Context-Aware Resiliency: Unequal Message Protection for Random-Access Memories

Clayton Schoeny, Frederic Sala, Mark Gottscho, Irina Alam, Puneet Gupta, and Lara Dolecek
Email: {cschoeny, fredsala, mgottscho, irina1}@ucla.edu, {puneet, dolecek}@ee.ucla.edu
University of California, Los Angeles (UCLA), CA 90095

***Abstract*—A common way to protect data stored in DRAM and related memory systems is through the use of a single-error-correcting/double-error-detecting (SECDED) code. Traditionally, these error-correcting codes provide equal protection guarantees to all messages. In a recent work, we demonstrated enhanced error correction capabilities for SECDED codes by taking into account contextual side-information about the data. This paper is concerned with a closely related scenario: *unequal message protection* (UMP), where a subset of special messages is afforded additional error-correction ability. UMP is relevant to computing systems where certain messages are critical and failures cannot be tolerated. We study practical UMP constructions where messages are guaranteed either one or two bit-error-correction. We provide upper and lower bounds on the number of special messages. We introduce an explicit and practical code construction based on BCH subcodes and demonstrate the efficacy of our technique on data from the AxBench and SPEC CPU2006 benchmark suites.**

I. Introduction

Error-correcting codes (ECCs) play a critical role in memory resiliency. Traditionally, the metric of interest is the minimum distance of a code, which provides guarantees on error-correction and error-detection capabilities. Intriguingly, *side-information* about the underlying data and communication channels can be used to enhance such a system, extending the traditional notions from classical coding theory.

Recently, we proposed *Software-Defined Error-Correcting Codes* (SDECC), a class of heuristic techniques to recover from detected but uncorrectable errors (DUEs) [1]. SDECC can be considered as a highly practical list-decoding framework that utilizes any linear code capable of correcting t errors and detecting $t+1$ errors. Traditionally, when a DUE occurs, the system will either crash or restore to a checkpoint. In our SDECC framework, however, when a DUE occurs, we first compute a list of *candidate codewords*—the closest neighboring codewords—and then probabilistically decode based on available side-information. SDECC is applicable to a wide variety of memory applications and systems ranging from large-scale servers in data centers to embedded systems in Internet-of-Things devices.

In this work, we focus on the encoding-side of SDECC: instead of using side-information to heuristically decode, we a priori designate specific messages to have extra protection against errors. This *unequal message protection* (UMP) is fundamentally different from unequal error protection (UEP) [2], in which extra protection is provided for specific bit positions or certain error patterns (such as adjacent bit errors) among all codewords. UMP is a powerful approach when the special messages are chosen based on knowledge of data patterns. It is particularly useful when compression is not possible yet specific messages—or parts of specific messages—are very frequently stored/transmitted. Additionally, given side-information about the meaning of underlying messages, we can add extra protection to those messages whose miscorrections would be very costly at a system level.

Due to its frequent application in computer memories such as DRAM and SRAM [3], [4], we take particular interest in the extended Hamming code. This simple code is capable of correcting any single bit error and detecting any double bit error. Our goal is to guarantee single bit error correction for every codeword while maximizing the number of *special codewords* that are also correctable in the case of a double bit error. In the SDECC framework, special codewords can be viewed as a set with the property that no two elements from the set are ever in the same candidate list.

The paper organization is as follows. In the remainder of this section, we provide preliminaries, notation, objectives, and an overview of related work. In Section II, we derive an upper bound on the number of special messages using a nonlinear program and compare it to a sphere packing upper bound. In Section III, we provide two lower bounds on the maximal number of special messages; the first using a maximal independent set argument from graph theory, and the second using properties of sub-codes. We develop an explicit code construction in Section IV, using a BCH code as a sub-code—without using additional redundancy— motivated by analysis of memory benchmarks.

A. Preliminaries and Notation

A code $\mathcal{C}$ is a subset of $\{1, 2, \ldots, q\}^n$, where $q \geq 2$ is the alphabet size and n is the code length. We set $M = |\mathcal{C}|$ to be the cardinality of the code. As usual, for linear block codes the parameter k is the code dimension (so that $M = q^k$ messages can be represented). In this work we only consider binary linear codes, i.e., $q = 2$. Code $\mathcal{C}$ has minimum distance d if $d = \min_{\mathbf{x},\mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}} d_H(\mathbf{x}, \mathbf{y})$, where d_H is the Hamming distance. If $\mathcal{C}$ has minimum distance d, it can correct $t = \lfloor (d-1)/2 \rfloor$ errors. We use the standard (n, k, d) notation to denote a code's length, dimension, and minimum distance properties and $d(\mathcal{C})$ as shorthand for the minimum distance of $\mathcal{C}$.

We partition the M codewords into the sets $\mathcal{M}_i$, where the codewords in $\mathcal{M}_i$ have the property that they are guaranteed correctable in the presence of up to (and not more than) i errors. In this work, we focus solely on the partition of the M codewords into the sets $\mathcal{M}_1$ and $\mathcal{M}_2$. Let $M_1 = |\mathcal{M}_1|$ and $M_2 = |\mathcal{M}_2|$. We call such a code a *single-error-correcting/special-message-double-error-correcting* (SEC-smDEC) code. We thus have the following minimum distance properties:

$$\min_{\mathbf{x},\mathbf{y}\in\mathcal{M}_1,\mathbf{x}\neq\mathbf{y}} d_H(\mathbf{x},\mathbf{y}) \geq 3, \tag{1}$$

$$\min_{\mathbf{x}\in\mathcal{M}_1,\mathbf{y}\in\mathcal{M}_2} d_H(\mathbf{x},\mathbf{y}) \geq 4, \tag{2}$$

$$\min_{\mathbf{x},\mathbf{y}\in\mathcal{M}_2,\mathbf{x}\neq\mathbf{y}} d_H(\mathbf{x},\mathbf{y}) \geq 5. \tag{3}$$

For code parameters (n, k), our objective is to fully partition the code into the sets $\mathcal{M}_1$ and $\mathcal{M}_2$ and maximize M_2. That is, we require that every codeword is correctable given a single error, and we seek to maximize the number of codewords that are correctable in the presence of up to two errors. We denote the codewords in $\mathcal{M}_2$ as *special* codewords (and their associated messages as special messages).

Let us examine the code parameters of interest. A $(2^m - 1, 2^m - m - 1, 3)$ Hamming code is a *perfect code* since it achieves the Hamming bound, thus $M_2 = 0$ (any codeword given double error protection properties would necessarily force other codewords to lose their guaranteed correction capabilities). Any code with fewer redundancy bits than the associated Hamming code is not in our region of interest since it cannot be a valid SEC-smDEC code. However, with the addition of a single overall parity-bit, the Hamming code is transformed into the $(2^m, 2^m - m - 1, 4)$ *extended* Hamming code. Extended Hamming codes are routinely used as single-error-correcting/double-error-detecting (SECDED) codes [3], [4]; however, we can give up the DED property to create a SEC-smDEC code with the same level of redundancy. In the following example, we use the same codewords from the $(8, 4)$ extended Hamming code, but we consider the Hamming sphere around the all-1s codeword and the all-0s codeword to have radius 2.

Example 1. *$\mathcal{C} = \mathcal{M}_1 \cup \mathcal{M}_2$ is an $(8, 4)$ SEC-smDEC code.*

$$\begin{aligned}\mathcal{M}_1 = \{&(11100001), (10011001), (01010101), (00101101),\\ &(00110011), (01001011), (10000111), (01111000),\\ &(10110100), (11001100), (11010010), (10101010).\\ &(01100110), (00011110)\}.\\ \mathcal{M}_2 = \{&(00000000), (11111111)\}.\end{aligned}$$

At the other extreme of interest is the $(2^m - 1, 2^m - 2m - 1, 5)$ BCH code, which is trivially a SEC-smDEC code with $M_2 = |\mathcal{C}|$. Therefore, in this work we are interested in the redundancy parameters between (but not including) the Hamming code and the $t = 2$ BCH code; we denote the following as our SEC-smDEC parameters of interest:

$$n - 2\lceil\log(n)\rceil - 1 < k \leq n - \lceil\log(n)\rceil - 1.$$

B. Related Work

The majority of research into UEP codes has focused on *bit-wise* UEP, in which specific positions of a codeword are more robust to errors [2], [5]. Masnick and Wolf [2] created a framework for constructing linear bit-wise UEP codes in which each bit in a codeword is assigned an error protection level. Bit-wise UEP codes are useful when errors in specific bit positions are more severe, e.g., the most significant bit of a binary integer or the destination address header of a packet.

Another type of UEP is *error-wise* UEP, in which specific error patterns are guaranteed to be correctable. Error-wise UEP codes are useful when bit-error locations are not independent. A code that is designed to correct burst errors can be thought of as an error-wise UEP code. For example, *single-error-correction/double-error-detection/double-adjacent-error-correction* (SEC-DED-DAEC) codes guarantee correction in the case of one bit in error or two adjacent bits in error [6], [7]. Error-wise UEP codes are also useful when different sections of the codeword are stored in different chips in computer hardware, in which case a faulty chip only causes errors on a specific subsection of the codeword [8], [9].

In this work, we focus on UMP, i.e., *message-wise* UEP, in which specific messages have extra protection from errors. In this setting, Broade *et al.* used an information-theoretic approach to prove that it is possible to encode many special messages, even at rates approaching the channel capacity [10].

Shkel *et al.* [11] also examine the UMP problem. The main distinction between this work and ours is that they are concerned with producing information-theoretic bounds (achievability and converse) for such codes with average and maximal error probability over a probabilistic channel. Shkel *et al.* follow the line of work considered in Broade *et al.*, but they also look at the finite-length regime by applying the finite blocklength framework from Polyanski *et al.* [12] to the UMP setting. Nevertheless, theirs is a different setting compared to ours: we are interested in adversarial, not probabilistic, errors and we wish to produce explicit non-randomized code constructions.

Lastly, UMP is related to the *red alert problem*, in which a specific message not only requires a small probability of missed detection, but also a small probability of false alarm [13]. In this work, we are not concerned with mitigating false alarms of our special messages.

II. Upper Bounds on Special Messages

In this section, we derive upper bounds on the cardinality of the special message set with two different methods. First, we derive a modified version of the classical sphere-packing bound and then we create a modified version of Delsarte's linear programming bound [14]. Both of these bounds are valid for any parameters (n, k) in our region of interest.

A. Sphere-Packing Bound

The sphere-packing bound (also known as the Hamming bound) is a simple upper bound on the cardinality of a block code. We can rewrite the sphere-packing bound as a function of the radii of the Hamming spheres (as opposed to the minimum distance of the code):

$$|C| \leq \frac{q^n}{\sum_{j:\mathcal{M}_j\neq\{\emptyset\}} M_j \sum_{i=0}^{j} \binom{n}{i}(q-1)^i}.$$

For our purposes, we fix n and k, and we wish to maximize M_2. We use the same idea as in the classic sphere-packing bound, but with differently-sized balls $\mathcal{B}_1$ and $\mathcal{B}_2$, corresponding to message sets $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. In the binary setting we solve for M_2 as follows:

$$\begin{aligned} M_1|\mathcal{B}_1| + M_2|\mathcal{B}_2| &\leq 2^n \\ \implies (2^k - M_2)\sum_{j=0}^{1}\binom{n}{j} + M_2\sum_{j=0}^{2}\binom{n}{j} &\leq 2^n \\ \implies M_2 &\leq \frac{2^n - 2^k(n+1)}{\binom{n}{2}}. \end{aligned} \tag{4}$$

The resulting bound is intuitive: there are $2^n - 2^k(n+1)$ points outside of the radius-1 Hamming spheres, which can become radius-2 Hamming spheres with the addition of $\binom{n}{2}$ points. Note that for $(2^m, 2^m - m - 1, 4)$ extended Hamming codes, the sphere packing bound simplifies to $M_2 \leq 2^{n-2\log(n)}$.

B. Nonlinear Programming Bound

For traditional codes, the sphere-packing bound is not the tightest upper bound available in either the finite-length or asymptotic regimes. A better bound is provided in both cases by Delsarte's linear programming (LP) bound [14]. The LP bound considers the *distance distribution* vector $\mathbf{g} = (g_0, g_1, g_2, \ldots, g_n)$, where

$$g_i = |\{(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{C}, d_H(\mathbf{x}, \mathbf{y}) = i\}|/|\mathcal{C}|.$$

All values of g_i are nonnegative, $g_0 = 1$, and $g_i = 0$ for $1 \leq i < d_{min}$. The remaining condition in the LP bound is $\mathbf{gQ} \geq 0$, where $\mathbf{Q}$ is the so-called *second eigenmatrix* of the Hamming association scheme on $\mathbb{F}_2^n$. Delsarte showed that $\mathbf{Q}$ can be formed by the relation $\mathbf{Q}_{i,j} = K_j(i)$, with the Krawtchouk polynomial defined as

$$K_k(x) = \sum_{j=0}^{k} (-1)^n \binom{x}{j}\binom{n-x}{k-j}. \tag{5}$$

Clearly, we have that $\sum_{i=0}^{n} g_i = |\mathcal{C}|$, and thus maximizing $\sum_{i=0}^{n} g_i$ also maximizes the size of the code.

We are ready to introduce our modified bound. For our purposes, we have two classes of codewords, regular codewords and special codewords, and we create three separate distance distribution vectors. Distance distribution vectors $\mathbf{a}, \mathbf{b}$, and $\mathbf{c}$ contain distances between two regular codewords, one regular codeword and one special codeword, and two special codewords, respectively:

$$\begin{aligned} a_i &= |\{(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{M}_1, d_H(\mathbf{x}, \mathbf{y}) = i\}|/|\mathcal{C}|, \\ b_i &= |\{(\mathbf{x}, \mathbf{y}) :\ \mathbf{x} \in \mathcal{M}_1, \mathbf{y} \in \mathcal{M}_2 \text{ or} \\ &\qquad \mathbf{x} \in \mathcal{M}_2, \mathbf{y} \in \mathcal{M}_1, d_H(\mathbf{x}, \mathbf{y}) = i\}|/|\mathcal{C}|, \\ c_i &= |\{(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in \mathcal{M}_2, d_H(\mathbf{x}, \mathbf{y}) = i\}|/|\mathcal{C}|. \end{aligned}$$

Recall that our goal is to maximize M_2. Summing over all the entries in $\mathbf{c}$, we have:

$$\sum_{i=0}^{n} c_i = \frac{(M_2)^2}{|\mathcal{C}|} \implies M_2 = \sqrt{2^k \sum_{i=0}^{n} c_i},$$

and thus, for given n and k, our objective function is to maximize $\sum_{i=0}^{n} c_i$. Note that $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{a} + \mathbf{b} + \mathbf{c}$ are valid (scaled) distance distribution vectors of codes. Due to the multiple distance distribution vectors, there are a number of substantial differences in the constraints of our programming bound and those from Delsarte's LP bound.

We first establish *inequality* constraints. Our first three inequality constraints are $\mathbf{aQ} \geq 0$, $\mathbf{cQ} \geq 0$, and $(\mathbf{a} + \mathbf{b} + \mathbf{c})\mathbf{Q} \geq 0$, where $\mathbf{Q}$ is the same eigenmatrix based on the Krawtchouk polynomial in Equation 5. Similarly to the LP bound, we require all a_i, b_i and c_i to be nonnegative.

We now establish the *equality* constraints. We have $\sum_{i=0}^{n} b_i = \frac{2M_1M_2}{|\mathcal{C}|}$. Thus our total codewords condition is:

$$\sum_{i=0}^{n}(a_i + b_i + c_i) = \frac{(M_1)^2 + 2M_1M_2 + (M_2)^2}{|\mathcal{C}|} = \frac{|\mathcal{C}|^2}{|\mathcal{C}|} = 2^k.$$

Due to the minimum Hamming distances in the distribution vectors, we have $a_i = 0$ for $1 \leq i \leq 2$, $b_i = 0$ for $0 \leq i \leq 3$ and $c_i = 0$ for $1 \leq i \leq 4$. Additionally, since $a_0 = M_1/|\mathcal{C}|$ and $c_0 = M_2/|\mathcal{C}|$, we have that $a_0 + c_0 = 1$.

So far, we have constructed a linear program, i.e., the objective function as well as all the constraints are all *affine* functions. Unfortunately, while the condition $a_0 + c_0 = 1$ is necessary, it is not specific enough to guarantee a solution consistent with our distribution vector definitions. We require an extra condition on a_i and c_i, as follows:

$$\begin{aligned} a_0 = \frac{M_1}{2^k} &\implies 2^k(a_0)^2 - \sum_{i=0}^{n} a_i = 0, \\ c_0 = \frac{M_2}{2^k} &\implies 2^k(c_0)^2 - \sum_{i=0}^{n} c_i = 0. \end{aligned}$$

This equality constraint is not affine, and thus our program is no longer a convex optimization.However, given the smoothness of our quadratic constraints, there are many efficient optimization techniques for this nonlinear program (NLP) [15].

Theorem 1. *For a given n and k, we have*

$$M_2 \leq \sqrt{2^k \textstyle\sum_{i=0}^{n} c_i^*},$$

where $\mathbf{c}^*$ is the solution to the following nonlinear program:

maximize: $\sum_{i=0}^{n} c_i$ *subject to:*

Inequality Constraints	*Equality Constraints*
$\mathbf{a} \geq 0,$	$a_i = 0, \ 1 \leq i \leq 2,$
$\mathbf{b} \geq 0,$	$b_i = 0, \ 0 \leq i \leq 3,$
$\mathbf{c} \geq 0,$	$c_i = 0, \ 1 \leq i \leq 4,$
$\mathbf{aQ} \geq 0,$	$a_0 + c_0 = 1,$
$\mathbf{cQ} \geq 0,$	$\sum_{i=0}^{n}(a_i + b_i + c_i) = 2^k,$
$(\mathbf{a} + \mathbf{b} + \mathbf{c})\mathbf{Q} \geq 0,$	$2^k(a_0)^2 - \sum_{i=0}^{n} a_i = 0,$
	$2^k(c_0)^2 - \sum_{i=0}^{n} c_i = 0.$

The NLP bound correctly returns *infeasible solution* for any parameters (n, k) with less redundancy than the associated

Hamming code. Additionally, for any (n, k) with more redundancy than the associated $t = 2$ BCH code, the program correctly returns $M_2 = 2^k$. As shown in Table I, the NLP bound is not strictly tighter than the sphere packing bound.

III. Lower Bounds on (Maximal) Special Messages

Next, we present two lower bounds on the maximal number of special messages. The first, based on graph coloring, is applicable to any given linear code with minimum distance $2t+2$. The second, based on subcodes, is stronger, but is only applicable to our specific SEC-smDEC regime, i.e., $t = 1$.

A. Graph Coloring Bound

Suppose we have a linear $(n, k, 2t+2)$ code $\mathcal{C}$, i.e., t errors are correctable and $t+1$ errors are detectable (in the traditional sense, without UMP). Our goal is to derive a lower bound on how many special codewords we can have; that is, we seek to partition $\mathcal{C}$ into $\mathcal{M}_t$ and $\mathcal{M}_{t+1}$ and to maximize M_{t+1}.

Code $\mathcal{C}$ yields the graph $G = (V, E)$ where vertices V represent the codewords in $\mathcal{C}$, so that $|V| = 2^k$. Two vertices have an edge between them if and only if their Hamming distance is exactly $2t + 2$. The following remark establishes the connection between G and our SEC-smDEC code.

Remark 1. *The size of the maximal independent set of G, denoted as $\alpha(G)$, is a lower-bound on M_{t+1} for $\mathcal{C}$.*

Proof: Recall that a codeword can correct up to t errors if its Hamming sphere around the codeword has radius t. In $\mathcal{C}$, with $d_{min} = 2t + 2$, no two codewords in $\mathcal{M}_{t+1}$ can be at a distance $2t + 2$ apart (or else their Hamming spheres would intersect). Thus in our graph, no two vertices associated with codewords in $\mathcal{M}_{t+1}$ can be adjacent. The vertices in the maximal independent set of G correspond directly to the codewords in $\mathcal{M}_{t+1}$, thus we have $\alpha(G) \geq M_{t+1}$. ■

The linearity of $\mathcal{C}$ implies that G is a regular graph, that is, each vertex is incident to the same number of edges. For regular graphs, the following bound (equivalent to Turan's theorem [16]) is tight: $\alpha(G) \geq |V|/(1 + \deg(V))$, where $\deg(V)$ is the degree of a vertex in the graph. The degree of each vertex is equal to the number of minimum weight codewords in $\mathcal{C}$, denoted as $W(2t + 2)$. Thus we have the following bound on the number of special codewords.

Lemma 1. *Any linear $(n, k, 2t+2)$ code $\mathcal{C}$ can be partitioned into $\mathcal{M}_t$ and $\mathcal{M}_{t+1}$ with*

$$M_{t+1} \geq (2^k)/(1 + W(2t + 2)).$$

To apply this bound to the SEC-smDEC scenario, note that for the $(2^m, 2^m - m - 1, 4)$ extended Hamming code, we have $W(4) = \binom{n}{2}(\frac{n}{2}-1)/6$, as expected. Applying Lemma 1 yields:

Lemma 2. *A $(2^m, 2^m - m - 1, 4)$ extended Hamming code can be partitioned into $\mathcal{M}_1$ and $\mathcal{M}_2$ with*

$$M_2 \geq \frac{6 \cdot 2^k}{6 + \binom{n}{2}(\frac{n}{2} - 1)}.$$

In Table I, we only include results for the extended Hamming code parameters and the optimum values of $W(4)$ for the $(22, 16)$ and $(39, 32)$ cases.

TABLE I
Upper and Lower Bounds on $\log(M_2)$

$(\mathbf{n}, \mathbf{k})$	Lower Bounds: Graph Col.	Lower Bounds: BCH Subcode	Upper Bounds: Sphere Packing	Upper Bounds: NLP
$(10, 4)$	–	1	4.17	3.59
$(9, 4)$	–	1	3.17	2.59
$(8, 4)$	1	1	2	2
$(7, 3)$	–	0	1.59	1.59
$(18, 11)$	–	7	10.51	10.33
$(17, 11)$	–	7	9.43	9.42
$(16, 11)$	3.91	7	8	8.73
$(15, 10)$	–	6	7.29	8
$(22, 16)$	8.03	11	13.51	13.75
$(39, 32)$	21.93	26	28.93	28.98

B. BCH Subcode Bound

A code $\mathcal{C}_S \subseteq \mathcal{C}_1$ is called a *subcode* of code $\mathcal{C}_1$. If $d(\mathcal{C}_S) \geq 5$, $d(\mathcal{C}_1) \geq 4$, and $\mathcal{C}_S \subseteq \mathcal{C}_1$, then Equations (1), (2), and (3) are satisfied with the following mapping: $c \in \mathcal{M}_2$ if $c \in \mathcal{C}_S$, and $c \in \mathcal{M}_1$ if $c \in \mathcal{C}_1$ and $c \notin \mathcal{C}_S$. Therefore, $\mathcal{C}_1$ can be used as a SEC-smDEC code with $M_2 = |\mathcal{C}_S|$.

The parity-check matrix of a $(2^m - 1, 2^m - m - 1, 3)$ Hamming code can be represented as $\boldsymbol{H} = [\mathbf{1}\ \boldsymbol{\alpha}\ \boldsymbol{\alpha^2}\ \cdots \boldsymbol{\alpha^{n-1}}]$, where $\boldsymbol{\alpha}$ is a primitive element of GF(2^m) (the bold elements represent column vectors of length m). We *expurgate* $\boldsymbol{H}$ by adding the row $[\mathbf{1}\ \boldsymbol{\alpha^3}\ \boldsymbol{\alpha^6}\ \cdots \boldsymbol{\alpha^{3(n-1)}}]$, resulting in the parity-check matrix for a $(2^m-1, 2^m-2m-1, 5)$ BCH code. Lastly, we *extend* the code by adding an overall parity bit, yielding $\boldsymbol{H_S}$, the parity-check matrix for the subcode:

$$\boldsymbol{H_S} = \begin{bmatrix} \mathbf{1} & \boldsymbol{\alpha} & \boldsymbol{\alpha^2} & \cdots & \boldsymbol{\alpha^{n-1}} & \mathbf{0} \\ \mathbf{1} & \boldsymbol{\alpha^3} & \boldsymbol{\alpha^6} & \cdots & \boldsymbol{\alpha^{3(n-1)}} & \mathbf{0} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}.$$

Notice that $\mathcal{C}_S$, the code associated with $\boldsymbol{H_S}$ is a $(2^m, 2^m - 2m-1, 6)$ extended binary BCH code that is a subcode of the $(2^m, 2^m - m - 1, 4)$ extended Hamming code. Therefore, for $(n, k) = (2^m, 2^m - m - 1)$, we have:

$$M_2 \geq 2^{2^m - 2m - 1} = 2^{n - 2\log(n) - 1}.$$

The subcode property is preserved in the process of shortening if in both the overall code and the subcode parity-check matrices the same columns are eliminated. The following lemma provides the results for Table I.

Lemma 3. *For a code $\mathcal{C}$ with parameters (n, k) in our SEC-smDEC region of interest, $M_2 \geq 2^{n - 2\lceil\log(n)\rceil - 1}$.*

IV. Explicit SEC-smDEC Code Construction

Data is often highly structured; for example, in instruction memory encoded with RISC-V instruction set architecture (ISA) v2.0 [17], the least-significant 7 bits are the *opcode* that primarily determines the action taken by the instruction.

Due to the popularity of $(39, 32)$ SECDED codes in byte-oriented architecures, we seek a $(39, 32)$ SEC-smDEC coding framework that efficiently maps special messages of our choice to special codewords. For our coding scheme, we build upon Section III-B by using a BCH subcode. For a $(39, 32)$ code, Lemma 3 yields $\log(M_2) \geq 26$. There are two natural choices for our special messages. First, since there are 25 non-opcode

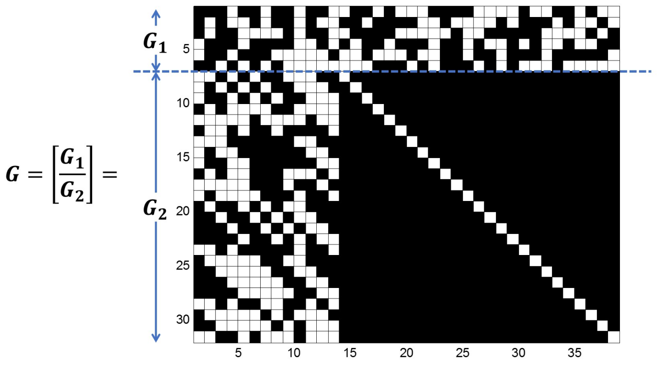

Fig. 1. The generator matrix $\boldsymbol{G}$ for our $(39, 32)$ SEC-smDEC code, and the generator matrix $\boldsymbol{G_2}$ for the $(39, 26)$ BCH subcode. The white and black squares represent 1s and 0s, respectively.

bits in the message, we are able to offer DEC protection to 2 opcodes and all of their associated messages. Alternatively, we can offer full DEC protection to any message beginning with a run of 0s of length at least $32 - 26 = 6$ bits.

To create the generator matrix $\boldsymbol{G}$ in Figure 1, start with the generator matrix for a $(63, 51, 5)$ BCH code, then

1) *Extend* the matrix with an overall parity bit.
2) *Shorten* the matrix to $(39, 26)$ and use elementary row operations to convert it to systematic form; this is $\boldsymbol{G_2}$.
3) *Augment* the matrix (by adding 6 rows) while ensuring the overall minimum distance is at least 4; this is $\boldsymbol{G}$.

Since the minimum distance of $\boldsymbol{G_2}$ is 6, and the minimum distance of the overall matrix $\boldsymbol{G}$ is 4, conditions (1), (2), and (3) are satisfied and we have a $(39, 32)$ SEC-smDEC code with $M_2 = 2^{26}$. $\boldsymbol{G}$ maps special messages to codewords as follows: pre-map the special messages so that the first 6 bits are all 0; this way, the encoding of these messages takes place entirely within $\boldsymbol{G_2}$. In general, this can be a daunting task; however, when there is structure to the special data, such as the opcodes, the user can easily perform the pre-mapping.

The parity-check matrix $\boldsymbol{H_2}$, corresponding to $\boldsymbol{G_2}$, can easily be found since $\boldsymbol{G_2}$ is in systematic form. To find $\boldsymbol{H}$, the overall parity-check matrix, we use elementary row operations to convert $\boldsymbol{G}$ to systematic form (this does not alter $\boldsymbol{H}$).

The decoding process follows a two-step approach. First, use $\boldsymbol{H}$ to decode as usual with a SEC-DED code. If there were no errors or only a single bit in error, then the decoding process is finished; however, if the error results in a DUE, then we use $\boldsymbol{H_2}$ to decode. After the decoding process, we map our resulting codeword back to its original message.

We collected dynamic memory access traces of various benchmarks and analyzed them to determine the most frequent opcodes (in instruction memory) and the relative frequency of common patterns (in data memory) over the entire suite; the results are shown in Table II. We find that the distribution of opcodes is highly asymmetric—the two most frequent instructions, `LOAD` and `OP-IMM` [17], comprise an average of 51% and 56% of the instructions in the AxBench [18] and SPEC CPU2006 suites, respectively. For data memory the majority of stored vectors begin with a run of 0s consistently throughout each benchmark (as demonstrated by the low variance values).

TABLE II
FRACTION OF SPECIAL MESSAGES PER BENCHMARK WITHIN SUITE

	Most Freq. 2 opcodes (Instruction Memory)			**First 6 bits are 0 (Data Memory)**		
Suite	**Max**	**Mean**	**Var**	**Max**	**Mean**	**Var**
AxBench	0.51	0.46	1.3E-3	0.92	0.86	4.0E-3
SPEC CPU2006	0.56	0.37	2.1E-2	0.99	0.89	2.0E-2

Using the same number of redundancy bits as the SECDED code, our SEC-smDEC coding scheme offers full DEC protection to special messages based on a customizable mapping scheme. Depending on the user goal, our construction could lead to system-level benefits such as less frequent checkpoints in supercomputers and decreased risk of catastrophic failure from erroneous special messages. Preliminary results indicate that the $(39, 32)$ SEC-smDEC scheme can improve the overall failure rate (due to a DUE or an MCE) by up to 9x with no additional redundancy using the leading run of 0s mapping technique.

V. Conclusion

We studied a practical class of UMP codes, introduced bounds on code cardinality, an explicit construction based on subcodes, and provided motivating applications from real data.

References

[1] M. Gottscho *et al.*, "Software-defined error-correcting codes," in *Proc. IEEE/IFIP Int. Conf. on Dependable Systems and Networks Workshops*, Jun.-Jul. 2016, pp. 276–282.
[2] B. Masnick and J. Wolf, "On linear unequal error protection codes," *IEEE Trans. Inf. Theory*, vol. 13, no. 4, pp. 600–607, Oct. 1967.
[3] P. Nikolaou *et al.*, "Modeling the implications of DRAM failures and protection techniques on datacenter TCO," in *Proc. ACM Int. Symp. on Microarchitecture*, Dec. 2015, pp. 572–584.
[4] J. Chang *et al.*, "The 65-nm 16-MB shared on-die L3 cache for the dual-core Intel Xeon processor 7100 series," *IEEE J. Solid-State Circuits*, vol. 42, no. 4, pp. 846–852, Mar. 2007.
[5] I. Boyarinov and G. Katsman, "Linear unequal error protection codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 2, pp. 168–175, Mar. 1981.
[6] N. Abramson, "A class of systematic codes for non-independent errors," *IRE Trans. Inf. Theory*, vol. 5, no. 4, pp. 150–157, Dec. 1959.
[7] P. Reviriego *et al.*, "A method to design SEC-DED-DAEC codes with optimized decoding," *IEEE Trans. Device Mater. Rel.*, vol. 14, no. 3, pp. 884–889, Sep. 2014.
[8] S. Kaneda and E. Fujiwara, "Single byte error correcting double byte error detecting codes for memory systems," *IEEE Trans. on Comput.*, vol. 31, no. 7, pp. 596–602, Jul. 1982.
[9] T. J. Dell, "A white paper on the benefits of chipkill-correct ECC for PC server main memory," *IBM Microelectronics Division*, vol. 11, 1997.
[10] S. Borade, B. Nakiboglu, and L. Zheng, "Unequal error protection: An information-theoretic perspective," *IEEE Trans. Inf. Theory*, vol. 55, no. 12, pp. 5511–5539, Dec. 2009.
[11] Y. Y. Shkel, V. Y. Tan, and S. C. Draper, "Unequal message protection: Asymptotic and non-asymptotic tradeoffs," *IEEE Trans. Inf. Theory*, vol. 61, no. 10, pp. 5396–5416, Oct. 2015.
[12] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.
[13] B. Nazer, Y. Y. Shkel, and S. C. Draper, "The awgn red alert problem," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2188–2200, Apr. 2013.
[14] P. Delsarte, "An algebraic approach to the association schemes of coding theory," Ph.D. dissertation, Université Catholique de Louvain, Jun. 1973.
[15] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.
[16] P. Turán, "On an extremal problem in graph theory," *Mat. Fiz. Lapok*, vol. 48, no. 436-452, p. 137, 1941.
[17] A. Waterman *et al.*, "The RISC-V Instruction Set Manual. Volume 1: User-Level ISA, Version 2.0," DTIC Document, Tech. Rep., 2014.
[18] A. Yazdanbakhsh *et al.*, "AxBench: A multi-platform benchmark suite for approximate computing," *IEEE Design & Test*, 2016.