal

This paper was presented in part at *50th Annual Allerton Conference on Communication, Control and Computing* 2012 [1]. This research was partially supported by NSF Awards 1055099, 1218235 and research gifts by Google, Intel and Microsoft.

locality in [15] [16] [17] [18] [19] [20].

An interesting problem related to constructing MSR codes that minimize the repair BW, is developing repair strategies for *existing* scalar MDS codes that are currently used in erasure coded storage systems. A major limiting issue of these codes is that they lack the fundamental ingredient of repair optimal ones: the vector-code property which enables efficient repair schemes. When an erasure occurs in scalar MDS encoded systems, the *naive* repair process generates BW equal in size to k information symbols in an (n,k) scalar MDS code. In this work, we focus on vectorizing scalar MDS codes, defined over a large extension field, over a suitable base field. Using the vectorization machinery, we formulate the repair problem as the problem of designing repair field elements.

Our contributions: In this work, we develop a framework to represent scalar-MDS codes in a vector form, when they are constructed over extension fields. The vector form provides more flexibility in designing non-trivial repair strategies. We pose the problem of repairing a systematic node as a problem of designing repair field elements satisfying some algebraic linear dependence properties. Using this framework, we develop a repair scheme that we call clique-repair for 2-parity scalar MDS codes. It gives a semi-analytical characterization of the relation between the structure of the code's generator matrix and the minimum repair bandwidth achievable. We show that, for the $(6,4)$ Reed Solomon code, using the clique repair scheme when the level of sub-packetization is the lowest, it is possible to obtain gains in terms of repair bandwidth. For $(5,3)$ and $(6,4)$ RS codes, these gains can be brought closer to the optimal cut-set bound of [2], by increasing the level of sub-packetization. Further, we present numerical results regarding repair of the $(14,10)$ RS code currently used in production [20] by Facebook Hadoop Analytics clusters. There we observe a 20% savings in terms of repair BW compared to naive repair.

II. Repair of MDS Storage Codes

In this section, we first state the repair BW minimization problem for vector codes to clarify the implications of storing vectors per coded symbol instead of scalars. Then, we see that scalar MDS codes have an inherent deficit by assuming indivisible coded symbols.

A. Vector MDS Codes

Let a file $\mathbf{x}$, without loss of generality be sub-packetized into $M = k(n-k)\beta$ p-ary information symbols such that $\mathbf{x} \in \mathbb{F}^{M\times 1}$ and partitioned in k parts $\mathbf{x} = \left[\mathbf{x}_1^T \dots \mathbf{x}_k^T\right]^T$, with $\mathbf{x}_i \in \mathbb{F}^{\frac{M}{k}\times 1}$, where M denotes the file size and $\beta \in \mathbb{Z}^+$ denotes the degree of sub-packetization and $\mathbb{F} \equiv \mathbb{GF}(p)$.

We want to store this file with rate $\frac{k}{n} \leq 1$ across k systematic and $n-k$ parity storage units with storage capacity $\frac{M}{k}$ p-ary symbols each. The encoding is given by $\mathbf{y} = [\mathbf{I}_M\ \mathbf{P}]^T\mathbf{x}$, where $\mathbf{I}_M$ denotes the $M \times M$ identity matrix and

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_1^{(k+1)} & \dots & \mathbf{P}_1^{(n)} \\ \vdots & \vdots & \vdots \\ \mathbf{P}_k^{(k+1)} & \dots & \mathbf{P}_k^{(n)} \end{bmatrix} \in \mathbb{F}^{M\times \frac{n-k}{k}M} \tag{1}$$

where $\mathbf{P}_i^{(j)} \in \mathbb{F}^{\frac{M}{k}\times\frac{M}{k}}$ represents a matrix of coding coefficients used by the jth node ($j \geq k+1$ and hence a parity node) to "mix" the symbols of the ith file piece $\mathbf{x}_i$. The MDS property is guaranteed if the file can be reconstructed from from any subset of size k of the n nodes storing the codeword $\mathbf{y}$.

Let $[k]$ denote the set $\{1,2,3\dots k\}$. To maintain the same redundancy when a single systematic node $i \in [k]$ fails, a repair process takes place to regenerate the lost data in a *newcomer* storage node. This process is carried out as linear operations on the content of the $n-1$ remaining nodes, namely, each parity node $j \in \{k+1\dots n\}$ sends data of size $\beta = \frac{M}{k(n-k)}$ (i.e., β equations) to the newcomer in the form

$$\begin{aligned}\mathbf{d}_i^{(j)} &= \left(\mathbf{R}_i^j\right)^T\left(\left(\mathbf{P}_1^{(j)}\right)^T\mathbf{x}_1 + \dots + \left(\mathbf{P}_k^{(j)}\right)^T\mathbf{x}_k\right)\\ &= \left[\mathbf{P}_1^{(j)}\mathbf{R}_i^j \dots \mathbf{P}_k^{(j)}\mathbf{R}_i^j\right]^T\mathbf{x},\end{aligned} \tag{2}$$

where $\mathbf{R}_i^j \in \mathbb{F}^{(n-k)\beta\times\beta}$ is a *repair matrix*, which can be suitably designed. In the same manner, all parity nodes proceed in transmitting a total of $\frac{M}{k}$ linear equations (i.e., the size of what was lost) to the newcomer, which eventually receives the following system of linear equations

$$\mathbf{d}_i = \underbrace{\begin{bmatrix}\left(\mathbf{P}_i^{(k+1)}\mathbf{R}_i^{k+1}\right)^T\\ \vdots \\ \left(\mathbf{P}_i^{(n)}\mathbf{R}_i^n\right)^T\end{bmatrix}\mathbf{x}_i}_{\text{useful data}} + \underbrace{\sum_{u=1,u\neq i}^{k}\begin{bmatrix}\left(\mathbf{P}_u^{(k+1)}\mathbf{R}_i^{k+1}\right)^T\\ \vdots \\ \left(\mathbf{P}_u^{(n)}\mathbf{R}_i^n\right)^T\end{bmatrix}\mathbf{x}_u}_{\text{interference by } \mathbf{x}_u}, \tag{3}$$

where $\mathbf{d}_i \in \mathbb{F}^{\frac{M}{k}}$. Solving for $\mathbf{x}_i$ is not possible due to the $(k-1)$ additive *interference* components in the received equations. To retrieve the lost piece of data, we need to "erase" the interference terms by downloading additional equations from the remaining $k-1$ systematic nodes and the resulting system has to be full-rank. To erase the interference generated by the undesired symbols ($\mathbf{x}_u$), we need to download from systematic node u the minimum number of equations that can re-generate the interference due to $\mathbf{x}_u$, i.e., we need to download data of size equal to rank $\left(\left[\ \mathbf{P}_u^{(k+1)}\mathbf{R}_i^{k+1} \dots\ \mathbf{P}_u^{(n)}\mathbf{R}_i^n\ \right]\right)$.

The cut-set bound of [2] states that β equations from each of the remaining nodes is the minimum one could achieve, i.e., the minimum rank of each interference space is β. This results in a minimum download bound of $\frac{n-1}{n-k}\frac{M}{k} = (n-1)\beta$. Observe that the above benefits can only be unlocked if we treat each stored symbol as a block of smaller $(n-k)\beta$ sub-symbols.

B. Scalar MDS Codes

When we consider scalar (n,k)-MDS codes, we assume that k information symbols $[x_1 \dots x_k] \in \mathbb{F}^{k\times 1}$ are used to generate n coded symbols $[y_1 \dots y_n] \in \mathbb{F}^{n\times 1}$ under the linear generator map

$$\mathbf{G} = [\mathbf{I}_k \;\; \mathbf{P}] \in \mathbb{F}^{k\times n}, \tag{4}$$

where $\mathbf{P}$ is the $k \times (n-k)$ matrix that generates the parity symbols of the code and $\mathbb{F} \equiv \mathbb{GF}(p^m)$ as before. Similar to the previous section, let $P_i^{(j)}$ denote the parity coefficient, drawn from $\mathbb{GF}(p^m)$, used by the jth parity node to multiply symbol x_i. Instead of matrices and vectors in the the previous case, here we have scalars drawn from the extension field $\mathbb{GF}(p^m)$. The MDS property is equivalent to the requirement that the k information symbols can be reconstructed from any subset of size k of the n coded symbols.

When a node, or a coded symbol is lost, if we wish to repair it using linear methods, we must download at least k coded symbols from a subset of size k of the $n-1$ surviving storage nodes, i.e., the size of the download is equal to the size of the whole file $\mathbf{x}$. Scalar-linear operations on this code binds us to this worst case repair bandwidth cost.

Moving away from linear methods, we could instead download "parts" of each symbol defined over $\mathbb{GF}(p^m)$. Observe that, over $\mathbb{GF}(p^m)$, each symbol consists of m sub-symbols defined over $\mathbb{GF}(p)$ and $\mathbb{GF}(p^m)$ is isomorphic to a vector space of dimension m over $\mathbb{GF}(p)$. This view can unlock a vector-like property for scalar codes.

In the following section, we describe how an extension field can be used to allow decomposition of each coded symbol into sub-symbols, such that scalar-MDS codes are interpreted as a vector-MDS code. The key ideas used are the following:

1) Each element of the generator matrix is viewed as a square matrix with dimensions $m \times m$ over the field $\mathbb{GF}(p)$.
2) Every data symbol x_i and every coded symbol y_i over $\mathbb{GF}(p^m)$ are viewed as vectors $\mathbf{x}_i$ and $\mathbf{y}_i$, respectively, of dimension m over the field $\mathbb{GF}(p)$.

III. Vectorizing scalar codes

We review some results [21] [22] regarding representations of finite field elements. Let the irreducible primitive polynomial $P(x)$ of degree m over the base field $\mathbb{GF}(p)$ that generates $\mathbb{GF}\left(p^m\right)$ be:

$$P(x) = a_0 + a_1 x + \dots a_{m-1}x^{m-1} + x^m, \tag{5}$$

where $a_0, \dots, a_{m-1} \in \mathbb{GF}(p)$. Let ζ be a root of the polynomial $P(x)$. Hence, ζ is a primitive element. Then, any field element $b \in \mathbb{GF}(p^m)$ can be written as a polynomial of ζ over $\mathbb{GF}(p)$ of degree at most $m-1$

$$b = b_0 + b_1\zeta + \dots b_{m-1}\zeta^{m-1} \tag{6}$$

where $b_i \in \mathbb{GF}(p)$, $i \in \{0, 1 \dots m-1\}$.

Definition 1: The *companion matrix* of the primitive polynomial $P(x) = a_0 + a_1 x + \dots a_{m-1}x^{m-1} + x^m$ is a $m \times m$ matrix given by:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & -a_2 \\ 0 & 0 & 1 & \cdots & 0 & -a_3 \\ . & . & . & \cdots & . & . \\ . & . & . & \cdots & . & . \\ . & . & . & \cdots & . & . \\ 0 & 0 & 0 & \cdots & 1 & -a_{m-1} \end{bmatrix}$$

1) *Vector Representation*: Any, $b \in \mathbb{GF}(p^m)$ can be interpreted as a vector that belongs to a vector space of dimension m over $\mathbb{GF}(p)$ with the following vector representation

$$f(b) = [b_0 \; b_1 \; \dots \; b_{m-1}]^T. \tag{7}$$

2) *Matrix Representation*: Any nonzero field element in $\mathbb{GF}(p^m)$ can be written as ζ^n, $0 \le n \le p^m - 2$. The mapping $g(\zeta^\ell) = \mathbf{C}^\ell$ is an isomorphism between $\mathbb{GF}\left(p^m\right)$ and the set of $m \times m$ matrices $\{0, \mathbf{C}^0, \mathbf{C}^1, \dots, \mathbf{C}^{p^m-2}\}$ over $\mathbb{GF}(p)$ that preserves the field multiplication and addition in terms of matrix multiplication and addition over the space of matrices $(\mathbb{GF}(p))^{m\times m}$.

We refer to $g(b) = \mathbf{B}$ as the "multiplication operator" corresponding to $b \in \mathbb{GF}(p^m)$ and to $f(b) = \mathbf{b}$ as the vector representation of $b \in \mathbb{GF}(p^m)$. Let $\mathcal{M}(\mathbb{F}) = \{\mathbf{0}, \mathbf{C}^0, \mathbf{C}^1 \dots \mathbf{C}^{p^m-2}\}$ be the set of multiplication operators. Then clearly,

- *P1 Additivity:* For any $c, d \in \mathbb{GF}(p)$ and $\mathbf{A}, \mathbf{B} \in \mathcal{M}(\mathbb{F})$, we have $c\mathbf{A} + d\mathbf{B} \in \mathcal{M}(\mathbb{F})$.
- *P2 Commutativity:* For any $\mathbf{A}, \mathbf{B} \in \mathcal{M}(\mathbb{F})$, we have $\mathbf{AB} = \mathbf{BA} \in \mathcal{M}(\mathbb{F})$.

Lemma 1: If $c = ab$ where $c, a, b \in \mathbb{GF}(p^m)$, then $\mathbf{c} = \mathbf{Ab}$ where $f(c) = \mathbf{c}$, $f(b) = \mathbf{b}$ and $g(a) = \mathbf{A}$.

Therefore, under the vector representation $f(\cdot)$, the information symbols x_i and the coded symbols y_i can be rewritten as m-dimensional vectors, $\mathbf{x}_i$ and $\mathbf{y}_i$ over $(\mathbb{GF}(p))^{m\times 1}$. Further, every entry of the generator matrix , i.e. $P_i^{(j)}$, can be represented in terms of the multiplication operator $\mathbf{P}_i^j$ under the matrix representation $g(\cdot)$ to give a vectorized code.

Setting $m = (n-k)\beta$, we observe that we have changed the scalar code in (4) into the vector code given by (1). The only difference is that the matrices $\mathbf{P}_i^{(j)}$ are multiplication operators that have the structure explained in this section. Note that the same construction has been recently used in [23]. Also, this construction can be taken to be the finite field analogue of the procedure for generating irrational dimensions out of a real dimension [24] that plays an important role in *real interference alignment*. Here, finite

number of dimensions are being generated out of a single element belonging to a large extension field. At this point, one could consider this to be a generic vector linear code and design repair methods. It is desirable to design repair vectors without consideration for the detailed structure of the multiplication operators. To this end, we will exploit properties of multiplication operators using the following lemma, concerning left multiplication of multiplication operators, to move towards a framework of designing repair field elements instead of repair vectors.

Lemma 2: For any two nonzero vectors $\mathbf{a}^T, \mathbf{b}^T \in \mathbb{GF}(p)^{1\times m}$, there always exists a multiplication operator (or a matrix) $\mathbf{M} \in \mathcal{M}(\mathbb{F})$ $(m \times m)$ such that $\mathbf{b}^T\mathbf{M} = \mathbf{a}^T$.

Proof: The proof is provided in the appendix. ■

Note that Lemma 2 does not follow directly from Lemma 1 as the property above concerns left multiplication while Lemma 1 is about right multiplication of multiplication operator matrices. Since the code is now assumed to be vectorized, we continue by considering the repair problem for single systematic node failures as in Section II-A and use Lemma 2. We download β equations from every node since we have vectorized over the field $\mathbb{GF}(p)$ and $m = (n-k)\beta$. Without loss of generality, let us consider the repair of node $i = 1$. As in section II-A, repair matrices which multiply the $n-k$ parities are denoted $\mathbf{R}^{k+1}, \dots \mathbf{R}^n \in (\mathbb{GF}(p))^{\beta(n-k)\times\beta}$, dropping the sub script i since we will state everything for repair of a specific node. Let $\mathbf{r}_j^\ell$ denote the j-th column of $\mathbf{R}^\ell$. This corresponds to the jth equation downloaded from node ℓ. Then, due to Lemma 2, we can take $\left(\mathbf{r}_1^{k+1}\right)^T$ as a reference vector and then obtain $\forall \ell,\ \forall j$,

$$\left(\mathbf{r}_j^\ell\right)^T = \left(\mathbf{r}_1^{k+1}\right)^T \mathbf{M}_j^\ell \tag{8}$$

where $\mathbf{M}_j^\ell \in \mathcal{M}(\mathbb{F})$. Because of the above ability to choose a reference vector, we have the following important theorem that gives an alternate algebraic characterization of any repair scheme.

Theorem 1: Consider a repair scheme where, after the design, the rank of the column of vectors corresponding to the ith data vector in (3) is

$$\text{rank}\left[\left(\mathbf{P}_i^{(k+1)}\mathbf{R}^1\right)^T \quad \left(\mathbf{P}_i^{(k+2)}\mathbf{R}^2\right)^T \quad \dots \right. \tag{9}$$

$$\left.\left(\mathbf{P}_i^{(n)}\mathbf{R}^n\right)^T\right]^T = r_i. \tag{10}$$

Then, one can find repair field elements $M_j^\ell \in \mathbb{F}, \ell \in [k+1, n], j \in [\beta]$, such that, the following holds $\forall i \in [k]$

$$r_i = \text{rank}_p\left(\left[M_1^{k+1}P_i^{k+1} \dots M_\beta^{k+1}P_i^{k+1} | \dots | M_1^n P_i^{(n)} \right.\right.$$
$$\left.\left. \dots M_\beta^n P_i^{(n)}\right]\right) \tag{11}$$

where $\text{rank}_p([a\ b])$ with $a, b \in \mathbb{GF}(p^m)$ is defined according to the following restricted definition of linear independence: A set of field elements $A_i \in \mathbb{GF}(p^m)$ are linearly independent over a sub-field $\mathbb{GF}(p)$ (i.e., have rank_p equal to the cardinality of this set of elements), when one cannot find non zero scalars $\alpha_i \in \mathbb{GF}(p)$ such that $\sum \alpha_i A_i = 0$. The converse of the above theorem also holds.

Proof: The proof uses Lemma 3, and is relegated to the appendix. ■

We call the above formulation of the repair problem as *repair field element design* as one needs to design these repair field elements (i.e M_j^ℓ's) that satisfy such linear independence properties. The repair bandwidth (in bits) of any such scheme with repair field elements is proportional to the sum of the ranks, i.e. $\sum_{i=1}^k r_i \log_2(p)$ bits where the code is sub packetized over $\mathbb{GF}(p)$.

The following intuitive result shows that any repair scheme which can be implemented with sub packetization over a higher field $\mathbb{GF}(p^r)$ can be implemented using an equivalent repair scheme with sub packetization over a lower field $\mathbb{GF}(p)$ and with the same benefits with respect to the repair bandwidth.

Lemma 3: Consider a scalar systematic (n, k) MDS code over a field $\mathbb{GF}(p^{a(n-k)})$ with an equivalent vector representation over $\mathbb{GF}(p^a)$. The repair involves downloading one equation from each parity node for repair. Let the associated repair field elements be $M_{k+1}, M_{k+2} \dots M_n$ that operate on the parity nodes numbering from $k+1$ to n. Consider the rank of the column of vectors in (11) corresponding to node i with respect to scalar combinations from $\mathbb{GF}(p^a)$. Let $r_i = \text{rank}_{p^a}\left(\left[M^{k+1}P_i^{k+1}\ M^{k+2}P_i^{k+2} \dots M^n P_i^n\right]\right)$. Define the new repair field elements, over the code sub packetized over $\mathbb{GF}(p)$ to be: $\left[\tilde{M}_1^i\ \tilde{M}_2^i \dots \tilde{M}_a^i\right] = \left[M^i,\ M^i\beta, \dots M^i\beta^{a-1}\right]$ corresponding to a equations being drawn from node i where $k+1 \le i \le n$. Here, $\beta \in \mathbb{GF}(p^a)$ is the solution of an irreducible polynomial over $\mathbb{GF}(p)$, of degree a, that defines the extension field $\mathbb{GF}(p^a)$. The system of new repair elements $\{\tilde{M}_j^i\}$ have the same repair bandwidth.

Proof: The proof is relegated to the appendix. ■

Therefore, once a repair scheme is determined at a higher level of sub packetization, then an equivalent repair scheme can be constructed at the lower level of sub packetization with the same repair bandwidth.

IV. Clique Repair

In this section, we use the above developed algebraic repair field formulation to prove the following theorem that gives an optimal achievable scheme if the sub-packetization level is $n-k$ or $\beta = 1$. Note that we made no assumption about $\mathbb{GF}(p)$. So this could be an extension field by itself. So, for a given extension field $\mathbb{GF}(p^{(n-k)\beta})$, where p is prime, one could do the vectorization over $\mathbb{GF}(p^\beta)$ so that the effective sub-packetisation level is $n-k$. The lowest

level is $n-k$ and the highest possible is $(n-k)\beta$ and any intermediate level would be $(n-k)r$ where r divides β. We now use this machinery to characterize the optimal repair scheme at level $n-2$ for any $(n, n-2)$ code. We call this scheme *Clique Repair*. From now on, without loss of generality, we assume that $P_j^{(k+1)} = 1$, i.e. all parity coefficients for parity node $k+1$ are 1. This can be justified as it does not affect the MDS repair problem.

Theorem 2: Consider a systematic $(n, n-2)$-MDS code over $\mathbb{GF}\left(p^{2r}\right)$ and an undirected graph $G(V, E)$ such that $|V| = k$ and $(i,j) \in E$ iff $P_i^{(k+2)}\left(P_j^{(k+2)}\right)^{-1} \in \mathbb{GF}\left(p^r\right)$. Then, with linear repair schemes, node i cannot be repaired with BW less than $M - \frac{C_i}{2}\frac{M}{k}$, where C_i is the size of the largest clique of G not containing node i.

Proof: If $P_x^{(k+2)}\left(P_y^{(k+2)}\right)^{-1} \in \mathbb{GF}\left(p^r\right)$ and $P_y^{(k+2)}\left(P_z^{(k+2)}\right)^{-1} \in \mathbb{GF}\left(p^r\right)$, then $P_x^{k+2}\left(P_y^{(k+2)}\right)^{-1} P_y^{k+2}\left(P_z^{k+2}\right)^{-1} = P_x^{k+2}\left(P_z^{k+2}\right)^{-1} \in \mathbb{GF}(p^r)$. The transitivity property partitions the graph G into disjoint cliques.

Using Theorem 1 we have that there are two repair field elements, i.e. 1, $\mu \in \mathbb{GF}(p^{2r})$ corresponding to the two repair vectors that will be used to multiply the contents of the two parities respectively. The repair field elements for parity 1 is 1 because the corresponding repair vector acts as the reference vector. Then, the rank of i-th block is 2 if 1 and $\mu P_i^{(k+2)}$ are linearly independent over subfield $\mathbb{GF}(p^r)$. Similarly, the rank would be 1 if they are linearly dependent, i.e $\mu P_i^{(k+2)} \in \mathbb{GF}(p^r)$. Now we establish the following property: if i and j are in the same clique, then either both columns of elements are simultaneously linearly dependent or linearly independent over $\mathbb{GF}(p^r)$. This is due to the fact that $\mu P_i^{k+2} \in \mathbb{GF}\left(p^r\right)$ forces $\mu P_i^{k+2}\left(P_i^{k+2}\right)^{-1} P_j^{k+2} \in \mathbb{GF}\left(p^r\right)$.

Similarly, if i and j are in different cliques, then the corresponding columns of vectors cannot be linearly dependent simultaneously. Suppose they are, then $\mu\left(P_i^{(k+2)}\right)^{-1} \in \mathbb{GF}(p^r)$ and $\mu\left(P_j^{(k+2)}\right)^{-1} \in \mathbb{GF}(p^r)$. Therefore, $\left(P_j^{(k+2)}\right)^{-1}\mu^{-1}\mu P_i^{(k+2)} = \left(P_j^{(k+2)}\right)^{-1} P_i^{(k+2)} \in \mathbb{GF}(p^r)$. But $(i,j) \notin E$ and therefore a contradiction.

For repair of node i, μ is chosen in such a way that $\mu P_i^{(k+2)} \notin \mathbb{GF}(p^r)$, so that the corresponding column of vectors are linearly independent. This selection of μ forces all blocks corresponding to the nodes in the same clique to be linearly independent and it can at most make columns corresponding to exactly one other clique linearly dependent. Hence, the reduction in number of equations to be downloaded comes from the dependent clique. From this, the last claim in the theorem follows. ∎

Although the theorem above only specifies a lower bound, one can come up with an algorithm to achieve the optimum performance. It is easy to check that the following algorithm works. The algorithm Generate Clique gives disjoint cliques.

Algorithm 1 *Generate Clique*

```
while i = 1 → k do
  while j = 1 → i − 1 do
    if P_i^(k+2) (P_j^(k+2))^{-1} ∈ GF(p^r) then
      E ← (i, j)
    end if
  end while
end while
```

Let us assume that there is a list $\{C[i]\}_{1\le i\le m}$ such that $C[i]$ contains all vertices contained in clique i. The algorithm Find Repair finds the optimal repair field element μ. Notice

Algorithm 2 *Find Repair*

```
Find N : i ∈ C[N]
k_max ← arg max_{k≠N} ||C[k]||
Pick some node ℓ ∈ C_{k_max}.
μ ← (P_ℓ^(k+2))^{-1}
```

that the algorithm runs using $O(n^2)$ field multiplication operations. We observe that designing repair field elements for the highest level of sub-packetization, i.e $(n-k)\beta$, even when $n-k=2$, involves a brute force search using approximately $O(q^{\log q})$ operations. Since $q \ge n$ for most MDS codes, one needs at least $O(n^{\log n})$ operations. Moreover, apart from the complexity aspects, the scheme gives a semi-analytical connection between the repair BW and the coefficients of the generator matrix.

We present examples of bandwidth savings that are possible for $(5,3)$ and $(6,4)$ Reed Solomon Codes and for the $(14,10)$ Reed Solomon Code employed in HDFS open source module.

V. Analysis of the (5,3) and (6,4) Reed Solomon Codes

A. Repairing the $(5,3)$-RS Code

Consider a $(5,3)$-Reed Solomon code over $\mathbb{F} = \mathbb{GF}(2^4)$. Let ω be the fifth root of unity. Using the explicit formula for the generator matrix of the systematic Reed Solomon code given in [26], we obtain the following structure for the generator matrix $\mathbf{G}$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & \frac{(\omega^4-\omega^2)(\omega^4-\omega^3)}{(\omega-\omega^2)(\omega-\omega^3)} & \frac{(\omega^5-\omega^2)(\omega^5-\omega^3)}{(\omega-\omega^2)(\omega-\omega^3)} \\ 0 & 1 & 0 & \frac{(\omega^4-\omega)(\omega^4-\omega^3)}{(\omega^2-\omega)(\omega^2-\omega^3)} & \frac{(\omega^5-\omega)(\omega^5-\omega^3)}{(\omega^2-\omega)(\omega^2-\omega^3)} \\ 0 & 0 & 1 & \frac{(\omega^4-\omega)(\omega^4-\omega^2)}{(\omega^3-\omega)(\omega^3-\omega^2)} & \frac{(\omega^5-\omega)(\omega^5-\omega^2)}{(\omega^3-\omega)(\omega^3-\omega^2)} \end{bmatrix}.$$

Repair for node 1	$g_1(\omega) = \omega^3 + 1$ $g_2(\omega) = \omega^2 + 1$ $f_1(\omega) = \omega g_1(\omega)$ $f_2(\omega) = g_2(\omega)(\omega^2 + 1)$
Repair for node 2	$g_1(\omega) = \omega + 1$ $g_2(\omega) = \omega$ $f_1(\omega) = (\omega^2 + \omega + 1)g_1(\omega)$ $f_2(\omega) = g_2(\omega)(\omega^2 + 1)$
Repair for node 3	$g_1(\omega) = \omega + 1$ $g_2(\omega) = \omega$ $f_1(\omega) = (\omega^2 + \omega + 1)g_1(\omega)$ $f_2(\omega) = g_2(\omega)\omega$

TABLE I
REPAIR FIELD ELEMENTS FOR REPAIR OF SYSTEMATIC NODES FOR THE (5,3) RS CODE.

For the repair problem, without loss of generality, the generator matrix given above can be simplified by factoring out some coefficients along every row and renormalizing so that we can work on the following equivalent generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & \omega^2 + \omega + 1 \\ 0 & 1 & 0 & 1 & \omega \\ 0 & 0 & 1 & 1 & \omega^2 + 1 \end{bmatrix} \tag{12}$$

Let α be the primitive element of $\mathbb{F}$ corresponding to the primitive polynomial $P(x) = 1 + x + x^4$. Then, $\alpha^{15} = 1$ and $\omega = \alpha^3$. Consider the vector representation of the code over $\mathbb{GF}(2^2)$. Then by applying Theorem 2, we find that all the three nodes lie in the same clique. In other words, $(\omega^2+\omega+1)^{-1}\omega$, $\omega(\omega^2+1)^{-1}$ belong to $\mathbb{GF}(2^2)$. Hence, we cannot force any interference to lose rank without sacrificing the linear independence property. Hence, for this code, clique repair does not give any gain in terms of repair bandwidth.

We can, alternatively, consider the repair problem under the vector representation of the code over $\mathbb{GF}(2)$. For this case, $\beta = 2$. Hence, $\beta = 2$ equations are downloaded from the 2 parity nodes. We observe that in the fifth row $[1+\alpha^6+\alpha^3, \alpha^3, \alpha^6+1]$ of the generator matrix, two coefficients add up to give the third coefficient. This, as we will see, restricts the design of repair schemes. The cut-set bound (from the optimal repair bound) for this scenario is downloading 8 equations in total (i.e., $\frac{n-1}{n-k}\frac{M}{k}$).

The polynomials in Table I correspond to repair by downloading 10 equations in the event of a failure of any systematic node for the $(5, 3)$ Reed Solomon code over $\mathbb{GF}(16)$. It is possible to prove that it is not possible to repair linearly with less than 10 equations for any systematic node for this code. However, we skip the argument. This indicates that allowing higher sub packetization β can give better results.

Repair for node 1	$g_1(\alpha) = \alpha^{-2}$ $g_2(\alpha) = 1$ $f_1(\alpha) = 1$ $f_2(\alpha) = \alpha^2$
Repair for node 4	$g_1(\alpha) = 1$ $g_2(\alpha) = \alpha$ $f_1(\alpha) = 1$ $f_2(\alpha) = \alpha$

TABLE II
REPAIR FIELD ELEMENTS FOR REPAIR OF NODES 1 AND 4 FOR THE SYSTEMATIC (6,4) RS CODE WITH SUB PACKETIZATION OVER $\mathbb{GF}(2)$.

B. Repairing the $(6, 4)$-RS Code

Here, we consider a $(6, 4)$-RS code over $\mathbb{GF}(2^4)$. Using the formula in [26], we obtain the following generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & \alpha^3 + \alpha^2 + \alpha + 1 \\ 0 & 1 & 0 & 0 & 1 & \alpha + 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & \alpha^2 \end{bmatrix} \tag{13}$$

We consider the vector representation of the code over $\mathbb{GF}(2^2)$. If we apply Theorem 2 to this code, there are 3 cliques that are formed. The first clique contains nodes 1 and 4. The second contains 2 and the third clique contains node 3. By the repair algorithm presented in section V, the repair of nodes 2 and 3 require only 6 repair equations to be downloaded. For the repair of nodes 1 and 4, 7 equations need to be downloaded which is close to the filesize that requires $M = 8$, while the cut-set bound is $\frac{n-1}{n-k}\frac{M}{k} = 5$ equations.

Now, consider a higher level of sub-packetization, i.e. each node stores 4 elements over $\mathbb{GF}(2)$. Now, $M = 16$ elements. For this case, the repair scheme with repair field elements given in Table II improves the repair BW for nodes 1 and 4 to 12 equations from $7 \times 2 = 14$ equations. We get a uniformly good repair scheme (by Lemma 3, a good repair scheme for nodes 2 and 3 over $\mathbb{GF}(2^2)$ can be converted to a repair scheme over $\mathbb{GF}(2)$ with the same repair bandwidth in bits) that requires 12 equations to repair all systematic node failures, when the cut-set bound is 10 equations. It is possible to show that 12 equations is the optimal linear repair bandwidth for this code. We skip the argument.

We would like to note that for both codes, when $\beta = 2$, clique repair technique is not applicable, and repair schemes were designed based on observation.

VI. NUMERICAL RESULTS ON THE $(14, 10)$ REED-SOLOMON CODE IMPLEMENTED IN THE HADOOP FILE SYSTEM

The Apache Hadoop Distributed File System (HDFS) relies by default on block replication for data reliability. A module called HDFS RAID ([20], [27]) was recently developed for HDFS that allows the deployment of Reed-Solomon and also more sophisticated distributed storage

codes. HDFS RAID is currently used in production clusters including Facebook analytics clusters storing more than 30 PB of data. In this section, we present numerical results on improving the repair performance of the specific $(14, 10)$ Reed-Solomon code implemented in HDFS-RAID [27].

HDFS RAID implements a systematic Reed Solomon code over the extension field $\mathbb{GF}(2^8)$. Let α be the root of the primitive polynomial $1 + x^2 + x^3 + x^4 + x^8$ that generates the extension field. The exact generator matrix used is:

$$\mathbf{G} = [\mathbf{I}_{10} \quad \mathbf{P}]$$

where $\mathbf{P}$ is a 10×4 matrix given by:

$$\mathbf{P} = \begin{bmatrix} \alpha^{6} & \alpha^{78} & \alpha^{249} & \alpha^{75} \\ \alpha^{81} & \alpha^{59} & \alpha^{189} & \alpha^{163} \\ \alpha^{169} & \alpha^{162} & \alpha^{198} & \alpha^{131} \\ \alpha^{137} & \alpha^{253} & \alpha^{49} & \alpha^{143} \\ \alpha^{149} & \alpha^{177} & \alpha^{96} & \alpha^{205} \\ \alpha^{211} & \alpha^{71} & \alpha^{157} & \alpha^{134} \\ \alpha^{140} & \alpha^{236} & \alpha^{154} & \alpha^{43} \\ \alpha^{49} & \alpha^{213} & \alpha^{112} & \alpha^{88} \\ \alpha^{94} & \alpha^{171} & \alpha^{138} & \alpha^{95} \\ \alpha^{101} & \alpha^{13} & \alpha^{148} & \alpha^{173} \end{bmatrix}$$

Since the number of parities is 4 ($n - k = 4$), the clique repair technique is not applicable. We consider repair with the highest possible sub packetization, i.e. $\beta = 2$ and each node stores $(n-k)\beta = 8$ elements over $GF(2)$ and $M = 80$. The repair requires downloading 2 equations form every parity node. We provide a repair scheme in terms of the eight repair field elements $M_1^{11}, M_2^{11}, \ldots M_1^{14}, M_2^{14}$, belonging to $\mathbb{GF}(2^8)$, as in Theorem 1. The repair scheme lists the repair field elements that are required for repair of each systematic node and lists the total number of equations to be downloaded for repair over all nodes in each case. The average number of equations to be downloaded is 64.2 equations. The naive repair involves downloading 80 equations and the lower bound $\frac{n-1}{n-k}\frac{M}{k}$ gives 26 equations. We note that the repair scheme that we provide is not the optimal for the code because an exhaustive search involves checking a huge number of combinations (about 2^{64} combinations) of the repair field elements. We have searched over about 100000 random combinations of the repair field elements to produce this repair scheme that saves about 20 percent bandwidth. The repair scheme is given in Table III.

VII. Conclusion

We introduced a framework for repairing scalar codes by treating them as vectors over a smaller field. This is achieved by treating multiplication of scalar field elements in the original field as a matrix-vector multiplication operation over the smaller field. Interference alignment conditions map to designing repair field elements in the large field. Further using the conditions on designing repair field elements, we introduced the *clique repair* scheme for two parities when the level of sub-packetization is $n - k$ which establishes a semi-analytical connection between the coefficients of the generator matrix and the repair schemes possible. We exhibited good repair schemes for a few Reed Solomon codes including the one currently deployed in Facebook. This work hints at the existence of scalar MDS codes with good repair properties. An interesting problem would be to come up with easily testable conditions, similar in spirit to the clique repair scheme, for codes with larger number of parities and for higher sub-packetization. Sufficient conditions for a specific class of codes like Reed Solomon would be really interesting. More generally, scalar MDS codes with optimal/near optimal repair could be possibly designed using this framework.

References

[1] K. Shanmugam, D. S. Papailiopoulos, A. G. Dimakis, and G. Caire, "A repair framework for scalar MDS codes," in *50th Annual Allerton Conference on Communication,Control and Computing (Allerton), 2012*. IEEE, 2012, pp. 1166–1173.

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

[3] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[4] "Distributed storage wiki." http://tinyurl.com/storagecoding.

[5] I. Tamo, Z. Wang, and J. Bruck, "MDS array codes with optimal rebuilding," in *IEEE International Symposium on Information Theory Proceedings (ISIT), 2011*. IEEE, 2011, pp. 1240–1244.

[6] V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li, "Optimal repair of MDS codes in distributed storage via subspace interference alignment," *arXiv preprint arXiv:1106.1250*, 2011.

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

[8] C. Suh and K. Ramchandran, "Exact-repair MDS code construction using interference alignment," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1425–1442, 2011.

[9] K. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *47th Annual Allerton Conference onCommunication, Control, and Computing, 2009. Allerton*. IEEE, 2009, pp. 1243–1249.

[10] V. R. Cadambe, C. Huang, J. Li, and S. Mehrotra, "Polynomial length MDS codes with optimal repair in distributed storage," in *Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), 2011*. IEEE, 2011, pp. 1850–1854.

[11] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, "Repair optimal erasure codes through hadamard designs," in *49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2011*. IEEE, 2011, pp. 1382–1389.

[12] K. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," *arXiv preprint arXiv:1302.5872*, 2013.

[13] F. Oggier and A. Datta, "Coding techniques for repairability in networked distributed storage systems," 2012.

[14] A. Datta and F. Oggier, "An overview of codes tailor-made for networked distributed data storage," *arXiv preprint arXiv:1109.2317*, 2011.

[15] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *IEEE International Symposium on Information Theory Proceedings (ISIT), 2012*. IEEE, 2012, pp. 2771–2775.

[16] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration," *arXiv preprint arXiv:1211.1932*, 2012.

TABLE III
REPAIR SCHEME FOR $(14, 10)$ REED SOLOMON CODE THAT SAVES 20 PERCENT IN REPAIR BANDWIDTH. THE NAIVE REPAIR INVOLVES DOWNLOADING 80 BITS FOR REPAIR.

Systematic node repaired	Repair field elements $\left[M_1^{11}\ M_2^{11} \ldots M_1^{14}\ M_2^{14}\right]$	Repair Bandwidth (bits downloaded)
1	$\left[\alpha^{69}\ \alpha^{203}\ \alpha^{189}\ \alpha^{64}\ \alpha^{170}\ \alpha^{173}\ \alpha^{64}\ \alpha^{174}\right]$	65
2	$\left[\alpha^{8}\ \alpha^{191}\ \alpha^{175}\ \alpha^{248}\ \alpha^{18}\ \alpha^{1}\ \alpha^{69}\ \alpha^{126}\right]$	64
3	$\left[\alpha^{153}\ \alpha^{15}\ \alpha^{101}\ \alpha^{3}\ \alpha^{223}\ \alpha^{179}\ \alpha^{114}\ \alpha^{14}\right]$	64
4	$\left[\alpha^{92}\ \alpha^{86}\ \alpha^{31}\ \alpha^{129}\ \alpha^{67}\ \alpha^{213}\ \alpha^{67}\ \alpha^{144}\right]$	64
5	$\left[\alpha^{46}\ \alpha^{213}\ \alpha^{86}\ \alpha^{151}\ \alpha^{28}\ \alpha^{169}\ \alpha^{69}\ \alpha^{146}\right]$	63
6	$\left[\alpha^{83}\ \alpha^{164}\ \alpha^{182}\ \alpha^{116}\ \alpha^{104}\ \alpha^{185}\ \alpha^{245}\ \alpha^{178}\right]$	64
7	$\left[\alpha^{57}\ \alpha^{48}\ \alpha^{14}\ \alpha^{111}\ \alpha^{195}\ \alpha^{60}\ \alpha^{221}\ \alpha^{132}\right]$	64
8	$\left[\alpha^{51}\ \alpha^{174}\ \alpha^{206}\ \alpha^{224}\ \alpha^{104}\ \alpha^{100}\ \alpha^{52}\ \alpha^{143}\right]$	65
9	$\left[\alpha^{84}\ \alpha^{250}\ \alpha^{143}\ \alpha^{76}\ \alpha^{21}\ \alpha^{225}\ \alpha^{207}\ \alpha^{105}\right]$	65
10	$\left[\alpha^{161}\ \alpha^{180}\ \alpha^{131}\ \alpha^{89}\ \alpha^{69}\ \alpha^{37}\ \alpha^{15}\ \alpha^{177}\right]$	64

[17] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," 2011.

[18] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *arXiv preprint arXiv:1210.6954*, 2012.

[19] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin *et al.*, "Erasure coding in windows azure storage," in *USENIX conference on Annual Technical Conference, USENIX ATC*, 2012.

[20] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," *arXiv preprint arXiv:1301.3791*, 2013.

[21] R. Lidl and H. Niederreiter, *Finite fields*. Cambridge University Press, 1996, vol. 20.

[22] F. MacWilliams and N. Sloan, *The Theory of error-correcting codes*. North-Holland, 2006.

[23] S.-N. Hong and G. Caire, "Structured lattice codes for 2 × 2 × 2 MIMO interference channel," *arXiv preprint arXiv:1301.6453*, 2013.

[24] A. S. Motahari, S. O. Gharan, and A. K. Khandani, "Real interference alignment with real numbers," *arXiv preprint arXiv:0908.1208*, 2009.

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

[26] D. J. Versfeld, J. N. Ridley, H. C. Ferreira, and A. S. Helberg, "On systematic generator matrices for Reed–Solomon codes," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2549–2550, 2010.

[27] HDFS-Wiki:http://wiki.apache.org/hadoop/HDFS-RAID.

APPENDIX

A. Proof of Lemma 2

Proof: We note that multiplication of the matrix $\mathbf{M}$ from the left by $\mathbf{a}^T$ does not represent field multiplication. Hence, with respect to left multiplication, the matrix M does not *necessarily* act as a multiplication operator. The theorem implies that given any two arbitrary non zero repair vectors, one can find a multiplication matrix that connects both.

Since, non zero field elements in $\mathbb{F}$ are finite, there are finitely many operators in $\mathcal{M}(\mathbb{F})$. Let them be denoted by $\mathbf{M}_1, \mathbf{M}_2, \ldots..\mathbf{M}_{p^m-1}$. All these matrices (or operators) have full rank. We consider the products, $\mathbf{a}^T\mathbf{M}_i$. We show that all of them are distinct. Suppose for some $i \neq j, \mathbf{a}^T\mathbf{M}_i = \mathbf{a}^T\mathbf{M}_j$, then

$$\mathbf{a}^T\left(\mathbf{M}_i - \mathbf{M}_j\right) = 0 \tag{14}$$

But $\mathbf{M}_i - \mathbf{M}_j$ is another multiplication operator by additivity property. It is non zero and has full rank since $\mathbf{M}_i \neq \mathbf{M}_j$. This means all the $p^m - 1$ products are different. Since there are only $p^m - 1$ non zero repair vectors $\mathbf{b}^T$, given any $\mathbf{b}^T$, one can always find a $\mathbf{M}_j$ such that $\mathbf{a}^T\mathbf{M}_j = \mathbf{b}^T$. ■

B. Proof of Theorem 1

Proof: Taking the first repair vector $\left(\mathbf{r}_1^{k+1}\right)^T$ as reference, every other repair vector can be written in the form given by (8) using Lemma 2. It involves repair matrices $\mathbf{M}_\mathbf{j}^\ell$, representing download of repair equation j from node ℓ, corresponding to repair field elements M_j^ℓ. All multiplication matrices are full rank matrices. Combined with property P1, it gives the rank condition over sub-field $\mathbb{GF}(p)$ as stated in (11). Also, given a set of repair field elements it is possible to construct a set of repair multiplication operators and together with an arbitrary choice of non-zero reference repair vector, one can construct repair vectors satisfying the same rank conditions. ■

C. Proof of Lemma 3

Proof: It is enough to show that b field elements of the form $\{M_jP_j\},\ 1 \leq j \leq b$ are linearly dependent over $\mathbb{GF}(p^a)$ if and only if ab field elements $\{M_jP_j\beta^s\},\ 1 \leq j \leq b,\ 0 \leq s \leq a-1$ are also linearly dependent over $\mathbb{GF}(p)$. Here, M_j correspond to the repair field elements and P_j correspond to the coefficients of the generator matrix corresponding to the parity node. Linear dependence over $\mathbb{GF}(p^a)$ implies that there exists scalars $\gamma_j \in \mathbb{GF}(p^a)$, with at least one of them non-zero, such that $\sum_j \gamma_j M_j P_j = 0$. Let us rewrite field elements γ_j in $\mathbb{GF}(p^a)$ as polynomials in β with coefficients γ_{js} from $\mathbb{GF}(p)$. Hence, the linear dependency relation becomes $\sum_j \sum_s \gamma_{js} M_j P_j \beta^s = 0$. Hence over $\mathbb{GF}(p)$, $\{M_jP_j\beta^s\},\ 1 \leq b,\ 0 \leq s \leq a-1$ are linearly dependent. The converse is also true since some scalar set $\gamma_{js} \in \mathbb{GF}(p)$ with one non zero element determines a non-zero $\gamma_j \in \mathbb{GF}(p^a)$. Hence the claim in the theorem follows. ■