T-Code: 3-Erasure Longest Lowest-Density MDS Codes

Sheng Lin, Gang Wang, Member, IEEE, Douglas S. Stones, Xiaoguang Liu and Jing Liu

Abstract—In this paper, we study longest lowest-density MDS codes, a simple kind of multi-erasure array code with optimal redundancy and minimum update penalty. We prove some basic structure properties for longest lowest-density MDS codes. We define a “perfect” property for near-resolvable block designs (NRBs) and establish a bijection between 3-erasure longest lowest-density MDS codes (T-Codes) and perfect NRB(3k+1,3,2)s. We present a class of NRB(3k+1,3,2)s, and prove that it produces a family of T-Codes. This family is infinite assuming Artin’s Conjecture. We also test some other NRBs and find some T-Code instances outside of this family.

Index Terms—3-erasure correcting codes, parity array codes, near-resolvable design, perfect one-factorization.

I. INTRODUCTION

In the last two decades, along with the fast progress of large-scale data storage systems, especially large-scale networked storage systems, multi-erasure coding techniques have attracted increasing attention. An m-erasure code for a storage system is a scheme that encodes the content on n data disks into m check disks so that the system is resilient to any m disk failures [1]. Erasure codes have been used for many applications, such as traditional disk arrays, data grids, peer-to-peer applications, digital fountains, etc [1]. Unfortunately, there is no consensus on the best coding technique for general m, n > 1.

Reed-Solomon (RS) codes [2] are a well-known example of a multi-erasure code. They are also the only known maximum distance separable (MDS) codes for arbitrary n and m. MDS codes have optimal storage efficiency. On the other hand, the computational complexity of using MDS codes poses a significant problem. To relieve this problem, optimized algorithms have been developed for operations over a Galois field [3].

Binary linear codes [4] are linear codes that are inherently XOR-based. They have minimal computational complexity, hence allowing for more efficient encoding and decoding algorithms. Binary linear codes are divided into overlapping parity groups, i.e., each data disk participates in multiple parity groups (in our case m parity groups) in order to recover from m erasures. However, poor storage efficiency is a drawback to binary linear codes.

In this paper, we consider parity array codes, which divide each disk into stripe units (or strips [5]) all of the same size. A collection of n stripe units from distinct disks, each having the same offset, is called a stripe [5]. Individual stripes are a self-contained m-erasure correcting unit. The disk layout is just the cyclic repetition of a stripe, so we will typically focus only on a single stripe when designing a code. Each stripe unit is further divided into packets of a fixed size [5]. Each packet either stores data or parity information, which will be called a data packet or parity packet, respectively. Each packet is organized into overlapping parity groups.

We say that a parity group i appears in a packet (data or parity) if that packet has a non-trivial intersection with i. We say that i appears in a disk if it appears in some packet in that disk. If i does not appear in a disk, then it is called a hole of the disk.

The purpose of parity array codes is to combine the most favorable attributes of RS codes and binary linear codes. Parity array codes possess the XOR-based architecture, while packing the data and parity information into fewer disks.

In Fig. 1 we give an example of a 7-disk parity array code, in which we decompose a given packet into one of the following.

1. Six parity packets P_i in which parity group i appears.
2. Fifteen data packets D_{i,j} in which parity groups i and j appear.

For reasons of cost, performance and simplicity, we focus on m-erasure parity array codes that satisfy the following properties:

1. They are inherently XOR-based, allowing for efficient practical computation.
2. They are MDS codes, minimizing redundancy and therefore minimizing storage space costs.
3. They have minimum update penalty in terms of both the number of disk operations and the computational complexity. Therefore, each data packet should participate in exactly m parity groups.
4. They have a minimum number of packets for given parameters n and m. We will show that an m-erasure code with n disks is composed of either N = n(n−1)/m
We call these properties the performance properties. Following the terminology of [6], we call a code lowest-density if it satisfies the first three properties. However, our definition does not confine the codes to be $F_2$-linear, unlike [6], although this is not an undesirable property. We call a code satisfying all four of the performance properties longest lowest-density. Despite the tautology, we will write “lowest-density MDS codes” and “longest lowest-density MDS codes” to emphasize that they are a special type of MDS code.

For example, EVENODD [7], RDP [8] and their multi-erasure extensions [9], [10] satisfy the first two performance properties; X-Code [11] satisfies the first three performance properties and, in the next section, we will show that B-Code [12] satisfies all four the performance properties. Therefore, a B-Code is a 2-erasure longest lowest-density MDS code. Fig. 1 gives an example of a 7-disk B-Code. A horizontal code contains only disks that consist entirely of data or entirely of parity information [13]. On the other hand, the disks in a vertical code have some data and parity packets stored within a single disk [13]. EVENODD, RDP and their multi-erasure extensions are examples of horizontal codes. B-Code and X-Code are examples of vertical codes.

Some 2-erasure horizontal codes, such as EVENODD and RDP, have been generalized to multiple-erasure successfully. However it is difficult to generalize 2-erasure vertical codes such as B-Code. Using combinatorial means, we will construct 3-erasure longest lowest-density MDS codes from B-Codes, which we call T-Codes. We also describe how to construct a family of T-Codes, that is likely to be infinite, using a combinatorial method. This family of T-Code is in fact equivalent to the codes presented in [14] although we discovered it independently of [14]. However, our contribution lies in a new combinatorial construction method which might provide some helpful insights to further research into finding longest lowest-density MDS codes with even larger distances.

Due to the XOR-based architecture of parity array codes, every bit within a packet may be treated independently. Thus, for our purposes, packet size is not important.

### II. From B-Codes To T-Codes

#### A. B-Code Review

An $n$-disk B-Code, for odd $n$, is identified with an $(n - 1)/2 \times n$ array, as described in [12]. We give an example of this array for $n = 7$ in Fig. 1. One disk contains only data packets and every other disk contains exactly one parity packet and $(n - 1)/2 - 1$ data packets. Disks that only contain data packets are called pure data disks. Deleting the pure data disk in Fig. 1 produces a 2-erasure longest lowest-density MDS code with $n - 1$ disks.

Xu et al. [12] established a bijection between B-Codes and so-called perfect one-factorizations (P1Fs) of complete graphs. A one-factor of a graph $G$ is a 1-regular spanning subgraph of $G$, i.e., a subgraph such that each vertex is adjacent to precisely one other vertex. A one-factorization of $G$ is a decomposition of $G$ into one-factors. A one-factorization is called perfect if the union any pair of distinct one-factors is a Hamiltonian cycle. A complete graph, denoted $K_n$, is a graph on $n$ vertices such that every pair of vertices are adjacent. Given a P1F of $K_{2n}$, it is possible to construct a B-Code with $2n - 1$ disks. A B-Code with $2n - 2$ disk can then be found by deleting the pure data disk, as mentioned above. We will later explicitly describe the construction of a B-Code with $2n - 1$ disks from a P1F.

There is a conjecture (see e.g. [15]) that every complete graph $K_{2n}$ on $2n$ vertices admits a P1F. Should this conjecture be true, it would imply that B-Codes exist for any number of disks. Several infinite families and individual constructions of P1Fs of complete graphs $K_{2n}$ have been identified [16], [17]. Therefore, in many cases we can construct B-Codes from P1Fs.

We will now describe a simple construction of a P1F $\{F_1, F_2, \ldots, F_7\}$ of $K_8$ with vertices labeled 0, 1, ..., 7. We choose the one-factor $F_1 = \{01, 27, 36, 45\}$, which we will call the starter. We generate the remaining one-factors by the cyclic automorphism $\alpha = (0)(1234567)$, i.e., we define $F_{\alpha^k(i)} = \{\alpha^k(i)\alpha^k(j) : ij \in F_1\}$ for all $1 \leq k \leq 6$. Fig. 2 depicts the starter in this construction; the remaining one-factors are formed by rotating Fig. 2 about the vertex 0. The one-factorization formed in this way is tabulated in Fig. 3, with each column identifying a one-factor. It is straightforward to verify that this one-factorization is indeed a P1F. In fact, this method produces a P1F of $K_{2n}$ if and only if $2n - 1$ is prime. This family of P1Fs is named $G_{2n}$ [17].

We note that $G_{2n}$ could be tabulated in numerous ways. For example, in Fig. 3 if (a) we permute the columns or (b) we permute the cells within a column, we will still have a table that describes the perfect one-factorization $G_{2n}$. However, the choice of table will be unimportant for our purposes.

Given a P1F of $K_{2n}$, we can construct a B-Code with $2n - 1$ disks in the following way. Each one-factor $F$ of the P1F corresponds to a unique disk $d$. To construct $d$ from $F$, we...
delete from $F$ the edge that has $2n - 1$ as an endpoint. We then replace the edge $0j \in F$ (if we haven’t deleted it), by the parity packet $P_j$. Finally, the remaining edges $ij \in F$ are replaced by the data packet $D_{ij}$. This process is illustrated in Fig. 4 and gives rise to the B-Code in Fig. 1. Xu et al. [12] have shown that the codes produced in this way are 2-erasures.

B. Structure Properties of Longest Lowest-Density MDS Codes

Xu et al. [12] proved that B-Codes satisfy the first three performance properties. We will now show that they also satisfy the last performance property. We will also show that a B-Code is composed of either $N = n(n-1)/m$ or $N = n^2/m$ packets. Upon inspection of Fig. 4, we can see that the layout of this particular B-Code satisfies:

I. There is at most one pure data disk.

II. The packets from the same disk do not share parity groups.

In general, we call these properties the structure properties. We will show that not only B-Codes, but also all longest lowest-density MDS codes satisfy these properties.

Theorem 1: Longest lowest-density MDS codes satisfy structure property II.

Proof: Suppose that an $m$-erasure correcting code.

1) $C$ is an $m$-erasure correcting code.

2) If we know the value of every segment $\{X_i\}_{i \in S}$ for some $S \subseteq \{1, 2, \ldots, n\}$ with $|S| \geq n-m$, i.e., if no more than $m$ erasures occur, then it is possible to find a unique solution to the system of Equations (2).

3) For any subset $S \subseteq \{1, 2, \ldots, n\}$ with $|S| = m$, the columns of the set of matrices $\{M_i\}_{i \in S}$ are linearly independent over GF(2). Equivalently, any non-empty XOR sum of columns of $H_S$ for any $S \subseteq \{1, 2, \ldots, n\}$ is non-zero.

4) For any $S \subseteq \{1, 2, \ldots, n\}$ with $|S| = m$, the matrix $H_S$ has full rank.

Fig. 5 displays the submatrix $H_{\{0,1\}}$, corresponding to disk0 and disk1 in the B-Code shown in Fig. 1.

Suppose that a disk $d$ is composed of $a_d$ data packets and $b_d$ parity packets. Let $x_d$ be the total number of parity groups that appear in $d$. Then $x_d \leq ma_d + b_d \leq m(p/m) = p$. There are $p = mk$ parity groups and $k$ packets in each disk. Therefore, we have $x_d = p$ if and only if $b_d = 0$, $a_d = k = p/m$, and each packet within the disk is disjoint.

If there is a parity group that does not appear in $d$, which occurs if $b > 0$, then the submatrix $M_i$ contains a row of zeroes. For example, the parity group 2 does not appear in disk0 in the B-Code shown in Fig. 1, thus the second row in its submatrix $M_0$ is a row of zeroes. Recall that, if a parity group does not appear in a disk, we call it a hole. We define the total number of holes in $C$ is the sum of the number of holes of each disk.

Claim: The total number of holes in $C$ is at most $p(m-1)$. Otherwise, for some parity group $i$, there are at least $m$ holes, by the pigeonhole principle. Therefore, we can find $S \subseteq \{1, 2, \ldots, n\}$ of cardinality $m$ for which each $M_i$ in $Y$ has $i$ as a hole. However, in this case, row $i$ of $H_S$ consists entirely of zeroes, implying it does not have full rank, contradiction condition 4.

Claim: The total number of holes in $C$ is at least $p(m-1)$. We know $x_d \leq ma_d + b_d = m(a_d + b_d) - b_d(m-1)$. Therefore the total number of holes in $C$ is $\sum d(p - x_d) \geq np - mN + p(m - 1) = p(m - 1)$ by (1).

To review, we showed that there are exactly $p(m-1)$ holes in $C$. They all arise due to the existence of a parity packet within a disk. Therefore, the packets from the same disk do not share parity groups.

We will now use Theorem 1 to deduce that longest lowest-density MDS codes also satisfy structure property I.

Corollary 2: Longest lowest-density MDS codes $C$ satisfy structure property I.

Proof: Theorem 1 implies that $C$ satisfies structure property II. Therefore if a disk $d$ is a pure data disk, then every
parity group appears in $d$. Suppose, seeking a contradiction, that there are two pure data disks in $C$. If we erase both of those disks, then that data cannot be uniquely recovered. Hence the code $C$ is not an $m$-erasure correcting code. In fact, $C$ is not even a 2-erasure correcting code. We conclude that $C$ contains no more than one pure data disk.

Since we seek minimum number of packets, each disk contains at most one parity packet. Hence $p$ is either $n-1$ or $n$. From Equation (1), we can deduce that the total number of packets in $C$ is either $N = n(n-1)/m$ or $N = n^2/m$.

The result in this section agree with [6]. However, our derivation is based on a combinatorial method, thus not requiring that the codes are $F_2$-linear. We have identified some structural properties of longest lowest-density MDS codes. Moreover, we can now derive some necessary conditions for the existence of a longest lowest-density MDS code. The remainder of this paper is devoted to constructing an infinite family of T-Codes.

### III. T-Codes and NRBs

We call a longest lowest-density MDS code a $T$-Code if it is a 3-erasure code. In this section, following the methodology used in [12], we will construct a bijection between T-Codes and a certain type of so-called perfect near-resolvable block designs. Since each disk contains $k = p/m$ packets and $m = 3$ now, $p$ must be divisible by 3 and therefore $p = 3k$ and $n = p = 3k$ or $n = p + 1 = 3k + 1$. We will consider the case when $n = 3k + 1$.

Fig. 6 displays a 13-disk T-Code. We observe that (a) it satisfies all the structure properties, (b) packets from the same disk do not share parity groups, (c) there is one pure data disk $d$ and (d) every disk other than $d$ contains a parity packet and has precisely two holes. We introduce a new parity group $P_0$ into the code in the following way. To each disk $d$ that contains a parity packet, we add a new data packet consisting of the parity group $P_0$ and the two parity groups originally missing from $d$. To the unique pure data disk, we instead add $P_0$. This completes the construction, which we will call the complement transformation, denoted $T$. In Fig. 7 we give the complement transformation $T$ of the T-Code in Fig. 6.

We see that each pair of distinct parity groups, $i$ and $j$, simultaneously appear in exactly two data packets in $T$. Taking the structure properties into account, this system is a so-called near-resolvable design [17] with parameters $(13,3,2)$.

A balanced incomplete block design, abbreviated BIBD($v,m,\lambda$), is a pair $(V,B)$, where $V$ is a set of cardinality $v$ and $B$ is a collection of subsets of $V$ each of cardinality $m$ such that $|B|=b$ and (a) every element $x \in V$ appears in exactly $r$ blocks and (b) every pair of distinct elements $x,y \in V$ appears simultaneously in $\lambda$ blocks. Every set in $B$ is called a block. It is necessary that the $r = \lambda(v-1)/(m-1)$ and $b = vr/m$, so these parameters are uniquely determined by $v, m$ and $\lambda$.

A near-resolvable (block) design, abbreviated NRB($v,m,\lambda$), is a balanced incomplete block design BIBD($v,m,\lambda$) with the additional property that $B$ can be partitioned into near parallel classes, i.e., there exists a partition $Q$ of $B$, such that (a) each part $q \in Q$ has cardinality $|q|=m$, (b) each element $x \in V$ appears in at most one part $q \in Q$ and (c) for each $x \in V$ there exists a unique $q \in Q$ with $x \notin Q$.

We can construct an NRB($v,3,2$) $B$ from the complement transformation $T$ of a T-Code $C$ in the following way. The parity groups form $V$ and each data packet $t$ identifies a block in $B$ corresponding to the parity groups that appear in $t$. The partition $Q$ is formed by the set of data packets in each disk. The missing element in each $q \in Q$ corresponds to the parity group of the parity packet.

We can reverse this construction by performing the following steps on each near parallel class $q \in Q$ to construct a disk.

1) We convert the missing element into the corresponding parity packet.

2) We convert each block into a data packet – the three elements in the block designate the parity groups which appear in the data packet.

We then pick a parity group $e$ and delete every packet (both data and parity) in which $e$ appears. We call this the elimination transformation. This raises a question: is the code constructed from an NRB($v,3,2$) necessarily a 3-erasure code? Actually, it turns out not to be the case, therefore we will need to introduce a stronger property.

Let $V$ be a set. Suppose $G$ is a collection of subsets (blocks) of $V$, and $S = \cup G$, we say that $G$ is a cover of $S$ and that $S$ is the base set of $G$, denoted by $S = \text{BASE}(G)$. We say $|G| := |S|$ is the size of the cover $G$. If each element of $S$ appears in an even number of blocks in $G$, then we call $G$ an even cover of $S$. Define $MC(G) = \min_{G'}(|G'|)$ where the minimum is taken over all even covers $G' \subseteq \text{BASE}(G)$. We call $MC(G)$ the minimum even cover size. If $G$ does not possess a subset $G'$ that is a even cover of $S$, then we use the convention $MC(G) = \infty$.

Given a block $g$ of an NRB($3k+1,3,2$), it is contained in a unique near-parallel class $q \in Q$. If $x \in V$ and $x \notin \cup g$ then $P := q \cup \{x\}$ is a partition of $V$ into $k$ parts of cardinality 3 (the original blocks of $q$) and one part of cardinality 1 (i.e. $\{x\}$). We call $P$ the complement block of $g$. We call an NRB($3k+1,3,2$) perfect, if for any three complement

---

**Fig. 6.** A 13-disk T-Code.

**Fig. 7.** T-Code—NRB.
blocks $B_1$, $B_2$ and $B_3$, we have $MC(B_1 \cup B_2 \cup B_3) = |V| = 3k + 1$. That is, every three complement blocks $B_1$, $B_2$ and $B_3$ in a perfect NRB($3k + 1, 3, 2$) are such that every even cover $G'$ of $B_1 \cup B_2 \cup B_3$ must have size $|G'| = |V|$. Observe that the union $B_1 \cup B_2$ of any pair of complement blocks of an NRB($3k + 1, 3, 2$) is an even cover of $V$, thus $MC(B) \leq |V| = 3k + 1$. Therefore, a perfect NRB($3k + 1, 3, 2$) maximizes the minimum even cover size of every $B_1 \cup B_2 \cup B_3$. We will now identify a bijection between T-Codes and perfect NRB($3k + 1, 3, 2$).

We can associate any subset $G \subseteq B$ of an NRB($3k + 1, 3, 2$) with a $(0, 1)$-matrix $H_G$ in the following way. Each row corresponds to an element $x \in V$ and each column corresponds to a block $g \in G$. The cell in row $x \in V$ and column $g \in G$ contains $1$ if $x \in g$, otherwise it contains $0$. The matrix $H_G$ is a submatrix of parity check matrix $H$ of the code produced by the elimination transformation on $B$.

**Lemma 3:** $MC(G) = \infty$ for every $G \subseteq B$ if and only if $H_G$ has full rank.

**Proof:** If $MC(G) < \infty$, then there exists a non-empty subset $G' \subseteq G$ that is an even cover. Therefore the XOR sum of the columns of $H_{G'}$ is the zero vector, implying that $H_{G'}$ and hence $H_G$ does not have full rank. Conversely, if $H_G$ does not have full rank, then there exists some cover $G' \subseteq G$, for which the XOR sum of the columns of $H_{G'}$ is the zero vector. Moreover, $G'$ must be an even cover, otherwise for some $x \in BASE(G')$, the row of $H_{G'}$ corresponding to $x$ contains an odd number of $1$ entries, giving a contradiction.

**Lemma 4:** Performing complement transformation on a T-Code $C$ with $n = 3k + 1$ disks produces a system $T$ in which each pair of distinct parity groups, $i$ and $j$, appears in precisely two pairs of data packets of $T$.

**Proof:** Observe that $T$ contains $n(n-1)/3$ data packets. Precisely three parity groups appear in each data packet. There are $(\binom{n}{2}) = n(n-1)/2$ pairs of distinct parity groups. Therefore, each pair of distinct parity groups simultaneously appears in two data packets on average. We will now show that each pair of distinct parity groups simultaneously appears in at most two data packets. Suppose, seeking a contradiction, that there exists a pair of distinct symbols $i$ and $j$ that simultaneously appear in three data packets.

Case I: $i = \infty$ or $j = \infty$. Without loss of generality, suppose that $i = \infty$. Since $j$ must be a hole in $C$ and each hole appears exactly twice in $C$, this pair $(\infty, j)$ appears exactly twice in $T$.

Case II: $i \neq \infty$ and $j \neq \infty$. If both $i$ and $j$ appear in three data packets, then structure property II implies that these data packets appear in distinct disks in $C$. Let $S$ index those disks and so $H_S$ is a square $(0, 1)$-matrix (since $m = 3$) with identical row $i$ and row $j$, by structure property II. Therefore, the rows are not linearly independent and $H_S$ does not have full rank, giving a contradiction.

**Theorem 5:** There is a bijection between T-Codes with $n = 3k + 1$ disks and perfect NRB($3k + 1, 3, 2$).

**Proof:** We have shown that an NRB($3k + 1, 3, 2$) $B$ can be converted into a code $C$ satisfying the structure properties by the elimination transformation. By definition, if $B$ is a perfect NRB($3k + 1, 3, 2$), then any three complement blocks $B_1$, $B_2$ and $B_3$ in $B$ have $MC(B_1 \cup B_2 \cup B_3) = |V| = 3k + 1$. During the elimination transformation we have deleted the set containing $e$ from each of $B_1$, $B_2$ and $B_3$ to form $B_1^*$, $B_2^*$ and $B_3^*$, respectively. Therefore, if $G'$ is an even cover and $G' \subseteq B_1^* \cup B_2^* \cup B_3^*$, then $|BASE(G')| \leq |V| - 1$. Moreover, $G' \subseteq B_1 \cup B_2 \cup B_3$, contradicting that $B$ is perfect. Hence $MC(B_1^* \cup B_2^* \cup B_3^*) = \infty$ and Lemma 3 implies that $C$ is a T-Code.

Conversely, according to Lemma 4, performing complement transformation on a T-Code $C$ with $3k + 1$ disks yields an NRB($3k + 1, 3, 2$) $B$. If $B$ is not perfect, there must be three complement blocks $B_1$, $B_2$ and $B_3$ with $MC(B_1 \cup B_2 \cup B_3) < |V| = 3k + 1$. So there is an even cover $R \subseteq B_1 \cup B_2 \cup B_3$ of size less than $|V|$. Since $R$ is an even cover, any element that is in some set in $R$ is in exactly two sets in $R$. If $\infty$ is not in some set in $R$, then $R$ is fully contained in $C$. Thus some columns selected form $C$'s parity check matrix are linear dependent, implying that $C$ is not a 3-erasur code, giving a contradiction. Thus $\infty$ is in exactly two sets in $R$. Without loss of generality, assume $\infty$ appears in $B_1$ and $B_2$. Let

$$B_1^R = R \cap B_1 \quad B_2^R = R \cap B_2 \quad B_3^R = R \cap B_3$$

Since each element is in exactly two sets in $R$ and $B_1$, $B_2$ and $B_3$ are mutually disjoint, we have

$$BASE(B_1^R) \cap BASE(B_2^R) \cap BASE(B_3^R) = \emptyset.$$
modulo \( p \) generated by 2. Since 2 is a primitive root modulo \( p \), we know that \( 2^{3k} \equiv 1 \pmod{p} \). We define the near parallel classes by
\[
\left\{(2^i + j, 2^{i+k} + j, 2^{i+2k} + j) | 0 \leq i \leq k-1\right\}_{0 \leq j \leq p-1}
\]
working modulo \( p \). For example, when \( p = 13 \) and \( k = 4 \), we construct the near parallel classes
\[
\begin{align*}
  j = 0 & \quad \{ \{1, 3, 9\}, \{2, 5, 6\}, \{4, 10, 12\}, \{7, 8, 11\} \} \\
  j = 1 & \quad \{ \{2, 4, 10\}, \{3, 6, 7\}, \{0, 5, 11\}, \{8, 9, 12\} \} \\
  j = 2 & \quad \{ \{3, 5, 11\}, \{4, 7, 8\}, \{1, 6, 12\}, \{0, 9, 10\} \} \\
  j = 3 & \quad \{ \{4, 6, 12\}, \{5, 8, 9\}, \{0, 2, 7\}, \{1, 10, 11\} \}
\end{align*}
\]
and so on, up to \( j = p-1 \). The NRB \((p, 3, 2)\), denoted \( B_p \), is the union of the parallel classes, which we will call a 2-NRB, since it is generated by a primitive root 2. We know that \( B_p \) can be converted into a code \( C \) by the elimination transformation. We will now show that \( C \), constructed from \( B_p \), is a T-Code, so Theorem 5 verifies that \( B_p \) is indeed an perfect NRB \((v, 3, 2)\). Note that, in the elimination transformation, the blocks containing 0 are deleted. It is straightforward to show that \( C \) satisfies the performance properties. Therefore, it is sufficient to prove that \( C \) is a 3-erasure code.

For this, we will identify \( C \) with a polynomial of the quotient ring \( R \) of polynomials in \( GF(2)[x] \) modulo the ideal generated by
\[
f_p(x) := 1 + x + x^2 + \cdots + x^{p-1}.
\]
Observe that
\[
x^p - 1 = (x - 1)(1 + x + x^2 + \cdots + x^{p-1}) = (x - 1)f_p(x).
\]
Therefore \( x^p = 1 \) in \( R \). An arbitrary polynomial \( g \in R \) can be written in the form
\[
g(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{p-1} x^{p-1},
\]
where \( a_0, a_1, \ldots, a_{p-1} \in GF(2) \). Observe that \( x^t g(x) \), for any integer \( t \), is the polynomial formed by cyclicly shifting coefficients \( a_0, a_1, \ldots, a_{p-1} \) in (3) of \( g(x) \) to the right by \( t \) steps.

For any \( 0 \leq i \leq k-1 \), let
\[
g_i(x) = x^{2^i} + x^{2^{i+k}} + x^{2^{i+2k}}.
\]
Let \( \mathcal{V} = GF(2)^{k+1} \) be the direct product of \( k+1 \) \( GF(2) \)'s. For arbitrary \( v \in \mathcal{V} \) we can write \( v = (a_0, a_1, \ldots, a_k) \) for some \( a_0, a_1, \ldots, a_k \in GF(2) \). We define \( h_v(x) \in R \) to be the polynomial
\[
h_v(x) = a_0 + \sum_{i=0}^{k-1} a_{i+1} g_i(x)
\]
for all \( v \in \mathcal{V} \).

For all \( 0 \leq j \leq p-1 \), we let \( B_j \) denote the parallel class in which \( j \) is not an element of any of the blocks. We assign an total order to the blocks in each near parallel class \( B_j \in B_p \) so that we can consider the \( i \)-th block of \( B_j \), where \( 0 \leq i \leq k-1 \). Given a near parallel class \( B_j \in B_p \) which produces the \( j \)-th disk of \( C \), we construct a vector \( v_j = (a_0, a_1, \ldots, a_k) \in \mathcal{V} \) by making the following assignments.

- \( a_0 \) is the value of the parity packet \( P_j \) if \( P_j \) appears in disk \( j \) and \( a_0 = 0 \) otherwise.
- \( a_{i+1} \) is the value of the packet defined by the \( i \)-th block of \( B_j \) if the data packet appears in disk \( j \) and \( a_{i+1} = 0 \) otherwise, for each \( 0 \leq i \leq k-1 \).

Hence the \( j \)-th disk of \( C \) gives rise to the polynomial \( x^j h_v(x) \in R \). Observe that \( x^j \) corresponds to the \( i \)-th parity group of \( C \), and the coefficient of \( x^j \) is the value of the packet in which the \( i \)-th parity group appears. Therefore adding the coefficients of \( x^j \) from different disks over \( GF(2) \) is equivalent to taking the XOR sum of the packets that intersect with the \( i \)-th parity group. So we have
\[
\sum_{j=0}^{p-1} x^j h_v(x) = 0.
\]

**Lemma 6:** For all \( 0 \leq i \leq k-1 \), \( g_i(x) = g_i(x)^{2^k} = g_i(x)^{2k} \).

**Proof:** To begin, observe that, for any \( a, b, c \),
\[
(x^a + x^b + x^c)^2 = x^{2a} + x^{2b} + x^{2c} + 2(x^{a+b} + x^{a+c} + x^{b+c}) = x^{2a} + x^{2b} + x^{2c},
\]
since coefficients are in \( GF(2) \). By repeated application we find that
\[
(x^a + x^b + x^c)^{2^k} = x^{2^k a} + x^{2^k b} + x^{2^k c}.
\]
In our case \( a = 2^i \), \( b = 2^{i+k} \) and \( c = 2^{i+2k} \). Hence
\[
g_i(x)^{2^k} = x^{2^k 2^i} + x^{2^k 2^{i+k}} + x^{2^k 2^{i+2k}} = x^{2^{i+k}} + x^{2^{i+2k}} + (x^{2^{3k}})^{2^i} = x^{2^{i+k}} + x^{2^{i+2k}} + x^{2^i} = g_i(x)
\]
since \( 2^{3k} \equiv 1 \pmod{p} \) and \( x^p = 1 \) in \( R \). The second equality follows from the first.

**Theorem 7:** Equation (4) admits a solution whichever three disks of \( C \) fail.

**Proof:** Without loss of generality, suppose that the 0-th disk, the \( a \)-th disk and the \( b \)-th disk fail, \( 1 \leq a < b \leq p-1 \). Let \( H(x) \) be the sum of the polynomials corresponding to the surviving disks. Equation (4) can be rewritten as
\[
H(x) = h_{v_0}(x) + x^a h_{v_a}(x) + x^b h_{v_b}(x).
\]

**Lemma 6** implies that
\[
\begin{align*}
H(x) &= h_{v_0}(x) + x^a h_{v_a}(x) + x^b h_{v_b}(x) \\
H(x)^{2^k} &= h_{v_0}(x) + x^{2^k a} h_{v_a}(x) + x^{2^k b} h_{v_b}(x) \\
H(x)^{2^{2k}} &= h_{v_0}(x) + x^{2^{2k} a} h_{v_a}(x) + x^{2^{2k} b} h_{v_b}(x).
\end{align*}
\]
Let
\[
\nu = \begin{pmatrix}
1 & x^a & x^b \\
1 & x^{2^k a} & x^{2^k b} \\
1 & x^{2^{2k} a} & x^{2^{2k} b}
\end{pmatrix}.
\]
The system of equations (6) admits a solution if and only if \( \det(\nu) \neq 0 \).

\[
\det(\nu) = x^a x^b \begin{vmatrix}
1 & 1 & 1 \\
x^{a(2^k-1)} & x^{b(2^k-1)} & 1 \\
x^{a(2^{2k}-1)} & x^{b(2^{2k}-1)} & 1
\end{vmatrix}
\]
Let \( c = a(2^k - 1) \) and \( d = b(2^k - 1) \). Since \( a \neq b \) and \( p \) is a prime, \( c \neq d \mod p \).

\[
\det(\nu) = x^a \cdot x^b = x^c \cdot x^d
\]

\[
\begin{vmatrix}
1 & 1 & 1 \\
1 & x^c & x^d \\
1 & x^{c(2^k+1)} & x^{d(2^k+1)}
\end{vmatrix}
\]

\[
= x^a \cdot x^b \cdot (x^c - 1) \cdot (x^d - 1) - (x^d - 1) \cdot (x^c)^{2^k+1} \cdot (x^d - 1) \cdot (x^c)^{2^k+1} \cdot (x^d - 1)
\]

\[
= x^a \cdot x^b \cdot (x^c - 1) \cdot (x^d - 1) \cdot \left( \sum_{i=0}^{2^k} x^{di} - \sum_{i=0}^{2^k} x^{ei} \right)
\]

Since the two monomials and the two binomials are all reversible [9], it follows that \( \det(\nu) \) is identically 0 if and only if

\[
\sum_{i=0}^{2^k} x^{di} = \sum_{i=0}^{2^k} x^{ei}.
\] \hspace{1cm} (7)

For each \( 0 \leq s \leq p - 1 \) define \( \ell_s \in R \) to be the polynomial

\[
\ell_s(x) = \sum_{i=0}^{\ell_s} x^i.
\]

By (7), \( \det(\nu) = 0 \) if and only if \( \ell_s = \ell_d \)

for some \( c \neq d \mod p \) and \( c, d \neq 0 \mod p \). Suppose \( d = c + r \mod p \) where \( r \neq 0 \mod p \). Then \( \ell_c = \ell_{c+r} = \ell_{c+2r} = \ell_{c+3r} \) and so on. Since \( r \neq 0 \mod p \) and \( p \) is prime, we eventually find that \( \ell_c = \ell_0 = 0 \) since there are an odd number of terms in (7). However, by (3), if \( \ell_1 = \ell_0 = 0 \), then the number of terms in \( \ell_1 \) is \( 2^k + 1 \equiv 1 \mod p \), giving a contradiction.

At first sight, \( v_j \mapsto x^j \cdot h_{u_j}(x) \) is not an injection. For example, if \( v_j = (0, 0, \ldots, 0) \in \mathcal{V} \) and \( v_j' = (1, 1, \ldots, 1) \in \mathcal{V} \), then \( x^j \cdot h_{u_j}(x) + x^j \cdot h_{u_j'}(x) = 1 + x + x^2 + \cdots + x^{p-1} = f_p(x) \).

However, in the process of constructing the \( j \)-th disk \( d \) of \( \mathcal{C} \) from \( B_p \), if \( j \neq 0 \) then we have deleted some block \((0, u, v)\) and inserted the \( j \)-th parity packet \( F_j \). Hence, the coefficients of \( x^{u+j} \) and \( x^{v+j} \) in \( x^j \cdot h_{u_j} \) are both zero, which implies that \( x^{u+j} \) and \( x^{v+j} \) should not appear. For \( j = 0 \), while no block has been deleted, there is no parity packet in this disk, so \( x^0 \) should not appear. Therefore for the \( v \in \mathcal{V} \) that we are interested in, this situation cannot arise.

Theorem 7 shows that the codes \( \mathcal{C} \), formed from \( B_p \) using the elimination transformation, are indeed 3-erasure codes, i.e. they are T-Codes.

To review, in this section we presented a combinatorial construction of a family of T-Codes that is likely to be infinite. However, the methodology presented is unable to resolve every case of \( n = 1 \mod 3 \). For other cases, inspecting all possible layouts, even with the aid of a computer, is not feasible. A more promising way is to check some special layouts that are likely to be T-Codes.

We have proved that 2-NRBs are perfect NRBs. An intuitive idea is to test NRBs based on Galois fields GF\( (p) \) that have a primitive root 3, 5, and so on. We have tested all prime of the form \( p = 6c + 1 \) less than 1000 and the results are tabulated in TABLE I. We were unable to find T-Codes for all primes of this form, however, we have succeeded in resolving many cases. For 4 disks, there is a simple construction of a T-Code – four-way mirroring. For \( p = 7 \) we performed an exhaustive search to prove that no T-Code exists with 7 disks. Therefore perfect NRBs and T-Codes do not exist for all \( n \) of the form \( 3k + 1 \). For \( n = 10 \), we were unable to exhaustively search every possible layout.

\section*{V. Conclusion}

In this paper, we studied lowest lowest-density MDS codes, a simple class of multi-erasure parity array codes with good performance. We proved some structure properties of this kind of codes. We defined a property, which we called “perfect”, for near-resolvable block designs (NRBs) and found a bijection between T-Codes, i.e., 3-erasure lowest lowest-density MDS codes, and perfect NRB\( (3k + 1, 3, 2) \). We also presented a combinatorial construction of a family of T-Codes that is likely to be infinite. This family of T-Codes is constructed from NRBs based on prime fields that have a primitive root 2. We proved that this kind of NRB is perfect, verifying that the codes we produced are indeed T-Codes. We also tested some other NRBs and found some T-Codes outside of this family.

An important avenue for future exploration would be to find more T-Codes, either in families or individual instances, by both theoretic and computational means. It would also be of value to develop efficient encoding and decoding algorithms for these codes.

\section*{Acknowledgment}

We would like to thank the referees for valuable feedback.

\section*{References}


Sheng Lin received the B.Sc. degree in traffic management engineering from Chinese People’s Public Security University, Beijing, China, in 1996, and the M.Sc. degree in computer science from Nankai University, Tianjin, China, in 2003. He is currently working toward the Ph.D. degree in computer science at Nankai University. His research interests include erasure-correcting codes and combinational algorithms.

Gang Wang received the B.Sc., the M.Sc. and the Ph.D. degrees in computer science from Nankai University, Tianjin, China, in 1996, 1999 and 2002, respectively. He is currently an associate professor at the College of Information Technical Science, Nankai University. Gang’s research interests include storage systems and parallel computing.

Douglas S. Stones received his B.Sc. in mathematics from Monash University, Melbourne, Australia, in 2005. He is currently a Ph.D. candidate in combinatorial mathematics, also at Monash University, under the primary supervision of Ian M. Wanless. In his thesis he finds new congruences satisfied by the number of Latin squares. It employs a range of techniques from combinatorics, number theory and group theory and also gives new results on other important aspects of Latin squares, for example sub-squares and autotopisms. He is expected to submit his thesis late 2009.

Xiaoguang Liu received the B.Sc. degree, M.Sc. degree and Ph.D. degree in computer science from Nankai University, Tianjin, China, in 1996, 1999 and 2002 respectively. He is currently an associate professor in computer science at Nankai University, Tianjin, China. His research interests include parallel computing and storage system.

Jing Liu received his M.Sc. degree in computer science from Nankai University in 1982. Since then, he worked at Nankai University. He is currently a professor and Ph.D supervisor in computer science at Nankai University, Tianjin, China. His current research interests include algorithm design and parallel computing.