ISIT'2013 1569718835

Characterizations and construction methods for linear functional-repair storage codes

Henk D.L. Hollmann and Wencin Poh
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore
Email: {lhenk.hollmann, wcpoh}@ntu.edu.sg

***Abstract*—We present a precise characterization of linear functional-repair storage codes in terms of *admissible states*, with each state made up from a collection of vector spaces over some fixed finite field. To illustrate the usefulness of our characterization, we provide several applications. We first describe a simple construction of functional-repair storage codes for a family of code parameters meeting the cutset bound outside the MBR and MSR points; these codes are conjectured to have optimal rate with respect to their repair locality. Then, we employ our characterization to develop a construction method to obtain functional repair codes for given parameters using symmetry groups, which can be used both to find new codes and to improve known ones. As an example of the latter use, we describe a beautiful functional-repair storage code that was found by this method, with parameters belonging to the family investigated earlier, which can be specified in terms of only eight different vector spaces.**

I. Introduction

A distributed storage system (DSS) typically stores data objects in encoded form on multiple storage units, commonly referred to as *storage nodes*. Over time, the DSS will have to handle the occasional loss of storage nodes, for example due to hardware or software failures or peer churning (in peer-to-peer storage systems). This is usually referred to as the *repair problem*. Under the simplest repair regime, each data block on a failed node has to be reconstructed *exactly* and stored on a newcomer node. Greater efficiency (e.g., higher rates) can sometimes be achieved by employing a repair regime called functional repair, where the replacement blocks do not need to be an exact copy of the lost data blocks, but are merely required to contain sufficient information to maintain data integrity (in a moment, this will be discussed in more detail). A *storage code* is the precise description of how the DSS handles the data storage and repair management.

Many different performance criteria for storage code efficiency have been considered. The *repair bandwidth*, the total amount of data traffic needed during repair, has received the most attention until now, and is currently the best understood. Another performance measure is the repair locality, the number of nodes that need to be contacted during repair. In applications such as cloud storage or deep archival, minimizing disk I/O seems to be the main consideration. Since the disk I/O is proportional to the number of nodes contacted during repair, the repair locality has emerged as an important parameter and has been much investigated recently.

Under the functional repair regime, lost data blocks have to be replaced by "functionally equivalent" blocks, in the following sense. The obvious requirement is that after repair the original information that was stored can still be retrieved. A less obvious demand is that the replacement must guarantee that also *future* (functional) repairs remain possible. Typically, the new data block is replaced by a linear combination of some of the data stored on a subset of the other blocks, in such a way that "enough independency" is maintained in the system. There are several problems with this description. First of all, it seems that some kind of "invariant" needs to be maintained in the system, but it is not clear what this invariant should be. Secondly, in contrast with an exact repair regime where for a given stored data file only a limited number of different data blocks can occur and each node has a clear functional identity, under a functional repair regime, in principle every possible data block might appear and nodes do not have a clear identity. These facts complicate the system management, and make code design less evident, especially for small repair locality. Indeed, few codes for the functional repair regime are known [1].

In this paper, we present a rather general method to construct functional-repair storage codes for given parameters and symbol alphabet size. To this end, we first offer a description of (exact and functional repair) *linear* storage codes in terms of vector spaces over finite fields. We remark that most known codes can be described in this way. One of the advantages offered by such a description is that it emphasizes the underlying structure of the storage code. This opens the way to investigate and construct storage codes with a certain common structure, similar to the situation in the theory of error-correcting codes, where one studies for example cyclic codes or other highly structured codes, thus facilitating code design or correction management. Here, we use this description to clarify the idea of a hidden "invariant" that has to be maintained, in a way that enables, in principle, to decide on the existence or non-existence of a linear storage code for given "small" parameters. The idea here is to characterize the code in terms of *admissible states*, where each state is a collection of vector spaces that essentially describes a possible set of data blocks stored on the nodes at some moment in time. Finally, we describe a method to use symmetry groups to search for small storage codes (having a small set of admissible states) with a high degree

of symmetry (technically, having a transitive automorphism group). Search methods for exact-repair storage codes are described for example in [2], but as far as we know search methods for functional-repair storage codes have not been published before.

To illustrate our ideas, we investigate a class of storage code parameters that has been proved or conjectured to provide optimal rate for a given repair locality [3]. In most cases, storage codes with these parameters are necessarily functional-repair. Since these codes can be considered as regenerating codes, it is known that functional repair codes with these parameters exist provided that the symbol alphabet size is large enough. We use our methods to give a simple, explicit construction for these codes, which enables the use of our group-theoretical construction techniques to search for smaller codes with the same parameters. As an example, we describe one such code thus found that can be described in terms of only eight different binary vector spaces.

For an overview of DSS and storage codes, we refer to [4] or [5], or to the Storage Wiki [1].

II. Storage codes

A *storage code* with parameters $(m, n, k, r, \alpha, \beta)$ is a code that allows *resilient* storage of m information symbols from some finite alphabet $\mathbb{F}$, in encoded form, onto n storage nodes, each capable of holding α data symbols from $\mathbb{F}$. We will refer to α as the *storage node capacity*. The parameter k indicates that at all times, the original stored information can be recovered from the data stored on *some* set of k nodes. If *any* set of k nodes will do, the the code is referred to as MDS [6]. The *rate* of the code is the fraction $m/(n\alpha)$ of information per stored symbol. The resilience of the code is described in terms of a parameter r, referred to as the *repair locality*, and a parameter β, referred to as as the *transport capacity* of the code. If a node fails, then a newcomer node is allowed to contact *some* set of r live nodes, called the *repair set*; each of these r nodes computes an amount of β data symbols, which are then downloaded by the newcomer node in order to regenerate some of the lost information, in the form of a replacement block again consisting of α data symbols. If the code is *exact-repair*, then we require that this replacement block is an *exact* copy of the lost data block. However, for certain purposes this repair modality is too strict, and it is convenient to consider a weaker form of resilience. We say that the code allows *functional repair* if the replacement block is *information equivalent* to the lost data block, while ensuring the possibility of future functional repair of other nodes. The notion of functional repair is more subtle and much more difficult to grasp; we will provide more explanation and various examples later in this paper. The storage code describes the entire data and repair management of the DSS.

In [3], a framework has been developed for the description of linear storage codes in terms of vector spaces over a finite field. Given a linear storage code for the exact repair regime, it is rather straightforward to construct such a description, however for linear storage codes under functional repair, things are less clear. In the remainder of this section, we briefly review the relevant notions from [3]; in the next section, we will use these ideas to derive a characterization of linear functional repair storage codes.

A. Linear storage codes

In this paper, we will think of a *linear* exact-repair storage code as a collection of n subspaces $U_1, \ldots, U_n$ of an m-dimensional vector space $\mathbb{F}^m$, each of dimension α. We will refer to $\mathbb{F}^m$ as the *message space* and to the U_i as the *storage node spaces* or, more briefly, as the *node spaces*. The integer α is called the *(storage) node capacity*.

A *recovery set* of the storage code is a subset of the storage node spaces that together span the entire message space $\mathbb{F}^m$. Here, the *span* of a collection of vector spaces $W_1, \ldots, W_k$ is the collection of all vectors $w_1 + \cdots w_k$ with $w_i \in W_i$ for all i, that is, the smallest vector space containing all the vector spaces $W_1, \ldots, W_k$. The *recovery dimension* of the storage code is defined as the *smallest* size k of the a recovery set.

A linear storage code as above can be used to store an amount of m data symbols in n storage nodes, in the following way. Associate the n storage nodes $v_1, \ldots, v_n$ with the n storage spaces $U_1, \ldots, U_n$, and choose a fixed basis in each of them. Now, represent the data to be stored as an vector $x \in \mathbb{F}^m$; then, in node v_i, we store the α inner products of x with the α basis vectors of U_i. In other words, if B_i is the $m \times \alpha$ matrix that has as its columns the basis vectors for U_i, then storage node v_i stores $x^\top B_i$. Note that, consequently, the DSS can compute the inner product of the data vector x with any vector contained in one of the node spaces; moreover, from the data stored in a recovery set, which by definition is a spanning subset of the node spaces, the DSS can recover x.

Now let us consider the repair problem. Fix some positive integer β, referred to as the *transport capacity* of the storage code. We will say that a collection R of the node spaces is a *repair set* for a certain node space $U_\ell \notin R$ if it is possible to choose in each $U_i \in R$ a β-dimensional *repair space* $W_{i,\ell}$ such that U_ℓ is contained in the span of the repair spaces $W_{i,\ell}$. Note that if we represent each repair space $W_{i,\ell}$ by a fixed $m \times \beta$ matrix $B_{i,\ell}$ having the vectors of a fixed basis for $W_{i,\ell}$ as its columns, then the vector $x^\top B_\ell$ stored in node v_ℓ can be recovered from a linear combination of the vectors $x^\top B_{i,\ell}$, which in turn can be computed from the vectors $x^\top B_i$ stored in the nodes v_i involved in the repair set.

If *each* node space in the code has a repair set of size r with respect to transport capacity β then we say that the storage code has *repair locality* r with respect to transport capacity β. We will refer to a storage code with all the above parameters as a *linear exact-repair* $(m; n, k, r, \alpha, \beta)$*-storage code*. Note that such a storage code will have a coding rate $R = m/(n\alpha)$. The following simple example illustrates the above notions.

Example 2.1: Consider the linear storage code with node spaces $U_0 = \langle e_0, e_2 + e_3 \rangle$, $U_1 = \langle e_1, e_3 + e_0 \rangle$, $U_2 = \langle e_2, e_0 + e_1 \rangle$, and $U_3 = \langle e_3, e_1 + e_2 \rangle$, considered as subspaces of $\mathbb{F}_2^4$. Here, we write $\langle w_1, \ldots, w_k \rangle$ to denote the span of the vectors

$w_1, \ldots, w_k$, the vector space consisting of all linear combinations of $w_1, \ldots, w_k$. We claim that these node spaces constitute an $(m = 4; n = 4, k = 2, r = 3, \alpha = 2, \beta = 1)$ linear exact-repair storage code. Indeed, there are $n = 4$ node spaces U_i, each of dimension $\alpha = 2$. Furthermore, $k = 2$ since any two subspaces intersect trivially, so together span the entire space $\mathbb{F}_2^4$. Furthermore, the set $R = \{U_1, U_2, U_3\}$ is a repair set for node space U_0, with respect to transport capacity $\beta = 1$. Indeed, if we choose repair spaces $W_{1,0} = \langle e_0 + e_3 \rangle \subseteq U_1$, $W_{2,0} = \langle e_2 \rangle \subseteq U_2$, and $W_{3,0} = \langle e_3 \rangle \subseteq U_3$ (each of dimension $\beta = 1$), then $U_0 \subseteq \langle e_0 + e_3, e_2, e_3 \rangle = W_{1,0} + W_{2,0} + W_{3,0}$ as required. The storage code has rotational symmetry: the linear transformation given by $e_i \mapsto e_{i+1}$ (indices modulo 4) maps U_i to U_{i+1}; as a consequence, repair sets for the other node spaces can be obtained by symmetry. With the bases as suggested by the above description, this code stores a data vector $x = (x_0, \ldots, x_3)$ by letting node 0 hold x_0 and $x_2 + x_3$, (and letting node i hold x_i and $x_{i+2} + x_{i-1}$ for $i = 1, 2, 3$), and repairs node 0 by downloading $x_0 + x_3$ from node 1, x_2 from node 2, and x_3 from node 3. (Storage and repair for the other nodes follows by symmetry.) □

B. Linear functional-repair storage codes

The linear exact-repair storage codes in the previous section had the property that a *fixed* vector space was associated with each node. For linear *functional-repair* storage codes, this will no longer be the case. Recall that under the regime of functional repair, a data block on a failed storage node has to be replaced by a data block on a newcomer that is *information equivalent* to the one on the failed node, while ensuring the possibility of future functional repair of other nodes. Linear functional-repair storage codes are perhaps best thought of as a specification of *admissible* node space arrangements, with the property that in every such arrangement, a node space can be "repaired" by replacing it with a (possibly different) space so that the resulting arrangement again satisfies the specifications. Consider the following example.

Example 2.2: We will construct a linear functional-repair storage code with parameters $(m = 5; n = 4, k = r = 3, \alpha = 2, \beta = 1)$, so with coding rate $R = 5/8$. As message space we take $\mathbb{F}_2^5$. We will ensure that at any moment, the four 2-dimensional node spaces $U_1, \ldots, U_4$ associated with the four storage nodes satisfy the following specification:

1) Any two of the node spaces intersect trivially, that is, $U_i \cap U_j = \{0\}$ when $i \neq j$;
2) Any three of the node spaces span the entire message space $\mathbb{F}^5$.

Assuming that $U_1, \ldots, U_4$ satisfy these constraints, suppose that node 4 fails. Without loss of generality, we may assume that $U_1 = \langle e_1, a_1 \rangle$, $U_2 = \langle e_2, a_2 \rangle$, and $U_3 = \langle e_3, a_3 \rangle$, for some basis e_1, e_2, e_3, a_1, a_2 of $\mathbb{F}^5$, with $a_1 + a_2 + a_3 = 0$. Indeed, U_3 must have trivial intersection with both U_1 and U_2, but has to intersects the 4-dimensional span $U_1 + U_2$, hence this intersection is of the form $a_1 + a_2$ with $a_i \in U_i^* = U_i \setminus \{0\}$. This shows that U_1, U_2, U_3 must have the indicated form. Now, to repair (or initially construct) the storage space U_4, given that $\beta = 1$ we must choose a vector $w_i \in U_i^*$ for $i = 1, 2, 3$, and let U_4 be some 2-dimensional subspace of their span $\langle w_1, w_2, w_3 \rangle$, which by rule 1 should not contain any of the w_i. Hence U_4 is of the form $\{0, w_1 + w_2, w_1 + w_3, w_2 + w_3\}$. Furthermore, $w_3 \neq a_1 + a_2$ since otherwise $U_4 \subset U_1 + U_2$, violating rule 2, and similarly, $w_1 \neq a_1, w_2 \neq a_2$. So $w_1 = e_1 + x_1 a_1, w_2 = e_2 + x_2 a_2, w_3 = e_3 + x_3 a_3$, and it is now easily verified that any choice of $x_1, x_2, x_3 \in \mathbb{F}_2$ is valid. (Initially, we can take, for example, $U_4 = \langle e_1 + e_2, e_1 + e_3 \rangle$.) This shows that we can maintain the specification forever, provided that no two nodes ever fail simultaneously. □

Using a functional-repair storage codes as above to actually store information is similar to using the exact-repair storage codes introduced earlier, except that now at each moment the other nodes and the data collector have to be informed of the actual *state* of a storage node, that is, of its current storage node space, and have to be aware of all the bases used. This extra overhead can be relatively small if the code is used to store a large number of messages *simultaneously*.

III. Admissible states

We will now offer a new characterization of linear functional-repair codes in terms of *admissible states*. Consider again the code in Example 2.2. The crucial facts making the construction work are that the collection $\mathcal{U} = \{U_1, U_2, U_3\}$ satisfies the specification and that a fourth node space U_4 can be constructed such that any triple from $\mathcal{U} \cup \{U_4\}$ again satisfies the specification. In a sense, these collections of triples form the "admissible repairing collections" of the code. This observation motivates the following two definitions.

Definition 3.1: Let $\mathcal{U}$ be a collection of α-dimensional subspaces of a vector space $\mathbb{F}^m$. We say that an α-dimensional subspace U of $\mathbb{F}^m$ can be obtained from $\mathcal{U}$ by (r, β)-*repair* if it is possible to choose r spaces $U_1, \ldots, U_r$ in $\mathcal{U}$, the *repair set*, and then a β-dimensional subspace $W_i \subseteq U_i$ in each of them, the *repair spaces*, such that U is contained in the span $W_1 + \cdots + W_r$ of the repair spaces.

Definition 3.2: A linear functional repair storage code with parameters $(m; n, k, r, \alpha, \beta)$ over some finite field $\mathbb{F}$ is a set $\mathcal{A}$ of $(n-1)$-tuples $\mathcal{U}$ of α-dimensional subspaces of $\mathbb{F}^m$, such that the following *repair property* holds. Given any $(n-1)$-tuple $\mathcal{U} = \{U_1, \ldots, U_{n-1}\}$ in $\mathcal{A}$, there exists an α-dimensional space U that can be obtained from $\mathcal{U}$ by (r, β)-repair such that for every $i = 1, \ldots, n-1$, the $(n-1)$-tuple $\mathcal{U} \cup \{U\} \setminus \{U_i\}$ is again in $\mathcal{A}$. Furthermore, we require that each $(n-1)$-tuple $\mathcal{U}$ in $\mathcal{A}$ contains a *spanning subset* of size k, that is, there can be found k spaces in $\mathcal{U}$ that together span $\mathbb{F}^m$.

We will sometimes refer to the $(n-1)$-tuples $\mathcal{U}$ in $\mathcal{A}$ as the *admissible repairing collections* of the code, and to the pairs $(\mathcal{U}, U)$ as the *admissible states* of the code. We invite the reader to verify that an exact-repair storage code with parameters as in Definition 3.2 and storage node spaces $U_1, \ldots, U_n$ is also of this type, with as admissible states $(\{U_1, \ldots, U_n\} \setminus \{U_i\}, U_i)$ for $i = 1, \ldots, n$. Returning to the code in Example 2.2, we recognize it to also be of this type, with as admissible repairing collections all triples

$\mathcal{U} = \{U_1, U_2, U_3\}$ of 2-dimensional node spaces in $\mathbb{F}^5$ that together span $\mathbb{F}^5$ in which any pair of node spaces intersecting trivially.

We hope that the following characterization result is evident by now. For a full explanation, we refer to [7].

Theorem 3.3: Every linear functional-repair code according to Definition 3.2 is indeed a storage code under the regime of fuctional repair. Conversely, every linear functional-repair code can be obtained from a collection $\mathcal{A}$ of admissible repairing collections with the properties as in Definition 3.2.

The above results open the way to construction methods for functional-repair codes, and methods to find better, smaller such codes, as we will illustrate in the remainder of this paper. To the best of our knowledge, a description of functional-repair codes in terms of "admissible states" is new. In a few published constructions for functional-repair codes, some notion of state can be recognized [8], [9].

IV. A family of functional-repair codes

In this section, we will illustrate the usefulness of our characterization of linear functional-repair codes developed above by proving the existence of a family of codes with parameters $(m_{r,s}; n = r+1, k = r, r, \alpha = s+1, \beta = 1)$ and rate $(r-s/2)/(r+1)$, where $m = m_{r,s} = (r-s)(s+1)+s+(s-1)+\cdots+1 = (r-s)(s+1)+\binom{s+1}{2}$ and $r > s \geq 0$. Note that these parameter sets all meet the cutset bound from [10], in points different from the MBR and MSR points. (The code in Example 2.2 is the case where $r = 3, s = 1$.) Moreover, these parameter sets also meet a bound for the maximal rate of a storage code with repair locality r, storage node capacity α, and transport capacity β conjectured in [11].

So the construction idea is to come up with a suitable set $\mathcal{A}$ of admissible repairing collections, and then to show that this set indeed satisfies the requirements in Definition 3.2. We will say that a collection of $(s+1)$-dimensional vector spaces $\mathcal{U} = \{U_1, \ldots, U_r\}$ in $\mathbb{F}^m$ with $m = m_{r,s}$ is (r, s)-*good* if the span of any $r-s+j$ of these subspaces has dimension $(r-s)(s+1)+s+\cdots+(s+1-j)$, for every $j = 0, \ldots, s$. In order to prove that this indeed provides a suitable set of admissible repairing collections, the following result is crucial.

Theorem 4.1: Suppose that $\mathcal{U} = \{U_1, \ldots, U_r\}$ is (r, s)-good over $\mathbb{F}$. Let $w_i \in U_i$ for $i = 1, \ldots, r$, and let U be an α-dimensional subspace of the span W of $w_1, \ldots, w_r$. Let $C \subseteq \mathbb{F}^r$ be the collection of all vectors $c = (c_1, \ldots, c_r)$ for which $\sum_{j=1}^{r} c_j w_j \in U$. Then the collections $\mathcal{U} \cup \{U\} \setminus \{U_i\}$ for $i = 1, \ldots, r$ are all (r, s)-good if and only if the vectors $w_1, \ldots, w_r$ are independent and the code C is an $[r, s+1, r-s]$ linear MDS code over $\mathbb{F}$.

For a proof of this result, we refer to [7]. Note that the mentioned MDS codes certainly exist for a field size $|\mathbb{F}| \geq r-1$, see, e.g., [6]. It follows immediately from the (r, s)-good property that independent vectors $w_1, \ldots, w_r$ as in the lemma can always be found, moreover, they can be used to recursively construct *at least one* (r, s)-good collection for all r and s with $r > s \geq 0$. Therefore, Theorem 4.1 in combination with Theorem 3.3 proves the existence of linear functional-repair codes for the above parameter sets, with field size equal to the smallest prime power q for which $q \geq r-1$.

V. A construction method using groups

One of the drawbacks of functional-repair storage codes is that the set of admissible repairing collections that define the code and the number of admissible states can be huge, which severely complicates the data management in the DSS. Therefore, it is desirable to find *small* codes. For example, if possible we would like to find a small *subset* of admissible repairing collections that itself again defines a code, or we would like to find a small set of admissible states for certain given parameters *directly*. We will now briefly sketch a method to do so, based on symmetry groups.

Suppose that we try to find a linear functional-repair storage code containing some "potential" admissible state $\sigma = (\mathcal{U}, U)$, so where U can be obtained from $\mathcal{U}$ by (r, β)-repair, that we guess to belong to the code. The idea is to construct a storage code with a large symmetry group in which σ is indeed an admissible state. Our method depends on the following.

Theorem 5.1: Let $\sigma = (\mathcal{U} = \{U_1, \ldots, U_{n-1}\}, U = U_n)$, with each of $U_1, \ldots, U_n$ an α-dimensional subspace of $\mathbb{F}^m$, such that U_n can be obtained by (r, β)-repair from $\mathcal{U}$, and with $\mathcal{U}$ containing a spanning k-subset if $k < n-1$. Suppose we can find for every $i = 1, \ldots, n-1$ an invertible linear map L_i mapping $\mathcal{U}$ to $\mathcal{U} \cup \{U_n\} \setminus \{U_i\}$. Then with $\mathcal{G} = \langle L_1, \ldots, L_n \rangle$, the group generated by $L_1, \ldots, L_n$, the set $\mathcal{A}$ consisting of all images $G(\mathcal{U})$ with $G \in \mathcal{G}$ is a collection of admissible repairing collections for a linear storage code with parameters $(m; n, k, r, \alpha, \beta)$.

For the proof of this theorem, we again refer to [7]. We remark that it is sometimes advisable to construct a "very regular" potential (admissible) state $\alpha = (\mathcal{U}, U)$, one for which the stabilizer Γ of σ is transitive on $\mathcal{U}$; in that case, it is sufficient to find a single invertible map L mapping $\mathcal{U}$ to $\mathcal{U} \cup \{U_n\} \setminus \{U_1\}$. In practice, a search can then be set up to see if there is a small group $\mathcal{G}$ generated by (a transitive subgroup of) Γ and one such L.

There are two essentially different situations in which this theorem can be applied. First, if the desired storage code is *not* yet known to exist, then we must "guess" the form of a typical state for the code, after which the hunt for the group can begin. If there already *exists* a storage code with the desired parameters, then we just take any state of the code, and then use the theorem to search for a *small subcode* of the code at hand. As an example of the second type, in the next section we will describe a beautiful functional-repair storage code that was originally found by computer using this method.

VI. A combinatorial description of a small functional-repair code

A *vector space partition* for a vector space V is a collection of subspaces $V_1, \ldots, V_m$, not necessarily all of the same dimension, such that each nonzero vector in V is contained in exactly one of the spaces V_i. It turns out that our code arises from a vector space partition of Beutelspacher type,

see, e.g., [12]. Consider the finite field $W = \mathbb{F}_8$, constructed with a primitive element α with $\alpha^3 = \alpha + 1$. We will consider W as a 3-dimensional vector space over $\mathbb{F}_2$. Note that W has a 2-dimensional subspace $U = \{0, \alpha, \alpha^2, \alpha^4\}$ of W that is invariant under the Frobenius map $x \to x^2$. We will first construct a vector space partition for $V = W \oplus U$. To that end, for each $\beta \in \mathbb{F}_8$, we define $U_\beta = \{(\beta u, u) \mid u \in U\}$. It is easily seen that each U_β is a 2-dimensional subspace of V, with $U_0 = \{0\} \oplus U$; moreover, $U_\beta \cap U_\gamma = \{0\}$ when $\beta \neq \gamma$. Also note that $W = W \oplus \{0\}$ intersects each U_β only in the zero vector. There are 31 nonzero vectors in V, 7 of which are in W and another 3 in each of the 8 subspaces U_β. We conclude that W together with the 8 spaces U_β forms a vector space partition.

We will now describe a *small* linear functional-repair storage code with parameters $(m = 5; n = 4, k = r = 3, \alpha = 2, \beta = 1)$. Earlier, we constructed a storage code for these parameters in Example 2.2. Here, the admissible states of our code will be of the form $\sigma = (\mathcal{U}, U) = (\{U_\beta, U_\gamma, U_\delta\}, U_\epsilon)$ for all subsets $\{\beta, \gamma, \delta\}$ of size 3 from $\mathbb{F}_8$; for each such subset $\{\beta, \gamma, \delta\}$, there will be a *unique* ϵ such that $\sigma$is a coding state. It can be shown that, in fact, the unique choice for ϵ is to let $\epsilon^2 = \beta\gamma + \beta\delta + \gamma\delta$.

The group $G = A\Gamma L(1, \mathbb{F}_8)$ (sometimes written as $\Gamma A_1(8)$) consists of all semi-linear transformations $g : x \to g(x) = ax^{2^i} + b$ for $a \in \mathbb{F}_8 \setminus \{0\}$, $b \in \mathbb{F}_8$, and $i \in \mathbb{Z}_3$. It is not difficult to see that these operations indeed form a group: since $x \to x^2$ acts linear on $\mathbb{F}_8$, if $g : x \to ax^{2^i} + b$ and $h : x \to cx^{2^j} + d$, then $h \circ g : x \to ca^{2^j} x^{2^{ij}} + (cb^{2^j} + d)$ (note that $\alpha^{2^3} = \alpha = \alpha^{2^0}$, so it is proper to consider i modulo 3). We let G act on the vector space V above by associating with each group element $g : x \to ax^{2^i} + b$ the linear transformation $L_g : (w, u) \to (aw^{2^i} + bu^{2^i}, u^{2^i})$ on V. Since $L_h L_g = L_{h \circ g}$, this is a well-defined group action. As a result, we have that $L_g : U_\beta \to U_{g(\beta)}$. We remark that the subgroup $\mathrm{AGL}(1, \mathbb{F}_8)$ consisting of the *linear* transformations in $A\Gamma L(1, \mathbb{F}_8)$, acts regular and 3-homogeneous on $\mathbb{F}_8$, see, e.g., [13], which provides an alternative way to quickly check that we indeed obtain a storage code. This code is related to structures described in [14]. In fact, this code was originally obtained by the group-theoretical methods described in the previous section, employing an initial state involving an (essentially unique) $(r = 3, s = 1)$-good collection. The group fixing such a state has a transitive subgroup $\langle T \rangle$, and there are eight candidate maps L mapping the state to a suitable successor state, two of which generate a group $\langle T, L \rangle \cong G = A\Gamma L(1, \mathbb{F}_8)$. Since all other states are images of the initial state by some linear transformation, we know that the 8 subspaces U_β of $\mathbb{F}_2^5$ ($\beta \in \mathbb{F}_8$) have the property that any two are independent (have intersection $\{0\}$) and any three span the entire space $\mathbb{F}_2^5$. In fact, this property characterizes the storage code. Indeed, it is possible to show that 8 is the maximum number of 2-dimensional vector spaces in $\mathbb{F}_2^5$ that can have this property; moreover, any such collection of 8 spaces is equivalent under some linear transformation to the 8 spaces U_β defined above. For further details on precisely how this code was obtained and for proofs, we refer to [7].

VII. Conclusion

We have developed a characterization of linear functional-repair storage codes over a finite field $\mathbb{F}$ in terms of admissible states and admissible repairing collections of the code, collections of vector spaces over $\mathbb{F}$ with precisely described properties. We have illustrated the usefulness of this characterization by using it to construct functional-repair storage codes for a family of parameters conjectured to have maximal possible rate in terms of repair locality, and two other parameters, the storage node capacity α and transport capacity β; these parameter sets also meet the cutset bound, in a point different from the MBR and MSR points. Our new characterization has also provided a general construction method for linear storage codes employing symmetry groups. Finally, we have described a beautiful small functional-repair storage code with a large, transitive automorphism group obtained with our methods.

Acknowledgment

The authors would like to thank Lluis Pamies-Juarez and Frederique Oggier for proofreading and providing some useful feedback. The research of Henk D.L. Hollmann and Wencin Poh is supported by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

References

[1] "The erasure coding for distributed storage wiki." [Online]. Available: http://tinyurl.com/storagecoding

[2] D. Cullina, A. G. Dimakis, and T. Ho, "Searching for minimum storage regenerating codes," *CoRR*, vol. abs/0910.2245, 2009.

[3] H. D. Hollmann, "Storage codes – coding rate and repair locality," in *2013 International Conference on Computing, Networking and Communications (ICNC'13)*, San Diego, USA, Jan. 2013, pp. 830–834, available at http://arxiv.org/abs/1301.4300.

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

[5] A. Datta and F. Oggier. (2011) An overview of codes tailor-made for networked distributed data storage. [Online]. Available: http://arxiv.org/abs/1109.2317

[6] F. MacWilliams and N. Sloane, *The theory of error correcting codes*, ser. North-Holland mathematical library. North-Holland Pub. Co., 1978.

[7] H. D. Hollmann and W. Poh, "Linear functional-repair storage codes – characterizations and construction methods," in preparation.

[8] K. Shum and Y. Hu, "Functional-repair-by-transfer regenerating codes," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, july 2012, pp. 1192 –1196.

[9] Y. Hu, P. P. C. Lee, and K. W. Shum, "Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems," *CoRR*, vol. abs/1208.2787, 2012.

[10] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, 2010.

[11] H. D. Hollmann, "On the minimum storage overhead of distibuted storage codes with a given repair locality," in preparation.

[12] O. Heden, "A survey of the different types of vector space partitions," *Discrete Math., Alg. and Appl.*, vol. 4, no. 1, 2012.

[13] D. Livingstone and A. A. Wagner, "Transitivity of finite permutation groups on unordered sets," *Math. Zeitschr.*, vol. 90, pp. 393– 403, 1965.

[14] M. H. Klin, J. Lauri, and M. Ziv-Av, "Links between two semisymmetric graphs on 112 vertices via association schemes," *J. Symb. Comput.*, vol. 47, no. 10, pp. 1175–1191, 2012.