Repairing Reed-Solomon Codes over $GF(2^\ell)$

Weiqi Li, Zhiying Wang, and Hamid Jafarkhani

***Abstract*—We provide a detailed algorithm to repair a single node failure in an (n, k) Reed-Solomon code over $GF(2^\ell)$ with repair bandwidth $\frac{\ell}{a}(n-1)(a-s)$, for any integers a, s such that $a|\ell, 2^a \geq n+1, 2^s \leq n-k$. We present the constructions of necessary lookup tables for the repair. The storage overhead and the repair complexity of our algorithm are also analyzed. The algorithm can be applied to the $RS(14, 10)$ over $GF(2^8)$, which is a modification of the code in Facebook's f4 system, and reaches the lowest repair bandwidth among the existing schemes to the best of our knowledge. The algorithm can be also generalized to other codes, including the ones based on Yahoo Object Store and Baidu's Atlas.**

I. Introduction

Reed-Solomon (RS) codes are very popular in distributed systems because they provide efficient implementation and high failure-correction capability for a given redundancy level [1], [2]. They are widely used in practical systems such as Google's Colossus, Quantcast File System, Facebook's f4, Yahoo Object Store, Baidu's Atlas, Backblaze's Vaults, and Hadoop Distributed File System [3, Tab. I]. We define the repair bandwidth, b, as the required amount of transmission from the helpers (the remaining nodes) for a single node (codeword symbol) erasure, maximized over all single failures. Different repair schemes for reducing the repair bandwidth of RS codes have been proposed in the literature [4]–[7].

Let us consider the Galois field $\mathbb{F} = GF(q^\ell)$, where q is a prime number and ℓ is called the sub-packetization size. We denote a Reed-Solomon code with code length n and dimension k over $\mathbb{F}$ as $RS(n, k)$. The naive repair scheme requires the repair bandwidth of $b = k\ell$ symbols of the base field $\mathbb{B} = GF(q)$ to repair a single failure. To reduce the repair bandwidth, the symbols in the original field $\mathbb{F}$ can be mapped to sub-symbols in a smaller subfield using the *trace function*. The sub-symbols are downloaded from the helpers to repair the failure. The repair bandwidth is reduced when the required sub-symbols from the same helper are linearly dependent. For example, the full-length codes [4], [5] achieve the repair bandwidth of $(n-1)(l-s)$, where $s \leq \log_q(n-k)$. They are optimal when $n = q^\ell$ but do not perform well when $n \ll q^\ell$. The MSR (minimum storage regenerating) scheme [6] and the asymptotic MSR scheme [7] achieve the MSR lower bound $b \geq \frac{\ell(n-1)}{n-k}$ [8], using a super-exponential sub-packetization size ℓ. Constructions in [9] provide a tradeoff between the sub-packetization size and the repair bandwidth. The work in [10] provides schemes for Facebook's f4 [11] with repair bandwidth of 54 bits.

In this letter, we provide a detailed algorithm and the necessary lookup tables to implement the general ideas in [9] to repair $RS(n, k)$ over $GF(2^\ell)$. The computation complexity is analyzed in terms of the number of required finite field operations and required memories. Moreover, for $\ell = 8, 16, 32, 64$, which are frequently used sub-packetization sizes in practice, we provide an efficient computation of the trace function. **Notation:** We use $[a]$ to represent $\{1, 2, \ldots, a\}$ for a positive integer a.

II. Repair Algorithm for $RS(n, k)$ Code

First, we describe the repair algorithm using the example of the $RS(14, 10)$ code. The formulas for the general case will be provided at the end.

The $RS(14, 10)$ code has $n = 14$ codeword symbols and $k = 10$ information symbols. In a storage system, different symbols are stored in different nodes. The symbols are over the finite field $\mathbb{F} = GF(2^8) = \{0, 1, \beta, \beta^2, ..., \beta^{254}\}$, where β is the primitive element of $\mathbb{F}$. As a standard choice [12], β is the root of $1+x^2+x^3+x^4+x^8 = 0$. For the information symbols $u_j \in \mathbb{F}, j = 0, 1, \cdots, 9$, let f be the polynomial $f(x) = \sum_{j=0}^{9} u_j x^j$. RS codeword symbols are evaluations of the polynomial f and erasures can be corrected using interpolation. In this section, we apply the general construction in [9] to this specific code. By restricting the evaluation points to a subfield, one obtains a low repair bandwidth related to the field size. Consider the subfield $\mathbb{E} = GF(2^4) = \{0, 1, \beta^{17}, \beta^{17\cdot 2}, ..., \beta^{17\cdot 14}\}$ of $\mathbb{F}$. Let us choose the set of evaluation points from $\mathbb{E}$, and denote it by $A = \{\alpha_1, \alpha_2, ..., \alpha_{14}\} = \{1, \beta^{17}, \beta^{17\cdot 2}, \beta^{17\cdot 3}, ..., \beta^{17\cdot 13}\}$. Then, the 14 codeword symbols, in the 14 nodes, are $\{N_m = f(\alpha_m) = \sum_{j=0}^{9} u_j {\alpha_m}^j : \alpha_m \in A\}$.

We use the *trace function* to map symbols of $\mathbb{F}$ to sub-symbols of the base field $\mathbb{B} = GF(2)$:

$$tr_{\mathbb{F}/\mathbb{B}}(x) = x + x^2 + x^{2^2} + x^{2^3} + ... + x^{2^7}, \qquad (1)$$

where for every $x \in \mathbb{F}$, $tr_{\mathbb{F}/\mathbb{B}}(x) \in \mathbb{B}$, i.e., the result of Eq. (1) is 0 or 1. Any symbol $x \in \mathbb{F}$ can be reconstructed from the 8 bits evaluated using the trace function as:

$$x = \sum_{i=1}^{8} \gamma_i' tr_{\mathbb{F}/\mathbb{B}}(\gamma_i x) = \sum_{i=1}^{8} \gamma_i' b_i', \qquad (2)$$

where $\{\gamma_1, \gamma_2, ..., \gamma_8\}$ is a basis for $\mathbb{F}$ over $\mathbb{B}$ and $\{\gamma_1', \gamma_2', ..., \gamma_8'\}$ is its dual basis.

TABLE I
Dual basis table for $RS(14,10)$ over $GF(2^8)$, $a=4, s=2$. β is a root of the primitive polynomial $1+x^2+x^3+x^4+x^8$.

Failed node	1	2	3	4	5	6	7	8	9	10	11	12	13	14
γ_1'	β^{203}	β^{118}	β^{254}	β^{203}	β^{33}	β^{220}	β^{16}	β^{118}	β^{16}	β^{33}	β^{152}	β^{152}	β^{220}	β^{254}
γ_2'	β^{152}	β^{67}	β^{203}	β^{152}	β^{237}	β^{169}	β^{220}	β^{67}	β^{220}	β^{237}	β^{101}	β^{101}	β^{169}	β^{203}
γ_3'	β^{84}	β^{254}	β^{135}	β^{84}	β^{169}	β^{101}	β^{152}	β^{254}	β^{152}	β^{169}	β^{33}	β^{33}	β^{101}	β^{135}
γ_4'	β^{16}	β^{186}	β^{67}	β^{16}	β^{101}	β^{33}	β^{84}	β^{186}	β^{84}	β^{101}	β^{220}	β^{220}	β^{33}	β^{67}
γ_5'	β^{187}	β^{102}	β^{238}	β^{187}	β^{17}	β^{204}	1	β^{102}	1	β^{17}	β^{136}	β^{136}	β^{204}	β^{238}
γ_6'	β^{136}	β^{51}	β^{187}	β^{136}	β^{221}	β^{153}	β^{204}	β^{51}	β^{204}	β^{221}	β^{85}	β^{85}	β^{153}	β^{187}
γ_7'	β^{68}	β^{238}	β^{119}	β^{68}	β^{153}	β^{85}	β^{136}	β^{238}	β^{136}	β^{153}	β^{17}	β^{17}	β^{85}	β^{119}
γ_8'	1	β^{170}	β^{51}	1	β^{85}	β^{17}	β^{68}	β^{170}	β^{68}	β^{85}	β^{204}	β^{204}	β^{17}	β^{51}

Let us assume the node $N_* = f(\alpha_*)$ fails, $1 \le * \le 14$. Then, the transmitted symbols from the other nodes (called helpers) are related to the failed node symbol through the dual codewords $\{c_{mi} : m \in [14]\}$ of $RS(14,10)$, for $i \in [8]$:

$$c_{*i}N_* = \sum_{m \neq *} c_{mi}N_m. \tag{3}$$

8 dual codewords are required for each failure. To transmit symbols of the base field $\mathbb{B}$, we take the trace function on both sides of (3). It is required that the 8 symbols $\{c_{*i} : i \in [8]\}$ form a basis for $\mathbb{F}$ over $\mathbb{B}$, then we can treat them as $\{\gamma_i : i \in [8]\}$ and use (2) to repair the failed node.

The dual codeword symbols are chosen appropriately in [9, Theorem 1] as

$$\{c_{mi} : i \in [8]\} = \{\upsilon_m \eta_t p_{j,*}(\alpha_m) : t \in [2], j \in [4]\}, \tag{4}$$

where $m \in [14]$ and the subscript i is indexed by t and j as $i = 4 \times (t-1) + j$. Here $\{\upsilon_m, m \in [14]\} = \{\beta^{51}, \beta^{136}, 1, \beta^{51}, \beta^{221}, \beta^{34}, \beta^{238}, \beta^{136}, \beta^{238}, \beta^{221}, \beta^{102}, \beta^{102}, \beta^{34}, 1\}$ are the column multipliers, $\{\eta_1, \eta_2\} = \{1, \beta\}$, and the polynomial $p_{j,*}(x)$ in [9, Theorem 1] is

$$p_{j,*}(x) = \xi_j \prod_{w \in W} (x - \alpha_* + w^{-1}\xi_j), \tag{5}$$

where $\xi_j = \beta^{17(j-1)}, W = \{1, \beta^{17}, \beta^{68}\}$.

Let $rank_{\mathbb{B}}(\{a_1, a_2, ..., a_i\})$ be the cardinality of a maximal subset of $\{a_1, a_2, ..., a_i\}$ that is linearly independent over $\mathbb{B}$. As analyzed in [9, Theorem 1], we have

$$rank_{\mathbb{B}}(\{c_{mi} : i \in [8]\}) = \begin{cases} 8, & \text{if } m = *, \\ 4, & \text{if } m \neq *, \end{cases} \tag{6}$$

and $\{c_{mi} : i \in [8]\}, m \neq *$ lie on two subspaces of dimension 2 spanned by $\{\upsilon_m \eta_t p_{j,*}(\alpha_m) : j \in [4]\}, t \in [2]$. Now, we have $\{c_{*i} : i \in [8]\}$ form a basis for $\mathbb{F}$ over $\mathbb{B}$, and we can recover the failed node with only 4 bits from each helper. Specifically, for helper m, we represent the basis for the subspace spanned by $\{c_{mi} : i \in [8]\}$ as $\{\epsilon_{m,t,z} : t \in [2], z \in [2]\}$, where$\{\epsilon_{m,t,z} : z \in [2]\}$ are the first two independent elements (hense the basis) of $\{\upsilon_m \eta_t p_{j,*}(\alpha_m) : j \in [4]\}$. Denote $D_{m,v} = tr_{\mathbb{F}/\mathbb{B}}(\epsilon_{m,t,z} N_m), v = 2(t-1) + z$. Since the trace function is a linear function, $\{D_{m,v} : v \in [4]\}$ can reconstruct $tr_{\mathbb{F}/\mathbb{B}}(c_{mi}N_m), i \in [8]$. Hence the trace function of (3) can be evaluated.

Algorithm 1 Repair algorithm for $RS(n,k)$ over $GF(2^\ell)$. For integers a, s, the bandwidth is $\frac{\ell}{a}(a-s)$ per helper.

Input: The failed node index $*$ and the remaining nodes $N_m, m \in [n], m \neq *$.
Output: The failed node N_*.
Preprocessing Steps:
Step 1: Construct **dual basis table** and **dual codeword table** (e.g. Tables I and II): Find the dual basis of $\{c_{*i} : i \in [\ell]\}$, denoted by $\{\gamma_1', \gamma_2', ..., \gamma_\ell'\}$. For each helper node m, find the ℓ dual codewords $c_{mi}, i \in [\ell]$ in (4).
Step 2: Construct **subspace basis table** and **representation table** (e.g. Tables III and IV): Find the basis $\{\epsilon_{m,t,z}, t \in [\frac{\ell}{a}], z \in [a-s]\}$ of the subspace spanned by $\{c_{mi} : i \in [\ell]\}$, and the representation of $tr_{\mathbb{F}/\mathbb{B}}(c_{mi}N_m)$ for each helper m.
Repair Steps:
Step 3: Calculate and download the binary values $D_{m,v} = tr_{\mathbb{F}/\mathbb{B}}(\epsilon_{m,t,z}N_m)$ from each helper, $v \in [\frac{\ell}{a}(a-s)]$.
Step 4: Represent $tr_{\mathbb{F}/\mathbb{B}}(c_{mi}N_m)$ from $D_{m,v}$ and apply (3):

$$b_i' = tr_{\mathbb{F}/\mathbb{B}}(c_{*i}N_*) = \sum_{m=1, m\neq *}^{n} tr_{\mathbb{F}/\mathbb{B}}(c_{mi}N_m), i \in [\ell]. \tag{7}$$

Step 5: Reconstruct the failed node by

$$N_* = \sum_{i=1}^{\ell} \gamma_i' b_i'. \tag{8}$$

For the general case, Eq. (4) is replaced by

$$\{c_{mi} : i \in [\ell]\} = \{\upsilon_m \eta_t p_{j,*}(\alpha_m) : t \in [\frac{\ell}{a}], j \in [a]\}, \tag{9}$$

where $m \in [n]$ and the subscript i is indexed by t and j as $i = a \times (t-1) + j$. Here $\{\upsilon_m, m \in [n]\}$ are the column multipliers given by $\upsilon_m = \prod_{j \in [n], j \neq m} (\alpha_m - \alpha_j)$ [13, Thm. 4, Ch.10], and $\{\eta_t : t \in [\frac{\ell}{a}]\}$ is the basis for $\mathbb{F}$ over $\mathbb{E}$. The polynomial $p_{j,*}(x)$ is

$$p_{j,*}(x) = \xi_j \prod_{w \in W} (x - \alpha_* + w^{-1}\xi_j), \tag{10}$$

where $\{\xi_1, \xi_2, \ldots, \xi_a\}$ is a basis for $\mathbb{E}$ over $\mathbb{B}$, and $W = \{w_0 = 0, w_1, w_2, \ldots, w_{q^s-1}\}$ is an s-dimensional subspace

in $\mathbb{E}$, $2^s \leq n-k$. Parameters $\epsilon_{m,t,z}, D_{m,v}$, $v = (a-s)\times(t-1)+z, z \in [a-s], t \in [\frac{\ell}{a}]$, are defined similar to the case of $RS(14,10)$. Moreover, it can be shown that only $\frac{\ell}{a}(a-s)$ bits are downloaded from each helper, and the trace function of (3) can be evaluated.

The detailed steps are provided in Algorithm 1, in which we construct the lookup tables in Steps 1 and 2, then repair the failed node in Steps 3, 4 and 5. Due to space limitation, Tables II, III, and IV are only shown when node 1 fails.

Complexity. In what follows, we analyze the space and computation complexity of our algorithm.

During preprocessing in Steps 1 and 2, Tables I, II, III, and IV need to be calculated only once. In addition, the symbols in Table II are intermediate values and do not need to be stored. Thus, each node stores one corresponding column of Table I, and one corresponding column of 13 different variations of Tables III and IV, where a variation is for one potential failure. Note that $tr_{\mathbb{F}/\mathbb{B}}(c_{mi}N_m), i \in [8]$ lie on two spaces of dimension 2 spanned by $\{D_{m,1}, D_{m,2}\}$ and $\{D_{m,3}, D_{m,4}\}$, hence each symbol in Table IV is 2 bits. The storage overhead per node is 8 symbols of $GF(2^8)$ in Table I, $4 \times 13 = 52$ symbols of $GF(2^8)$ in Table III and $8 \times 13 = 104$ symbols of 2 bits in Table IV. In total, we need 688 bits per node. Moreover, this storage overhead is amortized over the total storage size of the node. For the general case, similar calculations show that the storage overhead per node is $\ell + \frac{\ell}{a}(n-1)(a-s)$ symbols of $GF(2^\ell)$ and $(n-1)\ell$ symbols of $a-s$ bits.

Next, we analyze the computation complexity in the repair steps (Steps 3, 4, and 5). We show that some steps do not require general operations over $\mathbb{F}$. For the commonly used cases of $\ell = 8, 16, 32, 64$, we present a lemma to calculate the trace function.

Lemma 1. Let $\mathbb{F} = GF(2^8), GF(2^{16}), GF(2^{32}), GF(2^{64})$ and $\mathbb{B} = GF(2)$. The trace function $tr_{\mathbb{F}/\mathbb{B}}(x)$ is equal to one single bit of x, for $x \in \mathbb{F}$.

Proof: We prove for the case of $\mathbb{F} = GF(2^8)$, the proof for other fields have similar steps and we only show the results. From [14, Thm 2.25], we know that $tr_{\mathbb{F}/\mathbb{B}}(x) = 0$ if and only if $x = \delta^2 + \delta$ for some $\delta \in \mathbb{F}$. Let us write x as

$$x = x_0 + x_1\beta + x_2\beta^2 + ... + x_7\beta^7, \tag{11}$$

where $x_i \in \mathbb{B}, i = 0, 1, ..., 7$ are the 8-bit presentation of x, and β is a root of the primitive polynomial $1 + x^2 + x^3 + x^4 + x^8 = 0$. Similarly, we write δ as $\delta = \delta_0 + \delta_1\beta + \delta_2\beta^2 + ... + \delta_7\beta^7$, where $\delta_i \in \mathbb{B}, i = 0, 1, ..., 7$.

Then, we have $tr_{\mathbb{F}/\mathbb{B}}(x) = 0$ if and only if

$$\begin{aligned} x = \delta^2 + \delta &= \delta_4 + \delta_6 + \delta_7 + (\delta_1 + \delta_7)\beta \\ &+ (\delta_1 + \delta_2 + \delta_4 + \delta_5 + \delta_6)\beta^2 \\ &+ (\delta_3 + \delta_4 + \delta_6)\beta^3 + (\delta_2 + \delta_5 + \delta_7)\beta^4 \\ &+ (\delta_3 + \delta_5)\beta^6 + (\delta_6 + \delta_7)\beta^7. \end{aligned} \tag{12}$$

In the above equation, we use the fact that $1 + \beta^2 + \beta^3 + \beta^4 + \beta^8 = 0$ and $\delta_i = {\delta_i}^2$ because δ_i is binary.

Because $\{1, \beta, \beta^2, ..., \beta^7\}$ are linearly independent over $\mathbb{B}$, from (11) and (12), we get $tr_{\mathbb{F}/\mathbb{B}}(x) = 0$ if and only if $x_5 = 0$. Thus, $tr_{\mathbb{F}/\mathbb{B}}(x)$ is the same as the 6-th bit of x.

The results for other field sizes are tabulated below:

ℓ	primitive polynomial	bit index
8	$x^8 + x^4 + x^3 + x^2 + 1$	6
16	$x^{16} + x^{14} + x^{10} + x^8 + x^3 + x + 1$	14
32	$x^{32} + x^{22} + x^2 + x + 1$	32
64	$x^{64} + x^4 + x^3 + x^2 + 1$	62

■

In Step 3, we need to perform 4 multiplications in $GF(2^8)$ to calculate $\epsilon_{m,t,z}N_m, t \in [2], z \in [2]$ and 4 trace functions to calculate $D_{m,v}$ from each helper. From Lemma 1, the cost of the trace function is just checking the 6-th bit and can be ignored. In Step 4, as shown in Table IV, the representation needs only 2 additions: $D_{m,1} + D_{m,2}$ and $D_{m,3} + D_{m,4}$, and Eq. (7) needs 8×13 additions in $GF(2)$. This step is done in the failed node. In Step 5, since $b'_i, i \in [8]$ are binary symbols, at most 7 additions in $GF(2^8)$ are needed. Therefore, in total we need to perform 4 multiplications in $GF(2^8)$ at each helper, plus 130 additions in $GF(2)$ and 7 additions in $GF(2^8)$ at the failed node. For general parameters, we need to perform $\frac{\ell}{a}(a-s)$ multiplications in $GF(2^\ell)$ at each helper, plus $(n-1)\frac{\ell}{a} \times [2^{a-s} - (a-s) - 1] + (n-1)\ell$ additions in $GF(2)$, and $\ell - 1$ additions in $GF(2^\ell)$ at the failed node.

III. Comparison

For $RS(14,10)$ in $GF(2^8)$, our algorithm requires a repair bandwidth of 52 bits for one failure, which is 35% better than the naive repair [11] that requires 80 bits. The full-length code in [4], [5] requires 78 bits using the general scheme. For the special case of $RS(14,10)$ in $GF(2^8)$ and through exhaustive search, [4] found a method that needs at most 64 bits, however, it does not work for other coding parameters. The MSR scheme in [6] and the asymptotic MSR scheme in [7] do not provide a solution for small fields like $GF(2^8)$. Instead, they require a field size of at least $\ell \approx 5 \times 10^{23}$ and $\ell \approx 2.7 \times 10^8$, respectively. For the special case of $RS(14,10)$, [10] provides three different repair schemes that require 60, 56, and 54 bits.

Our approach can be applied to any RS code as long as there is an integer a such that $a|l$ and $2^a > n+1$. For the $RS(11,8)$ in Yahoo Object Store [15], we can set $n = 11, k = 8, a = 4, s = 1$. When applied to $GF(2^8)$, the repair bandwidth of our algorithm is 60 bits, which is 6% better than the naive scheme. For the $RS(12,8)$ in Baidu's Atlas cloud storage [16], we can set $n = 12, k = 8, a = 4, s = 2$. When applied to $GF(2^8)$, we can achieve a repair bandwidth of 44 bits, which is 31% smaller than the naive scheme. In both cases, the full-length code's repair bandwidth [4], [5] are higher than those of the naive scheme.

TABLE II
Dual codeword table. It shows the symbols $c_{mi} = \upsilon_m \eta_t p_{j,*}(\alpha_m), i = 4(t-1)+j$, for $RS(14,10)$ when Node $* = 1$ fails.

helper m	2	3	4	5	6	7	8	9	10	11	12	13	14
$i=1(t=1,j=1)$	β^{17}	1	0	1	β^{34}	β^{187}	β^{102}	β^{238}	β^{17}	β^{17}	β^{119}	0	β^{119}
$i=2(t=1,j=2)$	β^{17}	β^{119}	β^{68}	0	β^{68}	β^{68}	β^{204}	β^{85}	β^{153}	β^{51}	0	β^{102}	β^{238}
$i=3(t=1,j=3)$	β^{17}	1	β^{204}	0	β^{170}	0	β^{68}	0	β^{17}	β^{51}	β^{238}	β^{136}	β^{17}
$i=4(t=1,j=4)$	β^{119}	β^{119}	0	β^{119}	β^{34}	β^{187}	β^{204}	0	β^{51}	β^{51}	β^{17}	β^{238}	β^{17}
$i=5(t=2,j=1)$	β^{18}	β	0	β	β^{35}	β^{188}	β^{103}	β^{239}	β^{18}	β^{18}	β^{120}	0	β^{120}
$i=6(t=2,j=2)$	β^{18}	β^{120}	β^{69}	0	β^{69}	β^{69}	β^{205}	β^{86}	β^{154}	β^{52}	0	β^{103}	β^{239}
$i=7(t=2,j=3)$	β^{18}	β	β^{205}	0	β^{171}	0	β^{69}	0	β^{18}	β^{52}	β^{239}	β^{137}	β^{18}
$i=8(t=2,j=4)$	β^{120}	β^{120}	0	β^{120}	β^{35}	β^{188}	β^{205}	0	β^{52}	β^{52}	β^{18}	β^{239}	β^{18}

TABLE III
Subspace basis table. It shows $\epsilon_{m,t,z}$ for $RS(14,10)$ when Node $* = 1$ fails.

helper m	2	3	4	5	6	7	8	9	10	11	12	13	14
$\epsilon_{m,1,1}$	β^{17}	1	β^{68}	1	β^{34}	β^{187}	β^{102}	β^{238}	β^{17}	β^{17}	β^{119}	β^{102}	β^{119}
$\epsilon_{m,1,2}$	β^{119}	β^{119}	β^{204}	β^{119}	β^{68}	β^{68}	β^{204}	β^{85}	β^{153}	β^{51}	β^{238}	β^{136}	β^{238}
$\epsilon_{m,2,1}$	β^{18}	β	β^{69}	β	β^{35}	β^{188}	β^{103}	β^{239}	β^{18}	β^{18}	β^{120}	β^{103}	β^{120}
$\epsilon_{m,2,2}$	β^{120}	β^{120}	β^{205}	β^{120}	β^{69}	β^{69}	β^{205}	β^{86}	β^{154}	β^{52}	β^{239}	β^{137}	β^{239}

TABLE IV
Representation table. It represents $tr_{\mathbb{F}/\mathbb{B}}(c_{mi}N_m)$ by $D_{m,v} = tr_{\mathbb{F}/\mathbb{B}}(\epsilon_{m,t,z}N_m)$, $v = 2(t-1)+z$, for $RS(14,10)$ when Node 1 fails.

helper	2	3	4	5	6	7	8	9	10	11	12	13	14
$i=1$	$D_{2,1}$	$D_{3,1}$	0	$D_{5,1}$	$D_{6,1}$	$D_{7,1}$	$D_{8,1}$	$D_{9,1}$	$D_{10,1}$	$D_{11,1}$	$D_{12,1}$	0	$D_{14,1}$
$i=2$	$D_{2,1}$	$D_{3,2}$	$D_{4,1}$	0	$D_{6,2}$	$D_{7,2}$	$D_{8,2}$	$D_{9,2}$	$D_{10,2}$	$D_{11,2}$	0	$D_{13,1}$	$D_{14,2}$
$i=3$	$D_{2,1}$	$D_{3,1}$	$D_{4,2}$	0	$D_{6,1}$ $+D_{6,2}$	0	$D_{8,1}$ $+D_{8,2}$	0	$D_{10,1}$	$D_{11,2}$	$D_{12,2}$	$D_{13,2}$	$D_{14,1}$ $+D_{14,2}$
$i=4$	$D_{2,2}$	$D_{3,2}$	0	$D_{5,2}$	$D_{6,1}$	$D_{7,1}$	$D_{8,2}$	0	$D_{10,1}$ $+D_{10,2}$	$D_{11,2}$	$D_{12,1}$ $+D_{12,2}$	$D_{13,1}$ $+D_{13,2}$	$D_{14,1}$ $+D_{14,2}$
$i=5$	$D_{2,3}$	$D_{3,3}$	0	$D_{5,3}$	$D_{6,3}$	$D_{7,3}$	$D_{8,3}$	$D_{9,3}$	$D_{10,3}$	$D_{11,3}$	$D_{12,3}$	0	$D_{14,1}$
$i=6$	$D_{2,3}$	$D_{3,4}$	$D_{4,3}$	0	$D_{6,4}$	$D_{7,4}$	$D_{8,4}$	$D_{9,4}$	$D_{10,4}$	$D_{11,4}$	0	$D_{13,3}$	$D_{14,2}$
$i=7$	$D_{2,3}$	$D_{3,3}$	$D_{4,4}$	0	$D_{6,3}$ $+D_{6,4}$	0	$D_{8,3}$ $+D_{8,4}$	0	$D_{10,3}$	$D_{11,4}$	$D_{12,4}$	$D_{13,4}$	$D_{14,1}$ $+D_{14,2}$
$i=8$	$D_{2,4}$	$D_{3,4}$	0	$D_{5,4}$	$D_{6,3}$	$D_{7,3}$	$D_{8,4}$	0	$D_{10,3}$ $+D_{10,4}$	$D_{11,4}$	$D_{12,3}$ $+D_{12,4}$	$D_{13,3}$ $+D_{13,4}$	$D_{14,1}$ $+D_{14,2}$

REFERENCES

[1] W. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge University Press, 2009.

[2] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," in *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*. IEEE, 1998, pp. 28–37.

[3] H. Dau, I. M. Duursma, H. M. Kiah, and O. Milenkovic, "Repairing reed-solomon codes with multiple erasures," *IEEE Transactions on Information Theory*, vol. 64, no. 10, pp. 6567–6582, 2018.

[4] V. Guruswami and M. Wootters, "Repairing Reed-Solomon codes," *IEEE Transactions on Information Theory*, 2017.

[5] H. Dau and O. Milenkovic, "Optimal repair schemes for some families of full-length Reed-Solomon codes," in *Information Theory (ISIT), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 346–350.

[6] M. Ye and A. Barg, "Explicit constructions of MDS array codes and RS codes with optimal repair bandwidth," in *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1202–1206.

[7] I. Tamo, M. Ye, and A. Barg, "Optimal repair of reed-solomon codes: Achieving the cut-set bound," in *2017 IEEE 58th Annual Symposium*

[8] 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.

on Foundations of Computer Science (FOCS). IEEE, 2017, pp. 216–227.

[9] W. Li, Z. Wang, and H. Jafarkhani, "On the sub-packetization size and the repair bandwidth of reed-solomon codes," *IEEE Transactions on Information Theory*, 2019.

[10] I. Duursma and H. Dau, "Low bandwidth repair of the RS (10, 4) Reed-Solomon code," in *2017 Information Theory and Applications Workshop (ITA)*. IEEE, 2017, pp. 1–10.

[11] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang *et al.*, "f4: Facebook's warm blob storage system," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014, pp. 383–398.

[12] https://wiki.apache.org/hadoop/HDFS-RAID, accessed on Jul. 2015.

[13] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*. Elsevier, 1977.

[14] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. Cambridge university press, 1994.

[15] https://yahooeng.tumblr.com/post/116391291701/yahoo-cloud-object-store-object-storage-at, yahoo Cloud Object StoreObject Storage at Exabyte Scale. Accessed: 2017.

[16] 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 *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2015, pp. 1–14.