Repairing Reed-Solomon Codes With Multiple Erasures
Hoang Dau, Member, IEEE, Iwan Duursma, Han Mao Kiah, Member, IEEE, and Olgica Milenkovic, Fellow, IEEE

Abstract—Despite their exceptional error-correcting properties, Reed-Solomon codes have been overlooked in distributed storage applications due to the common belief that they have poor repair bandwidth: A naive repair approach would require the whole file to be reconstructed in order to recover a single erased codeword symbol. In a recent work, Guruswami and Wootters (STOC’16) proposed a single-erasure repair method for Reed-Solomon codes that achieves the optimal repair bandwidth amongst all linear encoding schemes. Their key idea is to recover the erased symbol by collecting a sufficiently large number of its traces, each of which can be constructed from a number of traces of other symbols. We extend the trace collection technique to cope with two and three erasures.

I. INTRODUCTION

A. Background

The repair bandwidth is an important performance metric of erasure codes in the context of distributed storage [2, 3]. In such a system, for a chosen field $F$, a data vector in $F^n$ is mapped to a codeword vector in $F^m$, whose entries are stored at different storage nodes. When a node fails, the symbol stored at that node is erased (lost). A replacement node (RN) has to recover the content stored at the failed node by downloading repair data from the other nodes. The repair bandwidth is the total amount of information that the RN has to download in order to successfully complete the repair process. Bandwidth-efficient repair schemes not only allow low-cost recovery of permanently failed nodes, but also improve the performance of degraded reads during transient node failures.

Reed-Solomon (RS) codes [4], which have been extensively studied in theory [5] and widely used in practice, were believed to have prohibitively high repair bandwidth. In a naive repair scheme, recovering the content stored at a single failed node would require downloading the whole file, i.e., $k$ symbols over $F$. The poor performance in repairing failed nodes of RS codes motivated the introduction of repair-efficient codes such as regenerating codes [2, 3, 6] and locally repairable codes [7, 8, 9].

Guruswami and Wootters [10] recently proposed a bandwidth-optimal linear repair method based on RS codes. The key idea behind their method is to recover a single erased symbol by collecting a sufficiently large number of its (field) traces, each of which can be constructed from a number of traces of other symbols. As all traces belong to a subfield $B$ of $F$ and traces from the same symbol are related, the total repair bandwidth can be significantly reduced. The repair scheme obtained by Guruswami and Wootters [10], however, only applies to the case of one erasure, or in other words, one failed node.

B. Our Contribution

We propose an extension of the Guruswami-Wootters repair scheme that can ensure recovery from multiple erasures, more precisely two or three erasures. In particular, we provide two distributed repair schemes for RS codes for the case of two erasures, both of which use the same repair bandwidth $(n - 1)$ sub-symbols per erased symbol, equal to that of a single erasure (although note that the lower bound for the repair bandwidth per erasure in the case of multiple erasures may not be the same as that for the case of a single erasure). In these schemes, the two RNs first download repair data from all available nodes (Download Phase). They subsequently collaborate to exchange the data in order to complete the repair process at each node (Collaboration Phase). The first repair scheme requires one round

Fig. 1: A toy example illustrating the repair procedure for two failed nodes in a four-node storage system based on a $[4, 2]$ Reed-Solomon code over $\mathbb{F}_4$. The stored file is $\{(a_1, a_2), (b_1, b_2)\} \in \mathbb{F}_4^2$, where $a_1, a_2, b_1,$ and $b_2$ are bits in $\mathbb{F}_2$. Suppose that Node 2 and Node 3 fail simultaneously. In the Download Phase, each replacement node downloads two bits (along the solid arrows) from the two available nodes, namely Node 1 and Node 4. In the Collaboration Phase, the replacement nodes communicate with each other to complete their own repair processes by exchanging two extra bits (along the dashed arrows), computed based on the previously downloaded bits.
of collaboration, that is, in the Collaboration Phase, the two RNs send out repair data to each other simultaneously in one round. This scheme works whenever the field extension degree $t$ is divisible by the characteristic of the field $F$. An example illustrating the first scheme is given in Fig. [1] The second scheme, however, requires two rounds of collaboration, that is, in the Collaboration Phase, one RN receives the repair data from the other RN, completes its repair process, and then sends out its repair data to the other node. This scheme applies to all field extension degrees. Both distributed repair schemes can be easily modified to obtain centralized repair schemes, where the two erased symbols are recovered simultaneously by a repair center.

We also develop a centralized repair scheme to cater to three erasures, and precisely characterize the patterns of three erasures that can be recovered at a low cost. This scheme can be slightly modified to produce a distributed repair scheme with a repair bandwidth of $n - 1$ sub-symbols from $B$ per erased symbol, which equals the bandwidth needed for a single erasure repair. This distributed scheme requires three rounds of collaboration. Both schemes work whenever the extension degree $t$ is divisible by the characteristic of $F$.

| Storage system                   | Erasure code | # tolerable failures |
|----------------------------------|--------------|----------------------|
| Google File System II (Colossus) | RS(9,6)      | Three                |
| Quintcast File System [12]       | RS(9,6)      | Three                |
| Hadoop Distributed File System   | RS(9,6)      | Three                |
| Yahoo Cloud Object Store [14]    | RS(11,8)     | Three                |
| Backblaze Vaults [15]            | RS(20,17)    | Three                |
| Facebook f4 Storage System [16]  | RS(14,10)    | Four                 |
| Baidu Atlas Cloud Storage [11]   | RS(12,8)     | Four                 |

TABLE I: Some well known storage systems and their underlying Reed-Solomon codes (more precisely, their underlying generalized Reed-Solomon codes (see Section II-A)). However, it is common practice to refer to them as Reed-Solomon codes.

It is worth pointing out that most of the current distributed storage systems offer protection at most three or four node failures, which mitigates the need for higher-order failure correction (see Table I). Multiple failures, although significantly less frequent than single failures, do occur in these storage systems. For instance, it was observed [13], [19] that in the data-warehouse cluster in production at Facebook, a median of 180 Terabytes of data is transferred through the top-of-racks switches every day to repair 95,500 blocks of RS-coded data, 98.08% of which are single failures, 1.87% are double failures, and 0.05% are triple or higher order failures. Multiple-failure repair schemes are also useful in storage systems that adopt a lazy repair model such as Total Recall [20] or liquid storage systems [21]. In such a system, one waits until the number of failures reaches a certain threshold before performing the repair procedure. If designed properly, this repair mode can help to reduce the network recovery traffic, while still guaranteeing an acceptable low risk of permanent system failure [22], [23].

C. Organization

The paper is organized as follows. We first provide relevant definitions and introduce the terminology used throughout the paper. We then proceed to discuss the naive repair scheme as well as the Guruswami-Wootters repair scheme for RS codes in the presence of a single erasure in Section III. Our main results – repair schemes for RS codes in the presence of two erasures and three erasures – are presented in Section III and Section IV. We conclude the paper in Section V by discussing some open problems and related work.

II. REPAIRING REED-SOLOMON CODES WITH ONE ERASURE

We start by introducing relevant definitions and the notation used in all subsequent derivations, and then proceed to review the naive repair approach as well as the approach proposed by Guruswami and Wootters [10] for repairing a single erasure/node failure in RS codes.

A. Definitions and Notations

Let $[n]$ denote the set $\{1,2,\ldots,n\}$. Let $B = GF(p^m)$ be the finite field of $p^m$ elements, for some prime $p$ and $m \geq 1$. Let $F = GF(p^m)$ be a field extension of $B$, where $t \geq 1$. The prime number $p$ is called the characteristic of $F$, denoted char($F$). We often refer to the elements of $F$ as symbols and the elements of $B$ as sub-symbols. We can also treat $F$ as a vector space of dimension $t$ over $B$, i.e. $F \cong B^t$, and hence each symbol in $F$ may be represented as a vector of length $t$ over $B$. The rank of a set $V \subseteq F$ over $B$ is denoted by rank$_B(V)$.

A linear $[n,k]$ code $C$ over $F$ is a subspace of $F^n$ of dimension $k$. Each element of a code is referred to as a codeword. The dual of a code $C$, denoted $C^\perp$, is the orthogonal complement of $C$. The support of a vector $c = (c_1,\ldots,c_n) \in F^n$ is defined as $\text{supp}(c) = \{j : c_j \neq 0\}$. The Hamming weight of a vector $c \in F^n$, denoted $\text{wt}(c)$, is the size of its support $\text{supp}(c)$. The Hamming distance of any two vectors $c$ and $c' \in F^n$, denoted $d(c,c')$, is defined by $\text{wt}(c-c')$. The minimum distance of a code $C$, denoted $d(C)$, is the minimum Hamming distance between any two distinct codewords of $C$. The Singleton bound establishes that for any $[n,k]$ code of minimum distance $d$, the inequality $d \leq n-k+1$ holds (see [3]). A code that has minimum distance attaining this bound is called a maximum-distance separable (MDS) code.

Definition 1. Let $F[x]$ denote the ring of polynomials over $F$. The Reed-Solomon code $RS(A,k) \subseteq F^n$ of dimension $k$ over a finite field $F$ with evaluation points $A = \{\alpha_1,\alpha_2,\ldots,\alpha_n\} \subseteq F$ is defined as:

$$RS(A,k) = \{ (f(\alpha_1),\ldots,f(\alpha_n)) : f \in F[x], \; \deg(f) < k \}.$$ 

A generalized Reed-Solomon code, $GRS(A,k,\lambda)$, where $\lambda = (\lambda_1,\ldots,\lambda_n) \in F^n$, is defined similarly to a Reed-Solomon code, except that the codeword corresponding to a polynomial $f$ is now defined as $(\lambda_1f(\alpha_1),\ldots,\lambda_nf(\alpha_n))$, $\lambda_i \neq 0$ for all $i \in [n]$. It is well known that the dual of an RS code $RS(A,k)$, for any $n \leq |F|$, is a generalized RS code $GRS(A,n-k,\lambda)$, for some multiplier vector $\lambda$ (see [3, Chp. 10]). Whenever clear from the context, we use $f(x)$ to denote a polynomial of degree at most $k-1$, which corresponds to a codeword of the RS code $C = RS(A,k)$, and $p(x)$ to denote a polynomial of degree at most $n-k-1$, which corresponds to a dual codeword in $C^\perp$. Since $\sum_{\alpha \in A} p(\alpha) (\lambda_n f(\alpha)) = 0$, we refer to such a polynomial $p(x)$ as a check polynomial for $C$. Note that when $n = |F|$, we have $\lambda_\alpha = 1$ for all $\alpha \in F$. In general, as recovering $f(\alpha)$ is equivalent to recovering $\lambda_\alpha f(\alpha)$, to simplify the notation, we often omit the factor $\lambda_\alpha$ in the equation above.
B. Naive Repair of One Erasure

Suppose that the polynomial \( f(x) \in F[x] \) corresponds to a
codeword in the RS code \( C = RS(A,k) \) and that \( f(\alpha^* ) \) is
the erased symbol, where \( \alpha^* \in A \) is an evaluation point of the
code. Pick a check \( p(x) \in F[x] \) of \( C \) such that \( p(\alpha^*) \neq 0 \) and
the Hamming weight of the dual codeword corresponding to \( p(x) \) is
precisely equal to \( k+1 \). The existence of such check is guaranteed
by the fact that in an MDS code of length \( n \) and minimum distance
\( n - (n - k) + 1 = k + 1 \), each subset of \( [n] \) of size \( k + 1 \) is always
the support of some codeword. The check \( p(x) \) generates the following
repair equation:

\[
p(\alpha^* )f(\alpha^*) = -\sum_{\alpha \in A \setminus \{\alpha^* \}} p(\alpha)f(\alpha). \tag{1}
\]

Since there are precisely \( k \) evaluation points \( \alpha \neq \alpha^* \) where
\( p(\alpha) \neq 0 \), (1) implies that \( f(\alpha^*) \) can be computed from \( f(\alpha) \) for
\( k \) different \( \alpha \in A \setminus \{\alpha^* \} \). Hence, the naive repair scheme requires
downloading \( k \) symbols over \( F \), or equivalently, \( kt \) sub-symbols over \( B \),
which is equal to the size of the stored file. Although this is a well-known
fact regarding RS codes, it provides a smooth transition from the naive repair to
the trace repair framework, replacing the one repair equation (1) with \( t \) equations (3).

C. The Guruswami-Wootters Repair Scheme for One Erasure

Given that \( F \) is a field extension of \( B \) of degree \( t \), i.e. \( F = GF(p^{mt}) \)
and \( B = GF(p^m) \), for some prime \( p \), one may define the
field trace of any symbol \( \alpha \in F \) as

\[
\text{Tr}_{F/B}(\alpha) = \sum_{i=0}^{t-1} \alpha^{|B|^i},
\]

which is always a sub-symbol in \( B \). We often omit the subscript \( F/B \) for
succinctness. The key points in the repair scheme proposed by Guruswami and
Wootters [10] can be summarized as follows:

- Each symbol in \( F \) can be recovered from its \( t \) independent
  traces, which we call the target traces. More precisely, given
  a basis \( u_1, u_2, \ldots, u_t \) of \( F \) over \( B \), any \( \alpha \in F \) can be
  uniquely determined given the values of \( \phi(u_\alpha) \) for \( i \in [t] \), i.e.
  \( \alpha = \sum_{i=1}^{t} \text{Tr}(u_i)u^*_i \), where \( \{u^*_i\}_{i=1}^{t} \) is the dual
  (trace-orthogonal) basis of \( \{u_i\}_{i=1}^{t} \) (see, for instance [24]
  Ch. 2, Def. 2.30).)
- When \( n - k \geq |B|^{|t-1|} \), the \( t \) target traces can be computed
  from \( n - 1 \) sub-symbols, referred to as the repair traces,
downloaded from the \( n - 1 \) available nodes. Thus, the repair
  bandwidth is \( n - 1 \) sub-symbols.

Note that the checks of \( C \) are precisely those polynomials \( p(x) \in F[x] \) where
\( \deg(p) < n-k \). It turns out that when \( n-k \geq |B|^{|t-1|} \), we can
define checks that take in the repair process via the trace function.
More specifically, for each \( u \in F \) and \( \alpha \in F \), we define the polynomial

\[
p_u(\alpha) = \text{Tr}(u(x-\alpha))/(x-\alpha). \tag{2}
\]

By the definition of the trace function, the following lemma follows in a straightforward manner.

Lemma 1 [10]. The polynomial \( p_u(\alpha) \) defined in (2) satisfies the
following properties.

(a) \( \deg(p_u(\alpha)) = |B|^{|t-1|} - 1 \); (b) \( p_u(\alpha) = u \).

By Lemma 1 (a), \( \deg(p_{u,\alpha}) = |B|^{|t-1|} - 1 < n-k \). Therefore,
the polynomial \( p_{u,\alpha}(x) \) corresponds to a codeword of \( C^\perp \) and is
a check for \( C \). Now let \( U = \{u_1, \ldots, u_t\} \) be a basis of \( F \) over
\( B \), and set

\[
p_i(x) = p_{u_i,\alpha^*}(x) = \text{Tr}(u_i(x-\alpha^*)/(x-\alpha^*), \quad i \in [t].
\]

These \( t \) polynomials correspond to \( t \) codewords of \( C^\perp \). Therefore,
we obtain \( t \) equations of the form

\[
p_i(\alpha^* )f(\alpha^*) = -\sum_{\alpha \in A \setminus \{\alpha^* \}} p_i(\alpha)f(\alpha), \quad i \in [t]. \tag{3}
\]

An essential step in the Guruswami-Wootters repair scheme is to
apply the trace function to both sides of (3) to obtain \( t \) different
repair equations

\[
\text{Tr}(p_i(\alpha^* )f(\alpha^*)) = -\sum_{\alpha \in A \setminus \{\alpha^* \}} \text{Tr}(p_i(\alpha)f(\alpha)), \quad i \in [t]. \tag{4}
\]

According to Lemma 1 (b), \( p_i(\alpha^*) = u_i \), for \( i = 1, \ldots, t \).
Moreover, by the linearity of the trace function, we also have

\[
\text{Tr}(p_i(\alpha)f(\alpha)) = \text{Tr}(u_i(\alpha - \alpha^*)) \times \text{Tr}
\]

\[\left(\frac{f(\alpha)}{\alpha - \alpha^*}\right). \tag{5}\]

Therefore, one can rewrite (4) as follows:

\[
\text{Tr}(u_i(\alpha - \alpha^*)) \times \text{Tr}(\frac{f(\alpha)}{\alpha - \alpha^*}), \quad i \in [t].
\]

The right-hand side sums of the equations (5) can be computed by
downloading the repair trace \( \text{Tr}(\frac{u_i(\alpha)}{\alpha - \alpha^*}) \) from the node
storing \( f(\alpha) \), for each \( \alpha \in A \setminus \{\alpha^* \} \). As a consequence, the
\( t \) independent target traces \( \text{Tr}(u_i(\alpha^*)) \), \( i = 1, \ldots, t \), of \( f(\alpha^*) \)
can be determined by downloading one sub-symbol from each of
the \( n - 1 \) available nodes. The erased symbol \( f(\alpha^*) \) can
subsequently be recovered from it \( t \) independent target traces.
By [25] Cor. 1, this scheme is bandwidth-optimal when \( n = |F| \)
and \( k = n(1 - 1/|B|) \).

III. REPAIRING REED-SOLOMON CODES WITH TWO
ERASURES

We consider the same setting as in Section II-C, i.e. \( n-k \geq |B|^{|t-1|} \),
where \( B = GF(p^m) \) and \( F = GF(p^{mt}) \), and assume
as before that \( C \) is a Reed-Solomon code \( RS(A,k) \) over \( F \) (see
Definition 1). However, we now suppose that two storage nodes
failed, or equivalently, that two codeword symbols, say \( f(\alpha) \) and
\( f(\beta) \), are erased. Two distributed repair schemes are proposed,
both of which use the same bandwidth per erasure as for the case of
a single erasure [10].

A typical distributed repair scheme consists of two phases: The
Download Phase and the Collaboration Phase. In the Download
Phase, the replacement nodes contact and download recovery
data from all available nodes. In the Collaboration Phase, the
replacement nodes collaborate by exchanging information among
themselves to help each other complete their repair processes.
The repair bandwidth (per erasure) is defined as the amount of
data each replacement node must download from the other nodes
during its repair process.
A. General Idea

We first discuss the challenges associated with repairing two erased symbols and then present some basic results that are useful for both Section III-B and Section III-C.

A check $p(x)$ is said to involve a codeword symbol $f(\alpha)$ if $p(\alpha) \neq 0$. Otherwise, $p(x)$ is said to exclude $f(\alpha)$. When only one symbol $f(\alpha^*)$ is erased, every check $p(x)$ that involves $f(\alpha^*)$ can be used to generate a repair equation as follows.

$$
\text{Tr}(p(\alpha^*)f(\alpha^*)) = - \sum_{\alpha \in A(\alpha^*)} \text{Tr}(p(\alpha)f(\alpha)).
$$

However, when two symbols $f(\alpha^*)$ and $f(\beta^*)$ are erased, in order to, say, recover $f(\alpha^*)$, we no longer have the freedom to use every possible check that involves $f(\alpha^*)$. Indeed, those checks that involve both $f(\alpha^*)$ and $f(\beta^*)$ cannot be used in a straightforward manner for repair, because we cannot simply compute the right-hand side sum of (6) without retrieving some information from $f(\beta^*)$.

The gist of our approach for the distributed repair schemes is to first generate those checks that only involve one erased symbol, $f(\alpha^*)$ or $f(\beta^*)$, but not both. We show that there exist $2(t-1)$ such checks, which are used in the Download Phase. Each RN uses $t - 1$ checks and downloads the corresponding $n - 2$ sub-symbols from each available node. Apart from the $t - 1$ checks that involve $f(\alpha^*)$ but not $f(\beta^*)$, and the $t - 1$ checks that involve $f(\beta^*)$ but not $f(\alpha^*)$, we also introduce two additional checks that involve both $f(\alpha^*)$ and $f(\beta^*)$, which are useful in the Collaboration Phase. It is not immediately clear how these last two checks can be used at all. However, we prove that when the extension degree $t$ is divisible by the characteristic of the field $F$, each erased symbol can be recovered at each RN using the aforementioned $t$ checks, at the cost of downloading in total $n - 1$ sub-symbols from $n - 2$ surviving nodes and from the other RN. In the first repair scheme, the two RNs exchange their repair data simultaneously (parallel repair), while in the second scheme, one node waits to receive the data from the other node before sending out its own repair data (sequential repair). By allowing one node to wait in the Collaboration Phase, we obtain a repair scheme that works for every field extension degree. Each distributed scheme can be easily modified to yield a centralized scheme by removing the Collaboration Phase.

To identify check equations that involve one codeword symbol $f(\alpha)$ but not the other symbol $f(\beta)$, we first introduce a special polynomial $Q_{\alpha,\beta}(z)$, defined as follows:

$$
Q_{\alpha,\beta}(z) = \text{Tr}(z(\beta - \alpha)).
$$

Let $K_{\alpha,\beta}$ denote the root space of $Q_{\alpha,\beta}(z)$. Then

$$
K_{\alpha,\beta} = \{z^* \in F : \text{Tr}(z^*\alpha) = \text{Tr}(z^*\beta)\}.
$$

Lemma 2. The following statements hold for every $\alpha$ and $\beta$ in $F$, $\alpha \neq \beta$.

(a) $K_{\alpha,\beta} \equiv K_{\beta,\alpha}$. In other words, the polynomial $Q_{\alpha,\beta}(z)$ and the polynomial $Q_{\beta,\alpha}(z)$ have the same root spaces.

(b) $\dim_B(K_{\alpha,\beta}) = \dim_B(K_{\beta,\alpha}) = t - 1$.

Proof. From (5), due to symmetry, $K_{\alpha,\beta} \equiv K_{\beta,\alpha}$. As the trace function is a linear mapping from $F$ to $B$, its kernel $K = \{k \in F : \text{Tr}(k) = 0\}$ is a $B$-subspace of dimension $t - 1$ (see, for instance [24] Thm. 2.23). Hence, the root space of $Q_{\alpha,\beta}(z)$, i.e., $K_{\alpha,\beta} = \frac{1}{\beta - \alpha} K$, is also a $B$-subspace of dimension $t - 1$.

We then use a root $z^*$ of the polynomial $Q_{\alpha,\beta}(z)$ to define a check equation according to (2).

$$
p_{z^*,\alpha}(x) = \text{Tr}(z^*(x - \alpha))/(x - \alpha).
$$

The following properties of $p_{z^*,\alpha}(x)$ will be used in our subsequent proofs.

Lemma 3. Suppose that $\alpha$ and $\beta$ are two distinct elements of $F$, and $z^*$ is a root of $Q_{\alpha,\beta}(z)$ or $Q_{\beta,\alpha}(z)$ in $F$, i.e., $z^* \in K_{\alpha,\beta}$. Then the following claim holds.

(a) $p_{z^*,\alpha}(\beta) = 0$.

Moreover, if the extension degree $t$ is divisible by $\text{char}(F)$ then

(b) $p_{\alpha,\alpha}(\beta)$ is a root of $Q_{\alpha,\beta}(z)$ and $Q_{\beta,\alpha}(z)$, for every $u \in F$, and so is $1/(\beta - \alpha)$.

Proof. Note that according to Lemma 2(a), the root spaces of $Q_{\alpha,\beta}(z)$ and $Q_{\beta,\alpha}(z)$ are the same. The first claim is clear based on the definitions of $Q_{\alpha,\beta}(z)$ and $p_{z^*,\alpha}(x)$. For the second claim, it is sufficient to show that both $1/(\beta - \alpha)$ and $p_{\alpha,\alpha}(\beta)$ is a root of $Q_{\alpha,\beta}(z)$. By the definition of $Q_{\alpha,\beta}(z)$, we have $Q_{\alpha,\beta}(z) = \text{Tr}(z(\beta - \alpha))$. Therefore,

$$
Q_{\alpha,\beta}(1/(\beta - \alpha)) = \text{Tr}(1) = \sum_{i=0}^{t-1} 1 = t = 0,
$$

whenever $t$ is divisible by the characteristic of the field. Hence $1/(\beta - \alpha)$ is a root of $Q_{\alpha,\beta}(z)$. Since $p_{\alpha,\alpha}(\beta) = b/(\beta - \alpha)$, where $b = \text{Tr}(u(\beta - \alpha)) \in B$, we deduce that

$$
Q_{\alpha,\beta}(p_{\alpha,\alpha}(\beta)) = bQ_{\alpha,\beta}(1/(\beta - \alpha)) = 0,
$$

because of the linearity of $Q_{\alpha,\beta}(z)$. Thus, $p_{\alpha,\alpha}(\beta)$ is also a root of $Q_{\alpha,\beta}(z)$.

The following lemma restates what is shown in Section III-C.

Lemma 4. For $\alpha \neq \alpha^*$ and $u \in F$,

$$
\text{Tr}(p_{\alpha,\alpha^*}(\alpha)f(\alpha)) = \text{Tr}(u(\alpha - \alpha^*)) \times \text{Tr}\left(\frac{f(\alpha)}{\alpha - \alpha^*}\right).
$$

Hence, the trace $\text{Tr}(p_{\alpha,\alpha^*}(\alpha)f(\alpha))$ can be determined by downloading the repair trace $\text{Tr}\left(\frac{f(\alpha)}{\alpha - \alpha^*}\right)$ from the node storing $f(\alpha)$.

B. A One-Round Distributed Repair Scheme for Two Erasures

The scheme comprises of two phases, the Download Phase, where each RN contacts and downloads data from the other $n - 2$ available nodes, and the Collaboration Phase, where the two RNs exchange the data, based on what they receive earlier in the Download Phase. The main task is to design the data to be exchanged during the two phases. This task can be completed via a selection of proper check polynomials to be used by each RN. We discuss the generation of these polynomials below.

Let $K_{\alpha,\pi}$ be the root space of the polynomial $Q_{\alpha,\pi}(z)$. By Lemma 2(b), $\dim_B(K_{\alpha,\pi}) = t - 1$. Let $U = \{u_1, u_2, \ldots, u_{t-1}\} \subseteq F$ and $V = \{v_1, v_2, \ldots, v_{t-1}\} \subseteq F$ be two arbitrary bases of $K_{\alpha,\pi}$ over $B$. We extend $U$ and $V$ to obtain the two bases $U' = \{u_1, \ldots, u_t\}$ and $V' = \{v_1, \ldots, v_t\}$ of $F$ over $B$, respectively.

Note that while it is not crucial to choose two different sets $U$ and $V$, it is not compulsory to use the same set $U \equiv V$ either. For $i \in [t]$, we set

$$
p_i(x) \triangleq p_{u_i,\alpha^*}(x) = \text{Tr}(u_i(x - \alpha^*))/(x - \alpha^*),
$$

$$
q_i(x) \triangleq p_{v_i,\alpha^*}(x) = \text{Tr}(v_i(x - \alpha))/(x - \alpha).
$$
**Download Phase.** In this phase, each RN contacts \( n-2 \) available nodes to download repair data. To determine what to download, the RN for \( f(\alpha^*) \) uses the first \( t-1 \) checks \( p_1, \ldots, p_{t-1} \) to construct the following \( t-1 \) repair equations.

\[
\text{Tr}(p_1(\alpha^*)f(\alpha^*)) = - \sum_{\alpha \in A \setminus \{\alpha^*, \alpha^* \}} \text{Tr}(p_1(\alpha)f(\alpha)), \quad i \in [t-1].
\]

Similarly, the RN for \( f(\overline{\alpha}) \) creates the following repair equations.

\[
\text{Tr}(q_i(\alpha)f(\alpha)) = - \sum_{\alpha \in A \setminus \{\alpha^*, \overline{\alpha} \}} \text{Tr}(q_i(\alpha)f(\alpha)), \quad i \in [t-1].
\]

By Lemma 3 (a), we have \( p_i(\alpha) = 0 \) and \( q_i(\alpha^*) = 0 \) for all \( i = 1, \ldots, t-1 \). Therefore, the right-hand sides of (11) and (12) do not involve \( f(\alpha^*) \) and \( f(\overline{\alpha}) \). As a result, each RN can recover \( t-1 \) independent target traces of the corresponding erased symbol by downloading \( n-2 \) sub-symbols (repair traces) from the available nodes. Corollary 1 which follows directly from Lemma 3 formally states this fact.

**Corollary 1.** In the Download Phase, the replacement node for \( f(\alpha^*) \) can recover \( t-1 \) independent target traces, namely \( \text{Tr}(p_1(\alpha^*)f(\alpha^*)), \ldots, \text{Tr}(p_{t-1}(\alpha^*)f(\alpha^*)) \), by downloading \( n-2 \) repair traces, i.e. \( \text{Tr}(\frac{f(\alpha)}{\alpha^*}) \) from the available node storing \( f(\alpha) \), for all \( \alpha \in A \setminus \{\alpha^*, \overline{\alpha} \} \). A similar statement holds for the replacement node for \( f(\overline{\alpha}) \), where the checks are \( q_i \) and the repair traces are \( \text{Tr}(\frac{f(\alpha)}{\alpha^*}) \).

**Collaboration Phase.** As one more independent target trace of each erased symbol is needed for a complete recovery, the two RNs create two additional repair equations for \( f(\alpha^*) \), \( f(\overline{\alpha}) \), respectively.

\[
\text{Tr}(p_1(\alpha^*)f(\alpha^*)) + \text{Tr}(p_1(\overline{\alpha})f(\overline{\alpha})) = - \sum_{\alpha \in A \setminus \{\alpha^*, \overline{\alpha} \}} \text{Tr}(p_1(\alpha)f(\alpha)).
\]

\[
\text{Tr}(q_i(\overline{\alpha})f(\overline{\alpha})) + \text{Tr}(q_i(\alpha^*)f(\alpha^*)) = - \sum_{\alpha \in A \setminus \{\alpha^*, \overline{\alpha} \}} \text{Tr}(q_i(\alpha)f(\alpha)).
\]

It is clear that from the repair traces \( \text{Tr}(\frac{f(\alpha)}{\alpha^*}), \alpha \in A \setminus \{\alpha^*, \overline{\alpha} \} \), retrieved in the Download Phase, the RHS of (13) can be determined. However, to extract the target trace \( \text{Tr}(p_1(\alpha^*)f(\alpha^*)) \), the RN for \( f(\alpha^*) \) needs to know the term \( \text{Tr}(p_1(\overline{\alpha})f(\overline{\alpha})) \) which would have been determined based on the repair trace \( \text{Tr}(\frac{f(\alpha)}{\overline{\alpha}}) \) downloaded from the node storing \( f(\overline{\alpha}) \) if it had not failed. The following lemma states that for certain field extension degrees, this missing piece of information can be created by the RN for \( f(\overline{\alpha}) \) based on what it obtains in the Download Phase. It can then send this repair trace to the RN for \( f(\alpha^*) \) to help complete the recovery of that symbol. A similar scenario also holds for \( f(\overline{\alpha}) \).

**Lemma 5.** If the field expansion degree \( t \) is divisible by \( \text{char}(F) \) then both \( p_1(\overline{\alpha}) \) and \( 1/(\overline{\alpha} - \alpha^*) \) can be written as \( B \)-linear combinations of elements in \( V = \{q_i(\alpha^*): i \in [t-1]\} = \{v_1, \ldots, v_{t-1}\} \). Likewise, both \( q_i(\alpha^*) \) and \( 1/(\alpha^* - \overline{\alpha}) \) can be written as \( B \)-linear combinations of elements in \( U = \{p_i(\alpha^*): i \in [t-1]\} = \{u_1, \ldots, u_{t-1}\} \).

**Proof.** Because of symmetry, it suffices to just prove the first statement of the lemma. By Lemma 1 (b), we have \( q_i(\overline{\alpha}) = v_i \), for every \( i \in [t-1] \). Therefore, \( \{q_i(\overline{\alpha}) : i \in [t-1]\} = \{v_1, \ldots, v_{t-1}\} = V \), which is a basis of the root space \( K_{\alpha^*, \overline{\alpha}} \) of the polynomial \( Q_{\alpha^*, \overline{\alpha}}(z) \). Therefore, in order to show that \( p_1(\overline{\alpha}) \) and \( 1/(\overline{\alpha} - \alpha^*) \) are \( B \)-linear combinations of elements in \( V \), it is sufficient to prove that they are roots of \( Q_{\alpha^*, \overline{\alpha}}(z) \). But this follows immediately from Lemma 3 (b), because \( p_1(x) \) equals \( p_{u_1, \alpha^*}(x) \) by its definition in (9).

From the linearity of the trace function, we arrive at the following corollary of Lemma 5.

**Corollary 2.** If the field expansion degree \( t \) is divisible by \( \text{char}(F) \) then the trace \( \text{Tr}(p_1(\overline{\alpha})f(\overline{\alpha})) \) and the repair trace \( \text{Tr}(\frac{f(\alpha)}{\alpha^*}) \) can be written as \( B \)-linear combinations of the target traces in \( \{\text{Tr}(q_i(\overline{\alpha})f(\overline{\alpha})): i \in [t-1]\} \). Also, the trace \( \text{Tr}(q_i(\alpha^*)f(\alpha^*)) \) the repair trace \( \text{Tr}(\frac{f(\alpha)}{\alpha^*}) \) can be written as \( B \)-linear combinations of the target traces in \( \{\text{Tr}(p_i(\alpha^*)f(\alpha^*)): i \in [t-1]\} \). Moreover, the coefficients of these combinations do not depend on \( f \).
Proof. By Lemma 1(b), $p_i(\alpha^*) = u_i$ and $q_i(\tau) = v_i$ for $i \in [t]$. Recall that the sets $U' = \{u_1, \ldots, u_t\}$ and $V' = \{v_1, \ldots, v_t\}$ are both linearly independent over $B$. Therefore, after the two phases, each RN obtains $t$ independent target traces for each erased symbol, $\text{Tr}(u_i f(\alpha^*))$, for $f(\alpha^*)$, and $\text{Tr}(v_i f(\tau))$, for $f(\tau)$, for all $i \in [t]$. Thus, each erased symbol will have $t$ independent traces for its recovery. Each RN downloads $n - 2$ sub-symbols in the Download Phase and one sub-symbol in the Collaboration Phase, according to Corollary 1 and Lemma 6 which sum up to a repair bandwidth of $n - 2$ sub-symbols.

C. A Two-Round Distributed Repair Scheme for Two Erasures

We modify the one-round repair scheme developed in the previous subsection to obtain a two-round scheme that works for all field extension degrees. We still generate the checks $p_1, \ldots, p_t$ and $q_1, \ldots, q_t$ as in the first scheme, given by (9) and (10), respectively. However, the RN for $f(\alpha^*)$, instead of $p_t$, uses the following checks

$$p_i^t(x) \triangleq \tau p_i(x), \quad \text{for all } i = 1, \ldots, t. \quad (15)$$

where $\tau$ is chosen so that $\tau \in F \setminus \{0\}$ and $\frac{\tau f(\alpha^*)}{\alpha - \alpha^*} \in K_{n - \alpha^*}$. For instance, we can set $\tau = (\tau - \alpha^*)q_1(\tau)$. Clearly, $\text{deg}(p_i^t) = \text{deg}(p_i) < n - k$ and hence, $p_i^t(x)$ serve as check polynomials of the code. Note that for this scheme to work, the RN for $f(\alpha^*)$ must know $v_1 = q_1(\tau)$, so that it can compute $\tau$. This assumption is satisfied as long as every RN uses a known deterministic repair scheme, which should be the case in practice.

Lemma 7. The check polynomials $p_i^t(x)$ defined as in (15) satisfy the following properties.

(P1) $p_1^t(\tau) = p_2^t(\tau) = \cdots = p_{t-1}^t(\tau) = 0$.

(P2) $\{p_1^t(\alpha^*), \ldots, p_t^t(\alpha^*)\}$ is a basis of $F$ over $B$.

(P3) $\text{Tr}(p_i^t(\tau) f(\tau))$ and $\text{Tr}(\frac{\tau f(\tau)}{\alpha - \alpha^*})$ are $B$-linear combinations of elements in $\{q_1(\tau)f(\tau), \ldots, q_{t-1}(\tau)f(\tau)\}$. Moreover, the coefficients in these combinations do not depend on $f$.

Proof. Property (P1) holds since $p_i^t(\tau) = \tau p_i(\tau)$ and $p_t(\tau) = 0$ for all $i \in [t-1]$ by Lemma 5(a). Property (P2) follows from the fact that $p_i^t(\alpha^*) = \tau p_i(\alpha^*)$, $\tau \neq 0$, and that $U' = \{u_1, \ldots, u_t\} = \{p_1(\alpha^*), \ldots, p_t(\alpha^*)\}$ is a $B$-basis of $F$. Property (P3) holds due to the definition of $\tau$ and the linearity of trace.

Download Phase. In this phase, the RN for $f(\alpha^*)$ uses the first $t - 1$ checks $p_1^t, \ldots, p_{t-1}^t$ to construct the $t - 1$ repair equations. For $i = 1, \ldots, t - 1$,

$$\text{Tr}(p_i^t(\alpha^*)f(\alpha^*)) = - \sum_{\alpha \in A \setminus \{\alpha^*\}} \text{Tr}(p_i^t(\alpha)f(\alpha)) = - \sum_{\alpha \in A \setminus \{\alpha^*\}} \text{Tr}(u_i(\alpha - \alpha^*)) \times \text{Tr}(\frac{\tau f(\alpha)}{\alpha - \alpha^*}). \quad (16)$$

By Lemma 7(P1), the RHS of (16) does not involve $f(\tau)$. Thus, the RN for $f(\alpha^*)$ can determine $t - 1$ target traces $\text{Tr}(p_i^t(\alpha^*)f(\alpha^*))$, $i \in [t - 1]$, of $f(\alpha^*)$ by downloading $n - 2$ sub-symbols $\text{Tr}(\frac{\tau f(\alpha)}{\alpha - \alpha^*})$ from the available nodes storing $f(\alpha^*)$, $\alpha \in A \setminus \{\alpha^*, \tau\}$. The RN for $f(\tau)$ follows the same procedure as in the first scheme (Section III-B).

Collaboration Phase. The last repair equation for $f(\alpha^*)$ is

$$\text{Tr}(p_t^t(\alpha^*)f(\alpha^*)) + \text{Tr}(p_t^t(\tau)f(\tau)) = - \sum_{\alpha \in A \setminus \{\alpha^*, \tau\}} \text{Tr}(p_t^t(\alpha)f(\alpha)) = - \sum_{\alpha \in A \setminus \{\alpha^*, \tau\}} \text{Tr}(u_t(\alpha - \alpha^*)) \times \text{Tr}(\frac{\tau f(\alpha)}{\alpha - \alpha^*}). \quad (17)$$

Clearly, the RN for $f(\alpha^*)$ can compute the RHS of (17) based on what it downloaded in the Download Phase. To obtain the last target trace $\text{Tr}(p_t^t(\alpha^*)f(\alpha^*))$, it computes $\text{Tr}(p_t^t(\tau)f(\tau)) = \text{Tr}(u_t(\alpha - \alpha^*)) \times \text{Tr}(\frac{\tau f(\alpha)}{\alpha - \alpha^*})$, where the repair trace $\text{Tr}(\frac{\tau f(\alpha)}{\alpha - \alpha^*})$
can be downloaded from the RN for \( f(\bar{\pi}) \). This is possible because the repair trace is a \( B \)-linear combination of \( \{ q_1(\bar{\pi}) f(\bar{\pi}), \ldots, q_{t-1}(\bar{\pi}) f(\bar{\pi}) \} \), due to Lemma 7 (P3), which is already available at the RN for \( f(\bar{\pi}) \). Then, by Lemma 7 (P2), the RN for \( f(\alpha^*) \) has \( t \) independent traces of \( f(\alpha^*) \) to recover this lost symbol. As \( f(\alpha^*) \) has been recovered, the RN for \( f(\bar{\pi}) \) downloads the repair trace \( \text{Tr}(f(\alpha^*/\alpha)) \) from the RN for \( f(\alpha^*) \) to compute the target trace \( \text{Tr}(q_{t-1}(\bar{\pi}) f(\bar{\pi})) \), and then can recover \( f(\bar{\pi}) \) completely. Note that the RN for \( f(\alpha^*) \) has to first receive the repair trace from the RN for \( f(\bar{\pi}) \) before computing and sending out its repair trace for \( f(\bar{\pi}) \) (see Fig. 3). Theorem 2 summarizes the discussion.

**Theorem 2.** The two-round distributed repair scheme can be used to repair any two erased symbols of a Reed-Solomon code \( RS(A, k) \) at a repair bandwidth of \( n-1 \) sub-symbols per erased symbol, for any extension degree \( t \), given that \( n-k \geq |B|^{t-1}. \)

**Remark 1.** In our repair scheme, each RN uses a bandwidth of \( n-1 \) sub-symbols. In a naive scheme, one RN first downloads \( kt \) sub-symbols from a set of \( k \) available nodes, recovers both erased symbols, and then sends the corresponding symbol to the other RN. The total bandwidth used is \( kt + t \), which is worse than the method we described if \( \frac{k+1}{n} > \frac{2}{t} \), i.e., when \( t \) is sufficiently large or when the code has high rate.

**Example 2.** Let \( B = GF(2) \), \( F = GF(8) \), \( n = |F| = 8 \), and \( k = n(1-1/|B|) = 4 \). The extension degree is \( t = 3 \), which is not divisible by the characteristic of the fields. Let \( \xi \) be a primitive element of \( F \) that satisfies \( \xi^3 + \xi + 1 = 0 \). Let \( A = \{0, 1, \xi, \xi^2, \xi^3, \xi^4\} \equiv F \) be the set of evaluation points and \( C = RS(A, k) \) be the Reed-Solomon code defined as in Definition 1.

Let \( f(x) \) be a polynomial over \( F[x] \) of degree at most 3. Then \( e_f = \{ f(0), f(1), \ldots, f(\xi^4) \} \) is a codeword of \( C \). Suppose that the first two codeword symbols \( f(0) \) and \( f(1) \) are erased. So \( \alpha^* = 0 \) and \( \bar{\pi} = 1 \). We now demonstrate how to construct a two-round distributed repair scheme for these two erasures. Firstly, we have \( Q_{\alpha^*, \bar{\pi}}(z) = \text{Tr}(z(\bar{\pi}-\alpha^*)) = \text{Tr}(z) = z^4 + z^2 + z \). We then choose the bases \( U \) and \( V \) for the root space \( K_{\alpha^*, \bar{\pi}} \) of \( Q_{\alpha^*, \bar{\pi}}(z) \) as \( U = V = \{ \xi, \xi^2 \} \). These sets \( U \) and \( V \) can be completed to \( F_2 \)-bases \( U' = V' = \{ \xi, \xi^2, \xi^3 \} \) of \( F \). The checks \( p_i \) for \( f(0) \) and \( q_i \) for \( f(1) \), \( i = 1, 2, 3 \), are constructed as follows.

\[
\begin{align*}
p_1(x) &= \text{Tr}(\xi(x-0))/(x-0) = \xi^4 + x^3 + \xi^2 x + \xi, \\
p_2(x) &= \text{Tr}(\xi^2(x-0))/(x-0) = \xi x^3 + \xi^4 x + \xi^2, \\
p_3(x) &= \text{Tr}(\xi^3(x-0))/(x-0) = \xi^5 x^3 + \xi^6 x + \xi^5, \\
p_1(1) &= \text{Tr}(\xi(\bar{\pi}-1))/(\bar{\pi}-1) = \xi^4 + \xi^4 x^2 + \xi, \\
p_2(1) &= \text{Tr}(\xi^2(\bar{\pi}-1))/(\bar{\pi}-1) = \xi x^3 + \xi^2 x^2 + \xi^2 x, \\
p_3(1) &= \text{Tr}(\xi^3(\bar{\pi}-1))/(\bar{\pi}-1) = \xi^5 x^3 + \xi^5 x^2 + \xi x + 1.
\end{align*}
\]

Note that in the two-round scheme, we use \( p_i(x) \) instead of \( p_i(1) \). Since \( \tau = (1-0)q_1(1) = \xi, \) we can determine \( p_i(x) \) as follows.

\[
\begin{align*}
p_1(x) &= \xi \xi^4 + x^3 + \xi^2 x + \xi, \\
p_2(x) &= \xi^2 x^3 + \xi^4 x + \xi^2, \\
p_3(x) &= \xi^6 x^3 + \xi^6 x + \xi^4.
\end{align*}
\]

The coordinates of the checks are given in Table [III]. These coefficients are precisely the coefficients to be used in the repair equations. The reader may refer to the caption of the table for the explanation of how this particular scheme works.

**D. Centralized Repair Schemes for Two Erasures**

In a centralized repair scheme, a repair center carries out the recovery of both erased symbols simultaneously at one place. Then it distributes the recovered symbols to the corresponding RNs. The repair bandwidth is defined to be the total amount of information the repair center downloads from the surviving nodes divided by the number of erased symbols. We can easily obtain a centralized repair scheme from each of the previously developed distributed schemes by simply letting the repair center download all the repair traces that both the RNs are supposed to download from other available nodes in the Download Phase. As the Collaboration Phase is no longer needed, the amount of information the repair center downloads from the \( n-2 \) available nodes is at most \( n-2 \) sub-symbols in \( B \) per erased symbol.

**Theorem 3.** The centralized repair schemes deduced from the two distributed repair schemes in Section [III-B] and Section [III-C] can recover any two erased symbols \( f(\alpha^*) \) and \( f(\bar{\pi}) \) at a repair bandwidth of \( n-2 \) sub-symbols in \( B \) per erased symbol. The first scheme requires that \( \text{char}(F) \) divides \( t \), while the second one works for every \( t \).

**Remark 2.** Whenever \( 2(n-2) < kt \), or equivalently, \( \frac{k}{n-2} > \frac{2}{t} \), the centralized schemes developed in this section have a strictly smaller repair bandwidth than the naive scheme.

**IV. REPAIRING REED-SOLOMON CODES WITH THREE ERASURES**

We extend our centralized and distributed schemes developed in Section [III] to address the case when three codeword symbols are erased in an RS code. We assume that \( n-k \leq |B|^{t-1} \) and that the field extension degree \( t \) is divisible by the characteristic of the field. The unique feature of these repair schemes is the concept of a repair cycle, in which the computation of a target trace enables the computation of another target trace, and so forth. The target traces found during the cycle are not known in advance, and are only determined once an activating trace is found, based on the approach of Theorem 3 and Theorem 5. When two such repair cycles are completed, \( t \) independent traces for each of the three erased symbols are obtained. Consequently,
TABLE III: (Example 2) The table of dual codewords (checks) used in the two-round distributed scheme to repair two erased codeword symbols \( f(0) \) and \( f(1) \), which correspond to the first two columns. The top row lists the 8 evaluation points. A dot “.” means a zero entry. The basis \( \xi^U = \{ \xi^2, \xi^3, \xi^4 \} \) of \( F = \text{GF}(8) \) over \( B = \text{GF}(2) \) corresponds to the evaluations of the three checks \( p_1^*, p_2^*, p_3^* \) at \( \alpha^* = 0 \). Similarly, the basis \( \xi^V = \{ \xi, \xi^2, \xi^3 \} \) corresponds to the evaluations of the three checks \( q_1, q_2, q_3 \) at \( \alpha = 1 \).

Within the first three rows, except the first two columns, the nonzero symbols in each column are all the same. This property reflects the fact that to compute the right-hand side sums in the repair equations generated by \( p_1^* \)'s, for each \( \alpha \neq \{0, 1\} \), the RN for \( f(0) \) only needs to download one trace (bit) \( Tr(f(\alpha)) \) from the node storing \( f(\alpha) \). Also, the first two rows have all zero entries at the second column, which means the first two checks \( p_1^* \) and \( p_2^* \) involve \( f(0) \) but not \( f(1) \). As a consequence, in the Download Phase, the first two independent traces of \( f(0), Tr(p_1^*(0)f(0)) \) and \( Tr(p_2^*(0)f(0)) \), can be determined from \( 6 = 8 - 2 \) downloaded bits. The third check \( p_3^* \) involves both \( f(0) \) and \( f(1) \), as described in the scheme. However, because of its definition, \( p_3^*(1) = \xi = q_1(1) \). This allows the RN for \( f(0) \) to compute the term \( Tr(p_3^*(1)f(1)) = Tr(q_1(1)f(1)) \) in the repair equation generated by \( p_3^* \), by downloading \( Tr(q_1(1)f(1)) \) from the RN for \( f(1) \) in the Collaboration Phase. The RN for \( f(0) \) can then extract the third trace of \( f(0) \), namely \( Tr(p_3^*(0)f(0)) \), from the repair equation generated by \( p_3^*(x) \). It now has three independent traces \( Tr(p_i^*(0)f(0)) \), \( i = 1, 2, 3 \), to completely recover the erased symbol \( f(0) \). It then sends the repair trace \( Tr(\frac{f(0)}{\beta}) \), or equivalently, \( Tr(q_i(0)f(0)) \) to the RN for \( f(1) \). The RN for \( f(1) \) already obtained two independent traces for \( f(1) \), namely \( Tr(q_i(1)f(1)) \), \( i = 1, 2 \), in the Download Phase, by downloading six bits from other available nodes. Once it receives the repair trace from the RN for \( f(0) \) in the Collaboration Phase, it can figure out the third trace \( Tr(q_3(1)f(1)) \) and complete the recovery of \( f(1) \). The repair bandwidth per erased symbol is seven bits.

A. A Centralized Three-Erasure Repair Scheme for Reed-Solomon Codes

**Theorem 4.** Suppose \( f(x) \in F[x] \) is a polynomial of degree at most \( k - 1 \), which corresponds to a codeword in a Reed-Solomon code \( C = \text{RS}(A,k) \). Assume that \( f(\alpha^*), f(\bar{\alpha}) \), and \( f(\bar{\alpha}^*) \) are the three erased codeword symbols. Given that the field extension degree \( t \) is divisible by the characteristic of \( F \), and that

\[
\left \{ \frac{\alpha - \alpha^*}{\alpha - \bar{\alpha}}, \frac{\alpha' - \bar{\alpha}}{\alpha' - \bar{\alpha}} \right \} \cap K \neq \emptyset
\]

holds, where \( K = \ker(Tr) \) is the kernel of the trace function, there exists a centralized repair scheme that recovers the three erased symbols by downloading three sub-symbols from each surviving node. In other words, the repair bandwidth of the scheme is \( n - 3 \) sub-symbols per erased symbol.

We divide our repair process into two phases. In the **Download Phase**, \( s \in \{t - 2, t - 1\} \) independent target traces for each erased symbol are produced at the cost of downloading a total of \( 3(n - 3) \) sub-symbols from \( n - 3 \) surviving nodes. We emphasize that no additional data download is needed afterward. In the **Collaboration Phase**, one or two additional target traces are generated for each erased symbol, depending on whether \( s = t - 1 \) or \( s = t - 2 \). In the more complicated case when \( s = t - 2 \), the Collaboration Phase involves two repair cycles, each of which produces three target traces. Within each cycle, the determination of one target trace leads to the determination of another target trace.

**Download Phase.** The repair center starts the whole repair process by producing \( s \in \{t - 2, t - 1\} \) independent target traces for each erased symbol. Lemma 8 explains the traces construction process. Recall that for any three distinct elements \( \alpha, \beta, \gamma \) in \( F \), one can define the polynomials \( Q_{\alpha,\beta}(z), Q_{\beta,\gamma}(z), \) and \( Q_{\gamma,\alpha}(z) \), according to [7]. Moreover, by [8], the intersection of the root spaces of any two among these three polynomials is

\[
K_{\alpha,\beta,\gamma} = \{ z^\ast \in F : Tr(z^\ast \alpha) = Tr(z^\ast \beta) = Tr(z^\ast \gamma) \}
\]

which is precisely the intersection of the root spaces of all three polynomials. We are interested in the case when \( \{ \alpha, \beta, \gamma \} = \{\alpha^*, \bar{\alpha}, \alpha'\} \).

**Lemma 8.** Let \( \alpha, \beta, \) and \( \gamma \) be three distinct elements of \( F \cong B^t \). Then

\[
t - 2 \leq \dim_B(K_{\alpha,\beta,\gamma}) \leq t - 1.
\]

**Proof.** We have \( K_{\alpha,\beta,\gamma} = K_{\alpha,\beta} \cap K_{\beta,\gamma} \). From Lemma 2 (b), \( K_{\alpha,\beta} \) and \( K_{\beta,\gamma} \) both have dimension \( t - 1 \). Hence,

\[
\dim_B(K_{\alpha,\beta,\gamma}) \leq t - 1.
\]

Moreover, we also have

\[
K_{\alpha,\beta} = \frac{1}{\beta - \alpha} K, \quad K_{\beta,\gamma} = \frac{1}{\gamma - \beta} K,
\]

where \( K = \ker(Tr) \). We may hence write

\[
|K_{\alpha,\beta,\gamma}| = \left| \left( \frac{1}{\beta - \alpha} K \right) \cap \left( \frac{1}{\gamma - \beta} K \right) \right| = \left| \left( \frac{\gamma - \beta}{\beta - \alpha} K \right) \cap K \right|
\]

and for \( \eta = \frac{\gamma - \beta}{\beta - \alpha} \), the previous expression reduces to

\[
|K_{\alpha,\beta,\gamma}| = |\eta K \cap K|.
\]

(18)
Consider the linear mapping $\sigma: K \to B$, where $\sigma(\kappa) = \text{Tr}(\eta \kappa)$. Then
$$\ker(\sigma) = \mathbb{K} K \cap K.$$ (19)

Since $\dim_B(B) = t - 1$, from (18) and (19) we deduce
$$|K_{\alpha, \beta, \gamma}| = |\ker(\sigma)| \geq |B|^{\dim_B(K) - \dim_B(B)} = |B|^{t - 2}.$$ (19)

Therefore, $\dim_B(K_{\alpha, \beta, \gamma}) \geq t - 2$.

According to Lemma 3, $\alpha \equiv \dim_B(K_{\pi, \alpha, t}) \in \{t - 2, t - 1\}$. We now select three $B$-bases $U = \{u_1, \ldots, u_s\}$, $V = \{v_1, \ldots, v_t\}$, and $W = \{w_1, \ldots, w_s\}$ of $K_{\alpha, \pi, \alpha}$. Note that one can set $U$, $V$, and $W$ to the same basis. Here we keep the setting general and assume that they can be any three bases of $K_{\alpha, \pi, \alpha}$. Based on these three sets, we may define three types of checks, namely:

1. $p_i(\pi) \equiv p_{ui, \alpha}(x) = \text{Tr}(u_i(x - \alpha)) / (x - \alpha)$, $i \in [s]$.
2. $q_i(\pi) \equiv p_{ui, \pi}(x) = \text{Tr}(u_i(x - \pi)) / (x - \pi)$, $i \in [s]$.
3. $r_i(\pi) \equiv p_{ui, \alpha}(x) = \text{Tr}(w_i(x - \alpha)) / (x - \alpha)$, $i \in [s]$.

From Lemma 1, these checks satisfy the following properties:

- $\{p_1(\pi), \ldots, p_s(\pi)\} \equiv U$, since $p_i(\pi) = u_i$,
- $\{q_1(\pi), \ldots, q_s(\pi)\} \equiv V$, since $q_i(\pi) = v_i$,
- $\{r_1(\pi), \ldots, r_s(\pi)\} \equiv W$, since $r_i(\pi) = w_i$.

where $\equiv$ stands for set equivalence. Moreover, by Lemma 3(a), the following additional properties also hold:

- $p_i(\pi) = 0$ and $p_i(\pi) = 0$, for $i \in [s]$.
- $q_i(\pi) = 0$ and $q_i(\alpha) = 0$, for $i \in [s]$.
- $r_i(\pi) = 0$ and $r_i(\pi) = 0$, for $i \in [s]$.

These $3s$ checks have support patterns as listed in Table IV. We summarize these facts in Lemma 9.

**Lemma 9.** The following statements hold.

- $p_i(x)$ involves $f(\alpha)$, but excludes $f(\pi)$ and $f(\alpha)$, $i \in [s]$.
- $q_i(x)$ involves $f(\pi)$, but excludes $f(\alpha)$ and $f(\alpha)$, $i \in [s]$.
- $r_i(x)$ involves $f(\alpha)$, but excludes $f(\alpha)$ and $f(\pi)$, $i \in [s]$.

| Checks | Erased Positions | $\alpha$ | $\pi$ | $\alpha'$ | Purpose |
|--------|------------------|---------|-------|-----------|---------|
| $p_1, \ldots, p_s$ | $\times$ | $\times$ | $\times$ | $\rightarrow$ target traces for $f(\alpha)$ |
| $q_1, \ldots, q_s$ | $\times$ | $\times$ | $\times$ | $\rightarrow$ target traces for $f(\pi)$ |
| $r_1, \ldots, r_s$ | $\times$ | $\times$ | $\times$ | $\rightarrow$ target traces for $f(\alpha)$ |

Note that by Lemma 2(b), the root spaces of all polynomials $Q_{\pi}(z)$ listed above have dimension $t - 1$. Since $s = t - 2$, $\dim_B(K_{\alpha, \pi} \cap K_{\alpha, \alpha}) = \dim_B(K_{\alpha, \pi, \alpha}) = t - 2$. Therefore, by Lemma 4, one can determine $s$ independent traces for each erased symbol by downloading three sub-bases (repair traces) from each surviving node, namely $\text{Tr}(\frac{f(\alpha)}{\alpha - \alpha})$, $\text{Tr}(\frac{f(\alpha)}{\alpha - \pi})$, and $\text{Tr}(\frac{f(\alpha)}{\alpha - \pi})$. In summary, after this phase, the repair center obtains the following target traces for the erased symbols:

- $s$ traces $\text{Tr}(u_i f(\alpha))$, $i \in [s]$, for $f(\alpha)$,
- $s$ traces $\text{Tr}(v_i f(\pi))$, $i \in [s]$, for $f(\pi)$,
- $s$ traces $\text{Tr}(w_i f(\alpha))$, $i \in [s]$, for $f(\alpha)$.

**Complement Phase.** By Lemma 5, $s \in \{t - 2, t - 1\}$. We consider the following two cases, depending on the value of $s$.

**Case 1:** $s = t - 2$. The repair center needs two more target traces to complete the recovery of each erased symbol. To that end, six additional checks, $p_{t-1}, p_t, q_{t-1}, q_t, r_{t-1}$, and $r_t$, are constructed, based on the following field elements (see Fig. 4):

- $u_{t-1} \in F$, so that $U \cup \{u_{t-1}\} = \{u_1, \ldots, u_s, u_{t-1}\}$ generates the root space of $K_{\alpha, \pi} \cap K_{\alpha, \alpha}$.
- $u_t \in F$, so that $U \cup \{u_t\} = \{u_1, \ldots, u_s, u_{t-2}, u_t\}$ generates the root space of $K_{\alpha, \pi} \cap K_{\alpha, \alpha}$.
- $v_{t-1} \in F$, so that $V \cup \{v_{t-1}\} = \{v_1, \ldots, v_{t-2}, v_{t-1}\}$ generates the root space of $K_{\alpha, \pi} \cap K_{\alpha, \alpha}$.
- $v_t \in F$, so that $V \cup \{v_t\} = \{v_1, \ldots, v_{t-2}, v_t\}$ generates the root space of $K_{\alpha, \pi} \cap K_{\alpha, \alpha}$.
- $w_{t-1} \in F$, so that $W \cup \{w_{t-1}\} = \{w_1, \ldots, w_{t-2}, w_{t-1}\}$ generates the root space of $K_{\alpha, \pi} \cap K_{\alpha, \alpha}$.
- $w_t \in F$, so that $W \cup \{w_t\} = \{w_1, \ldots, w_{t-2}, w_t\}$ generates the root space of $K_{\alpha, \pi} \cap K_{\alpha, \alpha}$.

![Fig. 4: The hierarchy of the $B$-subspaces of $F \cong B^t$.](image)

Note that by Lemma 2(b), the root spaces of all polynomials $Q_{\pi}(z)$ listed above have dimension $t - 1$. Since $s = t - 2$,

$\dim_B(K_{\alpha, \pi} \cap K_{\alpha, \pi}) = \dim_B(K_{\alpha, \pi, \alpha}) = t - 2$

therefore, by Lemma 4, we deduce that $u_{t-1} \notin K_{\alpha, \pi}$ and $u_t \notin K_{\alpha, \pi}$. Therefore, $U' = \{u_1, \ldots, u_t\}$ is a $B$-basis of $F$. Similarly, $V' = \{v_1, \ldots, v_t\}$ and $W' = \{w_1, \ldots, w_t\}$ are also $B$-bases of $F$. Thus, the six remaining checks can be defined as:

- $p_{t-1}(x) \equiv p_{u_{t-1}, \alpha}(x)$, $\text{Tr}(u_{t-1}(x - \alpha)/(x - \alpha))$,
- $p_t(\pi) \equiv p_{u_t, \pi}(x) = \text{Tr}(u_t(x - \alpha))/(x - \alpha)$,
- $q_{t-1}(x) \equiv p_{v_{t-1}, \pi}(x)$, $\text{Tr}(v_{t-1}(x - \pi)/(x - \pi))$,
- $q_t(\pi) \equiv p_{v_t, \pi}(x) = \text{Tr}(v_t(x - \pi))/(x - \pi)$,
- $r_{t-1}(x) \equiv p_{w_{t-1}, \alpha}(x)$, $\text{Tr}(w_{t-1}(x - \alpha)/(x - \alpha))$,
- $r_t(\pi) \equiv p_{w_t, \pi}(x) = \text{Tr}(w_t(x - \pi))/(x - \pi)$.

Lemma 10 concludes that these six checks satisfy the support patterns described in Table V.
TABLE V: (Case 1) The list of checks used in the Complement Phase when $s = t - 2$. A cross “×” denotes a non-zero value, while a dot “•” denotes a zero value.

| Checks | Erased Positions | $\alpha^*$ | $\alpha'$ | $\beta$ | Purpose |
|--------|------------------|------------|-----------|---------|---------|
| $p_{t-1}$ | × | × | × | → target trace for $f(\alpha^*)$ |
| $p_t$ | × | × | × | → target trace for $f(\alpha')$ |
| $q_{t-1}$ | × | × | × | → target trace for $f(\alpha^*)$ |
| $q_t$ | × | × | × | → target trace for $f(\alpha')$ |
| $r_{t-1}$ | × | × | × | → target trace for $f(\alpha^*)$ |
| $r_t$ | × | × | × | → target trace for $f(\alpha')$ |

**Lemma 10.** The following statements hold.
- $p_{t-1}(x)$ involves $f(\alpha^*)$ and $f(\alpha')$, but excludes $f(\beta)$.
- $p_t(x)$ involves $f(\alpha^*)$ and $f(\beta)$, but excludes $f(\alpha')$.
- $q_{t-1}(x)$ involves $f(\alpha^*)$ and $f(\alpha')$, but excludes $f(\beta)$.
- $q_t(x)$ involves $f(\alpha^*)$ and $f(\beta)$, but excludes $f(\alpha')$.
- $r_{t-1}(x)$ involves $f(\alpha')$ and $f(\alpha')$, but excludes $f(\alpha^*)$.
- $r_t(x)$ involves $f(\alpha^*)$ and $f(\alpha')$, but excludes $f(\beta)$.

**Proof.** We prove the first statement, which states that $p_{t-1}(\alpha^*) \neq 0$, $p_{t-1}(\alpha') \neq 0$, while $p_{t-1}(\beta) = 0$. The other statements can be derived in a similar manner. Indeed, Lemma 1 (b) implies that $p_{t-1}(\alpha^*) = u_{t-1} \neq 0$ and Lemma 3 (a) implies that $p_{t-1}(\beta) = 0$ since $u_{t-1} \in K_{\alpha^*,\beta}$. As $u_{t-1} \notin K_{\alpha',\alpha}$, we deduce that $p_{t-1}(\alpha') \neq 0$.

**Lemma 11 (Cycle Lemma I).** Suppose that all $3(n-3)$ repair traces downloaded and $3(t-2)$ target traces constructed in the Download Phase are given. Moreover, suppose that $t$ is divisible by $\text{char}(F)$. Then the following statements hold.

- If the target trace $\text{Tr}(r_{t-1}(\alpha')(f(\alpha')))$ is known, one can determine the target trace $\text{Tr}(p_{t-1}(\alpha^*)f(\alpha^*))$.
- If the target trace $\text{Tr}(p_{t-1}(\alpha^*)f(\alpha'))$ is known, one can determine the target trace $\text{Tr}(q_{t-1}(\alpha^*)f(\alpha'))$.
- If the target trace $\text{Tr}(q_{t-1}(\alpha)f(\beta))$ is known, one can determine the target trace $\text{Tr}(r_{t-1}(\alpha)f(\beta))$.

**Proof.** Due to symmetry, we only need to prove the first statement. Suppose that the target trace $\text{Tr}(r_{t-1}(\alpha')(f(\alpha')))$ of $f(\alpha')$ is known. According to Lemma 10, $p_{t-1}(\alpha) = 0$. Therefore,

$$\text{Tr}(p_{t-1}(\alpha^*)f(\alpha^*)) + \text{Tr}(p_{t-1}(\alpha')f(\alpha')) = -\sum_{\alpha \in \mathcal{A} \setminus \{\alpha^*,\alpha',\alpha\}} \text{Tr}(p_{t-1}(\alpha)f(\alpha)), \quad (20)$$

where the right-hand side sum can be computed from the repair traces obtained in the Download Phase. Our purpose is to extract the target trace $\text{Tr}(p_{t-1}(\alpha^*)f(\alpha^*))$ from this equation, and it suffices to show that we can determine the “interfering” trace $\text{Tr}(p_{t-1}(\alpha')f(\alpha'))$. Recall that $r_{t-1}(\alpha') = w_{t-1}$, and that $\{w_1, \ldots, w_{t-1}\}$ generates the root space of $Q_{\alpha',\alpha^*}(z)$. Since we already generated $\text{Tr}(w_if(\alpha'))$, for $i \in [t-1]$, by the linearity of the trace function, we only need to establish that $p_{t-1}(\alpha') = p_{w_{t-1},\alpha}(\alpha')$ is a root of the polynomial $Q_{\alpha',\alpha^*}(z)$. But this claim follows directly from Lemma 3 (b).

**Lemma 12 (Cycle Lemma II).** Suppose that all $3(n-3)$ repair traces downloaded and $3(t-2)$ target traces constructed in the Download Phase are given. Moreover, suppose that $t$ is divisible by $\text{char}(F)$. Then the following statements hold.

- If the target trace $\text{Tr}(q_{t}(\alpha)f(\beta))$ is known, one can determine the target trace $\text{Tr}(p_{t}(\alpha^*)f(\alpha^*))$.
- If the target trace $\text{Tr}(p_{t}(\alpha^*)f(\alpha'))$ is known, one can determine the target trace $\text{Tr}(r_{t}(\alpha')f(\beta))$.
- If the target trace $\text{Tr}(r_{t}(\alpha)f(\beta))$ is known, one can determine the target trace $\text{Tr}(q_{t}(\alpha)f(\beta))$.

**Proof.** The proof of this lemma proceeds along the same lines as the proof of Lemma 11.

To establish the validity of the procedure used in the Complement Phase, it remains to show that one can simultaneously “activate” the two cycles whenever one of the ratios of pairwise differences among $\alpha^*$, $\beta$, and $\alpha'$ belongs to the kernel $K$ of the trace function.

**Lemma 13 (Activation Lemma).** Suppose that all $3(n-3)$ repair traces downloaded and $3(t-2)$ target traces constructed in the Download Phase are given. Moreover, suppose that $t$ is divisible by $\text{char}(F)$. Then the following statements hold.
Lemma 10, and since which are obtained in the Download Phase. Note that can be written as a \(-x\)-linear combination of the \(q_{\alpha'}\)s is a root of both polynomials. The fact that \(q_{\alpha'}(\alpha') = p_{\alpha',\kappa}(\alpha')\) is a root of \(Q_{\alpha',\kappa}(\alpha')\) follows from Lemma 3 (b). To show that \(q_{\alpha'}(\alpha')\) is also a root of \(Q_{\alpha',\kappa}(\alpha')\), note that, since \(1 \in K\) (from the assumption that \(x\) is divisible by \(\text{char}(F)\)). Therefore, if we set \(b' = \text{Tr}(v_1(\alpha' - \overline{\alpha})) \in B\), then

\[
Q_{\alpha',\kappa}(\alpha') = \text{Tr}\left(\frac{b'}{\alpha' - \overline{\alpha}}(\alpha' - \overline{\alpha})\right) = b'Tr(\alpha') = 0.
\]

The proof follows.

Lemma 11, Lemma 12, and Lemma 13 complete the analysis of the case \(s = t - 2\) of the Complement Phase. At the end of the Complement Phase, the repair center has obtained \(t\) independent traces for each erased symbol and therefore, is capable of recovering all three lost symbols. It remains to consider the case when \(s = t - 1\).

Case 2: \(s = t - 1\). In this case, the roots spaces of the three polynomial \(Q_{\alpha',\kappa}(\alpha')\), \(Q_{\alpha',\kappa}(\alpha')\), and \(Q_{\alpha',\kappa}(\alpha')\) coincide. This happens, for example, when \(\overline{\alpha'} = \overline{\alpha'} - \alpha'\in B\). In such a case we get

\[
\ker(Q_{\alpha',\kappa}) = \frac{1}{\alpha' - \alpha'}K = \frac{1}{\alpha' - \alpha'}K = \ker(Q_{\alpha',\kappa}).
\]

In this case, for each erased symbol, as the repair center already obtained \(s = t - 1\) target traces in the Download Phase, one more target trace is needed for the recovery of the symbol.

We construct the last three checks \(p_s(x), q_s(x), \text{ and } r_s(x)\) via the field elements \(u_s, v_s, \text{ and } w_s\), chosen as follows:

- Choose \(u_s\) so that \(\text{rank}_B\{u_1, \ldots, u_t\} = t\).
- Choose \(v_s\) so that \(\text{rank}_B\{v_1, \ldots, v_t\} = t\).
- Choose \(w_s\) so that \(\text{rank}_B\{w_1, \ldots, w_t\} = t\).

We set

\[
p_s(x) \triangleq p_{u_s\alpha'}(x) = \text{Tr}(u_s(x) - \alpha')/(x - \alpha'),
\]

\[
q_s(x) \triangleq p_{v_s\alpha'}(x) = \text{Tr}(v_s(x) - \overline{\alpha'})/(x - \overline{\alpha'}),
\]

\[
r_s(x) \triangleq p_{w_s\alpha'}(x) = \text{Tr}(w_s(x) - \alpha')/(x - \alpha').
\]

Table VI summarizes the different types of checks and their uses in the repair process when \(s = t - 1\).

| Checks | Erased Positions | \(\alpha^*\) | \(\overline{\alpha'}\) | Purpose |
|--------|-----------------|---------------|-----------------|--------|
| \(p_s\) | \(x\)           | \(x\)          | \(x\)           | \(\rightarrow\) target trace for \(f(\alpha^*)\) |
| \(q_s\) | \(x\)           | \(x\)          | \(x\)           | \(\rightarrow\) target trace for \(f(\overline{\alpha'})\) |
| \(r_s\) | \(x\)           | \(x\)          | \(x\)           | \(\rightarrow\) target trace for \(f(\alpha')\) |

TABLE VI: (Case 2) The list of checks used in the Complement Phase when \(s = t - 1\). A cross \(\times\) denotes a nonzero value, while a dot \(\cdot\) denotes a zero value.

Each of these three checks involve all three erased symbols. Hence, in each repair equation, there are two “interfering” terms. However, based on Lemma 14, we can determine all of these terms from the repair and target traces obtained in the Download Phase. Consequently, the traces \(\text{Tr}(u_s(x) - \alpha')\), \(\text{Tr}(v_s(x) - \overline{\alpha'})\), and
Tr(\(w_i f(\alpha')\)), can also be computed. Note that no additional symbol download is required.

**Lemma 14.** Given \(3t-3\) target traces of \(f(\alpha^*), f(\pi), \) and \(f(\alpha')\), and \(3(n-3)\) repair traces obtained in the Download Phase, the target traces \(\text{Tr}(w_i f(\alpha^*))\), \(\text{Tr}(v_i f(\pi))\), and \(\text{Tr}(w_i f(\alpha'))\) can be determined as long as \(t\) is divisible by \(\text{char}(F)\).

**Proof.** By symmetry, it suffices to show that the target trace \(\text{Tr}(w_i f(\alpha^*))\) can be determined using the known traces. The repair equation generated by \(p_i(x)\) is given as follows.

\[
\text{Tr}(p_i(\alpha^*)f(\alpha^*)) + \text{Tr}(p_i(\pi)f(\pi)) + \text{Tr}(p_i(\alpha')f(\alpha')) = - \sum_{\alpha \in A \setminus \{\alpha^*, \pi, \alpha'\}} \text{Tr}(p_i(\alpha)f(\alpha)). \tag{22}
\]

Note that the right-hand side sum of this repair equation can be determined using the \(n-3\) repair traces \(\frac{f(\alpha')}{\alpha-\alpha'}\) retrieved in the Download Trace.

We now prove that the two “interfering” traces \(\text{Tr}(p_i(\pi)f(\pi))\) and \(\text{Tr}(p_i(\alpha')f(\alpha'))\) can be deduced from the known target traces \(f(\pi)\) and \(f(\alpha')\), respectively. Indeed, by Lemma 3(b), \(p_i(\pi) = p_{u_1, \alpha'}(\pi)\) belongs to the root space of \(Q_{\alpha', \pi}(z)\), which is generated by \(\{v_1, \ldots, v_{t-1}\}\). Therefore, the “interfering” trace \(\text{Tr}(p_i(\pi)f(\pi))\) is a \(B\)-linear combination of the known target traces \(\text{Tr}(v_i f(\pi))\), \(i \in [t-1]\). Similarly, \(p_i(\alpha') = p_{u_1, \alpha'}(\alpha')\) belongs to the root space of \(Q_{\alpha', \alpha'}(z)\), which is generated by \(\{w_1, \ldots, w_{t-1}\}\). The “interfering” trace \(\text{Tr}(p_i(\alpha')f(\alpha'))\) is a \(B\)-linear combination of the known target traces \(\text{Tr}(w_i f(\alpha'))\), \(i \in [t-1]\). This completes the proof. \(\blacksquare\)

**B. A Distributed Three-Erasure Repair Scheme for Reed-Solomon Codes**

We can easily modify the centralized repair scheme proposed in Section IV-A to obtain a distributed scheme. The following theorem is the distributed version of Theorem 4.

**Theorem 5.** Suppose \(f(x) \in F[x]\) is a polynomial of degree at most \(k-1\), which corresponds to a codeword in a Reed-Solomon code \(C = RS(A, k)\). Assume that \(f(\alpha^*), f(\pi), \) and \(f(\alpha')\) are the three erased codeword symbols. Given that the field extension degree \(t\) is divisible by the characteristic of \(F\), and that

\[
\left\{ \frac{\pi - \alpha^*}{\alpha^* - \alpha}, \frac{\pi - \alpha}{\alpha - \alpha'}, \frac{\alpha^* - \alpha'}{\alpha - \alpha}, \frac{\pi - \alpha}{\alpha - \alpha}, \frac{\alpha^* - \alpha'}{\alpha - \alpha} \right\} \cap \mathbb{K} \neq \emptyset
\]

holds, where \(K = \ker(\text{Tr})\) is the kernel of the trace function, there exists a distributed repair scheme where each lost symbol is recovered with a repair bandwidth of \(n-1\) sub-symbols.

**Remark 3.** In a naive scheme, one RN first downloads \(kt\) sub-symbols from a set of \(k\) available nodes, recovers all three erased symbols, and then sends the corresponding symbols to the other two RNs. The total bandwidth used is \(kt+2t\), which is worse than the scheme we proposed if \(\frac{k+2}{3} > \frac{2}{t}\), i.e. when \(t\) is sufficiently large or when the code has high rate.

We use the same notations as in Section IV-A The repair process also consists of two phases.

**Download Phase.**

- The RN for \(f(\alpha^*)\) downloads \(n-3\) repair traces \(\text{Tr}(\frac{f(\alpha)}{\alpha-\alpha^*})\) from \(n-3\) available nodes, where \(\alpha \in A \setminus \{\alpha^*, \pi, \alpha'\}\), and recovers \(s\) independent target traces of \(f(\alpha^*)\), i.e. \(\text{Tr}(u_i f(\alpha^*))\), \(i \in [s]\).

- The RN for \(f(\pi)\) downloads \(n-3\) repair traces \(\text{Tr}(\frac{f(\alpha)}{\alpha-\alpha^*})\) from \(n-3\) available nodes, where \(\alpha \in A \setminus \{\alpha^*, \pi, \alpha'\}\), and recovers \(s\) independent target traces of \(f(\pi)\), i.e. \(\text{Tr}(v_i f(\pi))\), \(i \in [s]\).

- The RN for \(f(\alpha')\) downloads \(n-3\) repair traces \(\text{Tr}(\frac{f(\alpha)}{\alpha-\alpha^*})\) from \(n-3\) available nodes, where \(\alpha \in A \setminus \{\alpha^*, \pi, \alpha'\}\), and recovers \(s\) independent target traces of \(f(\alpha')\), i.e. \(\text{Tr}(w_i f(\alpha'))\), \(i \in [s]\).

**Collaboration Phase.**

We describe the tasks the RN for \(f(\alpha^*)\) performs in this phase. Similar descriptions apply to the RNs for \(f(\pi)\) and \(f(\alpha')\). We consider two cases, depending on whether \(s = t-2\) or \(s = t-1\).

**Case 1:** \(s = t-2\). The RN for \(f(\alpha^*)\) computes the last two target traces \(\text{Tr}(u_{t-1} f(\alpha^*))\) and \(\text{Tr}(u_t f(\alpha^*))\) as follows. The first trace can be obtained by using the repair equation \(\tag{22}\) generated by \(p_{t-1}\), in which the only unknown term \(\text{Tr}(p_{t-1}(\alpha')f(\alpha'))\) can be determined by downloading an additional repair trace \(\frac{f(\alpha)}{\alpha-\alpha^*}\) from the RN for \(f(\alpha')\). The availability of this repair trace at the RN for \(f(\alpha^*)\) is explained by either Cycle Lemma I (Lemma 11) or Activation Lemma (Lemma 3). A similar assertion holds for the second trace. The RN for \(f(\alpha^*)\) now has \(t\) independent traces to recover this lost symbol. It has downloaded in total \(n-1 = (n-3) + 2\) sub-symbols (repair traces) from all other \(n-1\) nodes. We illustrate the flow of the recovery data among the RNs in Fig. 6 when \(\frac{\pi-\alpha}{\alpha-\alpha^*} \in K\). Essentially, if an RN can determine a new target trace, it can also compute the repair trace needed for the next RN along the cycle, and so forth.

**Fig. 6:** Illustration of the data flow in the Collaboration Phase when \(s = t-2\) and \(\frac{\pi-\alpha}{\alpha-\alpha^*} \in K\). The two repair cycles are first activated at the replacement nodes for \(f(\pi)\) and \(f(\alpha')\), which can compute new target traces from their previously obtained traces, thanks to the Activation Lemma. These two nodes then are able to compute the repair traces to send to the next RN along the two cycles. This process is repeated until every node is able to recover the two new target traces. The outer arrows represent the Repair Cycle I, while the inner arrows represent the Repair Cycle II. The two cycles operate independently in parallel.

**Case 2:** \(s = t-1\). The RN for \(f(\alpha^*)\) can compute the last target trace \(\text{Tr}(u_t f(\alpha^*))\) based on the repair equation \(\tag{22}\) generated by \(p_t\). The only unknown terms are \(\text{Tr}(p_t(\pi)f(\pi))\) and
Tr(p_2(α)f(α')). Accordingly to the proof of Lemma 14 these two terms can be determined by downloading two additional sub-symbols (repair traces) Tr(\(f(\tau)\pi^{-1}\alpha^{-1}\)) and Tr(\(f(\tau')\alpha^{-1}\)) from the RNs for \(f(\tau)\) and \(f(\tau')\), respectively. Thus, the total number of downloaded sub-symbols is \(n - 1 = (n - 3) + 2\).

C. Correctable Patterns of Three Erasures

We next evaluate the number of three-erasure patterns that are correctable at low costs by our centralized and distributed repair schemes developed in the previous sections.

Under the assumption that the field extension degree \(t\) is divisible by the characteristic of the fields \(B\) and \(F\), according to Theorem 4, the erased codeword symbols corresponding to three evaluation points \(α, β, \gamma\), that satisfy

\[
\left\{ \frac{β - α}{β - γ}, \frac{γ - β}{γ - α}, \frac{α - γ}{α - β} \right\} \cap K = \emptyset,
\]

(23)
can always be recovered by downloading \(3(n - 3)\) sub-symbols from the \(n - 3\) unerased codeword symbols. Fix two arbitrary evaluation points, say \(α\) and \(β\). We would like to know how many choices there are for the third point \(γ\) so that the triple of points satisfies (23). For two fixed distinct elements \(α\) and \(β\), and for each \(κ\in K\setminus\{0,1\}\), we can define

\[
γ = β - \frac{β - α}{κ}.
\]

It is straightforward to verify that \(γ \neq α, γ \neq β\), and that \(\frac{β - α}{κ - γ} \in K\). Therefore, there are \(|K| - 2 = \frac{n}{|B|} - 2\) choices for \(γ\) that ensure that \(\frac{β - α}{κ - γ} \in K\). The same assertion holds if we were to require \(\frac{γ - β}{γ - α} \in K\) or \(\frac{α - γ}{α - β} \in K\). Hence, the number of correctable triples is at least a fraction of \(\frac{1}{|B|}\) of the number of all triples.

A computer-aided count of the number of correctable triples, two of which are fixed, is given in Table VII for some small values of \(|B|\) and \(t\).

| \(|B|\) = 2, \(t = 4\) | \(|B|\) = 2, \(t = 6\) | \(|B|\) = 2, \(t = 8\) | \(|B|\) = 2, \(t = 10\) | \(|B|\) = 4, \(t = 4\) | \(|B|\) = 4, \(t = 6\) | \(|B|\) = 4, \(t = 8\) | \(|B|\) = 8, \(t = 4\) | \(|B|\) = 8, \(t = 6\) | \(|B|\) = 3, \(t = 3\) | \(|B|\) = 3, \(t = 9\) | \(|B|\) = 9, \(t = 3\) | \(|B|\) = 9, \(t = 9\) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #Correctable triples | #All triples = \(n - 2\) | #Correctable triples | #All triples = \(n - 2\) | #Correctable triples | #All triples = \(n - 2\) | #Correctable triples | #All triples = \(n - 2\) | #Correctable triples | #All triples = \(n - 2\) | #Correctable triples | #All triples = \(n - 2\) | #Correctable triples | #All triples = \(n - 2\) |
| 14 | 14 | 62 | 62 | 206 | 206 | 900 | 902 | 158 | 254 | 2,330 | 4,094 | 37,886 | 65,534 |
| 1,406 | 4,094 | 86,694 | 262,142 | 19 | 25 | 529 | 721 | 11,068 | 19,684 | 223 | 727 | 158,263 | 531,439 |

TABLE VII: Fixing two erased locations, the second column counts the number of the third erased location that result in correctable patterns of three erasures. The third column specifies the number of all possible choices for the third location once the first two are fixed, which is \(n - 2\).

V. CONCLUSIONS

A. Open Problems

We proposed centralized and distributed repair schemes for the recovery of two and three erasures/failures in Reed-Solomon codes. The distributed schemes offer the same repair bandwidths per erasure as in the case of a single erasure. Several open questions remain, including

(P1) Finding distributed repair schemes that can recover all three-erasure patterns with the repair bandwidths of \(n - 1\) sub-symbols (per erased symbol) and, in addition, when \(t\) is not divisible by \(\text{char}(F)\).

(P2) Establishing a lower bound on the repair bandwidth and developing repair schemes for an arbitrary number of erasures that meet the bound. We note that in a follow-up work, Bartan and Wootters [26] extended our centralized repair scheme to tackle up to \(n - k\) erasures. They were also able to improve the repair bandwidth of the scheme based on the observation that the downloaded data in our current scheme is often redundant. Such an extension in the distributed setting remains unknown.

(P3) Developing efficient repair schemes for codes with an arbitrary number of parities, with particular emphasis on those used in practice, which usually have short lengths, small redundancy, and \(t = 8, 16, 32\), which is closer to \(n\) than \(\log(n)\) as considered in the current setting. We remark that Duursma and Dau [27] recently obtained a reduction of 32.5% in repair bandwidth of RS(14,10) code that is used in Facebook F4 Storage System, which improved upon previous reductions of 18.75% in [28] and 20% in [10], for single erasures. It is of great practical interest to investigate the possibility of repairing such codes (see Table I) more efficiently in the presence of multiple erasures.

(P4) In the large subpacketization regime, Ye and Barg [29] extended the work of Tamo et al. [30] to construct RS codes that adopt centralized repair schemes with bandwidths achieving the cut-set bound ([2], [3]) for any number of erasures. Investigating distributed repair schemes for RS codes in that regime is another open problem for future research.

B. Related Works and Performance Comparison

Apart from those recent works on repairing RS codes mentioned in Section V-A, there is also a number of related results in the literature that extend minimum storage regenerating (MSR) codes [2], [3], originally designed to recover single erasures only, to cope with multiple erasures. In such settings, one often fixes \(n, k\), and a finite field \(B'\), and constructs an MDS code over a larger field \(F' \cong B'^t\). The field extension degree \(s\) is referred to as the subpacketization level. In most of these works, \(s\) is often larger than or equal to \(n - k\). By contrast, in our work, we fix \(n, k, F\), and choose a subfield \(B\) of \(F\) so that \(|F| = |B|^t\) and \(t \leq \log_{|B|} (n - k) \leq n - k\). By slightly adapting the notation of [31], we may define the rebuilding ratio as the fraction of data (originally) stored at each helper node that is sent to a RN, or to the repair center, in order to reconstruct the content of each failed node. Most of the codes known in the literature have a rebuilding ratio of \(1/(n - k)\), while Reed-Solomon codes with our repair schemes have a rebuilding ratio of \(1/t \geq 1/\log_{|B|} (n - k) + 1\). MDS codes with larger subpacketization levels allow for smaller rebuilding ratios at the cost of increasing encoding/decoding as well as code description complexities.

We briefly review these results and compare them with the results described in the previous sections.
The first line of work is concerned with the problem of repairing $e$ erasures ($e \geq 1$) in MDS codes in a distributed manner. The distributed repair process is often referred to as cooperative or collaborative repair. A cooperative repair scheme generally consists of two phases. In the first phase, the RNs contact and download recovery data from the available nodes. In the second phase, they exchange information in order to help each other to complete their repair processes. Our distributed repair schemes may be viewed as an adaptation of this general strategy to the trace-repair framework [10].

Hu et al. [32] presented a lower bound on the total repair bandwidth, which equals $\frac{(n - 1)\Delta}{n - k}$ sub-symbols in a subfield $B'$, where $s$ is the subpacketization level. Wang et al. [33] generalized this bound to $\sum_{t=1}^{d} \frac{d}{d_i - k + 1}$, by allowing each RN for $f(\alpha_i)$ to connect to $d_i$ nodes (including both available and RNs), where $d_i \leq n - 1$. MDS codes with repair bandwidths attaining this bound are referred to as minimum-storage cooperative regenerating (MSCR) codes. The authors of [32], [33] also provided probabilistic constructions of such MSCR codes with $s = n - k$. These codes ensure functional instead of exact repair, i.e. there is no guarantee that the repaired content of a failed node is the same as the original. When $d_i = n - 1$ for every $i \in [n]$, these MSCR codes have a rebuilding ratio of $1\ n$. Our distributed repair schemes for Reed-Solomon codes with $e = 2, 3$ (see Remark 1) achieve a building ratio $\frac{1}{d} \geq \frac{\log n (n-k)+1}{n-k}$. Thus, their random codes have a lower rebuilding ratio than the Reed-Solomon codes under our repair schemes, while employing a higher level of subpacketization. Le Scouarnec [34] proposed another construction of MSCR codes with exact repair for $e = 2$ erasures when $k = 2$. This result was later extended to cover all $k \geq 3$ and $n = 2k$ in the work of Chen and Shum [35]. Exact-repair MSCR codes also exist for $d = k$ and $e \leq n - d$, with $s = n - k$, as shown by Shum and Hu [36]. Li and Li [37] proved the interesting fact that any MSR code (see [2], [3]) is also an MSCR code for $e = 2$ erasures. Therefore, an MSCR code for $e = 2$ exists whenever an MSR code with the same parameters exists. The most recent work along this line is the paper by Shum et al. [38], which introduced a repair scheme that can recover any number of systematic node failures in the MISER code [39], for which $n = 2k$. This scheme is also bandwidth-optimal, and uses a level of subpacketization equal to $n - k$.

The problem of centralized repair of multiple erasures for MDS codes has also been studied in the literature. Cadambe et al. [40] proved that in order to repair $e$ erasures, the total repair bandwidth has to be at least $\frac{e(d + e)}{d_i - e + 1}$ sub-symbols from a subfield $B'$, where $d \leq n - e$ is the number of available nodes that the repair center contacts. MDS codes with repair bandwidths attaining this bound are referred to as minimum-storage multi-node regenerating (MSMR) codes. They also showed the existence of asymptotic MSMR codes with total repair bandwidths approaching the lower bound when the subpacketization level $s \to \infty$, given that the subfield $B'$ is fixed. When $d = n - e$, Tamo et al. [41] demonstrated that their Zigzag code is an exact-repair MSMR code if restricted to $e$ systematic node erasures, with subpacketization level $s = (n-k)^{k-1}$. Recently, this result was extended by Wang et al. [42] to cover all $e$-erasure patterns. Note that when $d = n - e$, the rebuilding ratio of an MSMR code is $\frac{1}{d} \geq \frac{1}{\log |B'| (n-k)+1}$ for our centralized repair schemes for Reed-Solomon codes with $e = 2, 3$. However, their subpacketization level $s = (n-k)^{k-1}$ is also much larger than ours, which equals $t \leq \log |B'| (n-k) + 1$. The centralized repair problem was also investigated in the context of wireless distributed storage systems, where storage nodes are fully connected by a common broadcast channel [43].

Ye and Barg [44, 45] introduced a new notion of MDS codes with the universally error-resilient $(e, d)$-optimal repair property. These codes are MSMR codes for all $e \leq n - k$ and all $d \leq n - e$ simultaneously. The authors of [44, 45] also provided a construction of such codes whenever the subfield $B'$ has size $|B'| \geq sn$, where $s = \lcm(1, 2, \ldots, n - k)$, with subpacketization level $t = s^n$.

ACKNOWLEDGMENT

This work has been supported in part by the NSF grant CCF 1526875, the Center for Science of Information under the grant NSF 0939370, and the NSF grant CCF 1619189. H. M. Kiah was also supported in part by the Singapore Ministry of Education under Research Grants MOE2016-T1-001-156 and MOE2015-T2-2-086.

REFERENCES

[1] H. Dau, I. Duursma, H. M. Kiah, and O. Milenkovic, “Repairing Reed-Solomon codes with two erasures,” in Proc. IEEE Int. Symp. Inform. Theory (ISIT), 2017, pp. 351–355.
[2] A. Dimakis, P. Godfrey, M. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” in Proc. 26th IEEE Int. Conf. Comput. Commun. (INFOCOM), 2007, pp. 2000–2008.
[3] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” IEEE Trans. Inform. Theory, vol. 56, no. 9, pp. 4539–4551, 2010.
[4] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” J. Soc. Ind. Appl. Math., vol. 8, no. 2, pp. 300–304, 1960.
[5] F. J. MacWilliams and N. J. A. Sloane, The Theory of Error-Correcting Codes. Amsterdam: North-Holland, 1977.
[6] A. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, “A survey on network codes for distributed storage,” Proc. IEEE, vol. 99, no. 3, pp. 476–489, 2011.
[7] F. Oggier and A. Datta, “Self-repairing homogeneous codes for distributed storage systems,” in Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM), 2011, pp. 1215–1223.
[8] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, “On the locality of codeword symbols,” IEEE Trans. Inform. Theory, vol. 58, no. 11, pp. 6925–6934, 2012.
[9] D. Papailiopoulos and A. Dimakis, “Locally repairable codes,” in IEEE Int. Symp. Inform. Theory (ISIT), 2012, pp. 2771–2775.
[10] V. Guruswami and M. Wootters, “Repairing Reed-Solomon codes,” in Proc. Annu. Symp. Theory Comput. (STOC), 2016.
[11] A. Fikes, Colossus, Successor to Google File System, http://static.googleusercontent.com/media/research.google.com/en/us/university/research/facultysummit2010/storage_architecture_and_challenges.pdf
[12] M. Ovsianikov, S. Rus, D. Reeves, P. Sutter, S. Rao, and J. Kelly, “The Quantcast File System,” in Proc. VLDB Endow., vol. 6, no. 11, 2013, pp. 1092–1101.
[13] R. Li, Z. Zhang, K. Zheng, and A. Wang, “Progress report: Bringing erasure coding to Apache Hadoop,” http://blog.cloudera.com/blog/2016/02/progress-report-bringing-erasure-coding-to-apache-hadoop/
[14] P. Narayanan, S. Samal, and S. Nanniyur, “Yahoo cloud object store - object storage at exabyte scale,” https://yahooeng.tumblr.com/post/116391291701/yahoo-cloud-object-store-object-storage-at
[15] B. Beach, “Backblaze Vaults: Zettabyte-scale cloud storage architecture,” https://www.backblaze.com/blog/vault-cloud-storage-architecture/
[16] S. M. et al., “FC Facebook’s warm BLOB storage system,” in Proc. 11th ACM/USENIX Symp. Oper. Syst. Des. Implementation (OSDI), 2014, pp. 383–398.
[17] C. Lai, S. Jiang, L. Yang, S. Lin, G. Sun, Z. Hou, C. Cui, and J. Cong, “Atlas: Baidu’s key-value storage system for cloud data,” in Proc. 31st Int. Mass Stor. Syst. Tech.(MSST), 2015, pp. 1–14.
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 Conf. Hot Topics Storage File Syst. (HotStorage), 2016, pp. 8–8.

K. V. Rashmi, N. B. S. D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, “A ‘Hitchhiker’s’ guide to fast and efficient data reconstruction in erasure-coded data centers,” in Proc. 2014 ACM Conf. SIGCOMM, 2014, pp. 331–342.

R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, “Total Recall: System support for automated availability management,” in Proc. 1st ACMUSENIX Symp. Networked Syst. Des. Implementation (NSDI), 2004, pp. 25–25.

M. G. Luby, R. Padovani, T. J. Richardson, L. Minder, and P. Aggarwal, “Liquid cloud storage,” 2017, available at https://arxiv.org/abs/1705.07983

F. Giroire, J. Monteiro, and S. Pérennes, “Peer-to-peer storage systems: A practical guideline to be lazy,” in IEEE Glob. Commun. Conf. (GLOBECOM), 2010.

M. Silberstein, L. Ganesh, Y. Wang, L. Alvisi, and M. Dahlin, “Lazy means smart: Reducing repair bandwidth costs in erasure-coded distributed storage,” in Proc. Int. Conf. Syst. Storage (SYSTOR), 2014, pp. 15:1–15:7.

R. Lidl and H. Niederreiter, Introduction to Finite Fields and Their Applications. Cambridge University Press, 1986.

H. Dau and O. Milenkovic, “Optimal repair schemes for some families of Reed-Solomon codes,” in Proc. IEEE Int. Symp. Theory (ISIT), 2017, pp. 346–350.

B. Bartan and M. Wootters, “Repairing multiple failures for scalar MDS codes,” in Proc. 55th Annual Allerton Conf. Comm Control Comput. (Allerton), 2017.

I. Duursma and H. Dau, “Low bandwidth repair of the RS(10,4) Reed-Solomon code,” in Information Theory and Application Workshop, 2017.

K. Shum, Y. Xu, D. S. Papailiopoulos, A. G. Dimakis, and G. Caire, “A repair framework for scalar MDS codes,” IEEE J. Selected Areas Comm. (JSAC), vol. 32, no. 5, pp. 998–1007, 2014.

M. Ye and A. Barg, “Repairing Reed-Solomon codes: Universally achieving the cut-set bound for any number of erasures,” 2017, available at https://arxiv.org/abs/1710.07216.

I. Tamo, M. Ye, and A. Barg, “Optimal repair of Reed-Solomon codes: Achieving the cut-set bound,” in Proc. 58th Annual IEEE Symp. Foundations Computer Sci. (FOCS), 2017.

I. Tamo, Z. Wang, and J. Bruck, “MDS array codes with optimal rebuilding,” in IEEE Int. Symp. Theory (ISIT), 2011, pp. 1240–1244.

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 Comm. (JSAC), vol. 28, no. 2, pp. 268–276, 2010.

X. Wang, Y. Xu, Y. Hu, and K. Ou, “MFR: Multi-loss flexible recovery in distributed storage systems,” in IEEE Int. Conf. Commun. (ICC), 2010, pp. 1–5.

N. L. Scouarneec, “Exact scalar minimum storage coordinated regenerating codes,” in IEEE Int. Symp. Theory (ISIT), 2012, pp. 1197–1201.

J. Chen and K. W. Shum, “Repairing multiple failures in the Sub-Ramchandran regenerating codes,” in IEEE Int. Symp. Theory (ISIT), 2013, pp. 1441–1445.

K. W. Shum and Y. Hu, “Cooperative regenerating codes,” IEEE Trans. Inform. Theory, vol. 59, no. 11, pp. 7229–7258, 2013.

J. Li and B. Li, “Cooperative repair with minimum-storage regenerating codes for distributed storage,” in Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM), 2014, pp. 316–324.

K. Shum and J. Chen, “Cooperative repair of multiple node failures in distributed storage systems,” Inderscience, Int. J. Inform. Coding Theory, vol. 4, no. 3, 2016.

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. Inform. Theory, vol. 58, no. 4, pp. 2134–2158, 2013.

V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh, “Asymptotic interference alignment for optimal repair of MDS codes in distributed storage,” IEEE Trans. Inform. Theory, vol. 59, no. 5, pp. 2974–2987, 2013.

I. Tamo, Z. Wang, and J. Bruck, “Zigzag codes: MDS array codes with optimal rebuilding,” IEEE Trans. Inform. Theory, vol. 59, no. 3, pp. 1597–1616, 2013.

Z. Wang, I. Tamo, and J. Bruck, “Optimal rebuilding of multiple erasures in MDS codes,” IEEE Trans. Inform. Theory, vol. 63, no. 2, pp. 1084–1101, 2016.

P. Hu, C. W. Sung, and T. H. Chan, “Broadcast repair for wireless distributed storage systems,” in Int. Conf. Inform. Commun. Signal Processing (ICICS), 2015, pp. 1–5.