Segmented Reverse Concatenation: A New Approach to Constrained ECC

Ryan Gabrys[*‡], **Paul H. Siegel**[*] and **Eitan Yaakobi**[†]

[*]University of California San Diego, La Jolla, CA 92093
[†]Technion — Israel Institute of Technology, Haifa, Israel 3200003
[‡]Naval Information Warfare Center, San Diego, CA 92152
ryan.gabrys@gmail.com, psiegel@ucsd.edu, yaakobi@cs.technion.ac.il

***Abstract*—In this work, a new coding scheme called *segmented reverse concatenation* is described, which generates constrained codes that can also correct a prescribed number of errors. Our codes are based upon the generalized reverse concatenation method; however, the key difference between our scheme and prior art is that in our scheme the redundancy symbols of the code are able to be partitioned into disjoint segments, each of which requires only a single parity bit to maintain the minimum distance of the code. We consider three potential applications of the new technique, and it is shown that in all three cases our approach improves upon prior art. Our scheme can be applied to many setups, although it is particularly well-suited for scenarios where the constrained encoder has a high information rate.**

I. Introduction

Constrained codes are used in almost all modern storage systems today. Traditional data encoders typically translate some sequence of information bits into codewords that not only comply with the underlying storage medium, but also provide data redundancy to ensure that the information sequence can be recovered if errors occur. In this context, a good code is one which achieves a high information rate while satisfying the coding constraint and having a large minimum distance.

This problem has been studied in various contexts for over 40 years in works such as [2], [3], [4], [5], [6], [7], and [12], to name a few. One of the principal challenges one encounters when constructing constrained error-correcting codebooks is how to efficiently combine an encoder for the constrained code with the encoder for the error-correcting code (ECC). Traditionally, the ECC encoder and the constrained encoder are concatenated by first passing the information sequence through the ECC encoder followed by the constrained encoder [7]. The primary drawback of this approach is that for long block lengths, it may suffer from high error-propagation, and as a result, traditional schemes usually relied on codes with relatively short block lengths [5].

Another approach, known as ***reverse concatenation***, inverts the order of the ECC encoder and constrained encoder as described in the previous paragraph. Under this setup, an information sequence is first passed through the constrained encoder and additional redundancy symbols are then computed from the output of the constrained encoder for the purposes of error-correction [2], [4], [6]. One of the drawbacks to this approach, and one which will be explained in more detail in the next section, is that the design of the redundancy symbols is oftentimes accomplished via brute force methods, which do not scale well as the block length (or the number of errors one wishes to correct) increases.

In this work, we propose a new coding scheme, which we call ***segmented reverse concatenation***, to address this issue. Our approach is a special case of the reverse concatenation method. As before, we begin by first passing our information sequence through a constrained encoder. Then we compute a block of redundancy symbols from the output of this constrained encoder. The key difference between our approach and the traditional reverse concatenation method is that in our approach the block of redundancy symbols is next segmented into a sequence of sub-blocks, referred as *segments*. As a final step, each segment is passed through the constrained encoder and each segment is further protected by an additional parity symbol. It will be shown in Section III that the result is a constrained error-correcting codebook.

This paper is organized as follows. In Section II, we provide more details on our construction and we introduce some tools that will be used for the remainder of the paper. In Section III, the correctness of the construction is proven. In Section IV, we discuss three applications of our technique and we highlight some cases where it improves upon current art. In particular, to the best of our knowledge, our approach can be used to generate the best-known error-correcting WOM-codes, error-correcting balanced codes, and error-correcting run-length limited codes. Due to space limitations, many of the proofs for this paper are omitted as well as discussions on efficient decoding algorithms. The full version is available [8].

II. Background and Preliminaries

In this section, we first review a well-known method, which has been used in the past to generate constrained error-correcting codes, known as reverse concatenation [7]. Afterwards, we discuss our approach and review classical Goppa codes, which will be used in the next section to describe our construction.

A. Reverse Concatenation

The basic idea behind the traditional reverse concatenation method is to make the difficult problem of designing codes that simultaneously satisfy both a minimum distance and a

E. Yaakobi acknowledges support from the Center for Memory and Recording Research at UCSD. This work is also funded by the United States-Israel BSF grant 2018048.

coding constraint necessary for only a small portion of the codeword. The first key ingredient in these constructions is the use of systematic ECCs. This family of codes allows one to mostly focus on designing an efficient encoder for the coding constraint, which is usually an easier problem.

Let $\mathscr{E} : \{0,1\}^* \to \{0,1\}^*$ be an invertible encoder whose image is a set of vectors that satisfy the coding constraint (but without any guarantees on the minimum distance). For our discussion, we assume that given a binary vector $\boldsymbol{v}$ of length N, the output $\mathscr{E}(\boldsymbol{v})$ is a vector of length aN where $a \geq 1$ is minimal and that the code has the fewest possible number of redundant bits. We also assume that the encoder $\mathscr{E}$ has polynomial time encoding/decoding complexity with respect to the length of the output sequence. Suppose that $R : \{0,1\}^N \to \{0,1\}^{t\lceil \log N \rceil}$ is a systematic encoder for a t-error-correcting code where for any two distinct vectors $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \{0,1\}^N$, the minimum Hamming distance between $(\boldsymbol{z}_1, R(\boldsymbol{z}_1))$ and $(\boldsymbol{z}_2, R(\boldsymbol{z}_2))$ is at least $2t+1$. For simplicity, we assume that the number of redundancy bits is $t\lceil \log N \rceil$; however, the technique can be extended to accommodate any systematic t-error-correcting code.

Using the maps defined in the previous paragraph suppose we start with an information vector $\boldsymbol{u} \in \{0,1\}^k$ and from this, we generate the redundancy bits $R(\mathscr{E}(\boldsymbol{u})) \in \{0,1\}^{t\lceil \log ak \rceil}$. Given the map $f : \{0,1\}^{t\lceil \log ak \rceil} \to \{0,1\}^{t\lceil \log ak \rceil + r}$, which will be defined shortly, a code produced according to the reverse concatenation method will have codewords of the following form:

$$\Big(\mathscr{E}(\boldsymbol{u}), f(R(\mathscr{E}(\boldsymbol{u})))\Big) \in \{0,1\}^{ak + t\lceil \log ak \rceil + r}.$$

In order to make this approach work, one still needs to design the map f so that $f(R(\mathscr{E}(\boldsymbol{u})))$ satisfies the coding constraint[1] and it belongs to a code with minimum Hamming distance $2t+1$. For the case where t is a constant with respect to ak, the map f can be designed efficiently using brute force methods in polynomial time with respect to the length of the output sequence.

The trouble with the reverse concatenation method is two-fold. First, it may be difficult to derive tight upper/lower bounds on the redundancy $|f(R(\mathscr{E}(\boldsymbol{u})))|$, and in general the quantity $|f(R(\mathscr{E}(\boldsymbol{u})))|$ depends on the results of a brute force search. Another potential issue with the reverse concatenation method is that it does not scale well as t increases. Even for the case where $t = \mathcal{O}(\log k)$ (or where t is larger), the design of f can no longer be accomplished in polynomial-time using brute force methods.

B. Our Approach

For our approach, suppose we wish to encode the information vector $\boldsymbol{u} \in \{0,1\}^k$ and $\boldsymbol{z} = \mathscr{E}(\boldsymbol{u})$. The high level idea behind our approach, and which will be explained in more detail in Section III, is to partition the data from $\boldsymbol{z} = R(\mathscr{E}(\boldsymbol{u}))$ into t segments where each segment is protected using only a single parity bit. To apply this technique, we begin by first computing a set of t *redundancy segments* based on the vector $\boldsymbol{z}$ as follows

$$\Big\{\boldsymbol{r}_1(\boldsymbol{z}), \boldsymbol{r}_2(\boldsymbol{z}), \ldots, \boldsymbol{r}_t(\boldsymbol{z})\Big\},$$

where for any $i \in [t]$, $\boldsymbol{r}_i(\boldsymbol{z})$ has length approximately $\log |\boldsymbol{z}| = \lceil \log ak \rceil$, and the length of $\boldsymbol{z}$ is ak. We use $\mathscr{E}$ to form the constrained redundancy segments:

$$\Big(\rho_1(\boldsymbol{z}), \ldots, \rho_t(\boldsymbol{z})\Big) = \Big(\mathscr{E}(\boldsymbol{r}_1(\boldsymbol{z})), \ldots, \mathscr{E}(\boldsymbol{r}_t(\boldsymbol{z}))\Big).$$

Then, for $i \in [t]$, let $P(\rho_i(\boldsymbol{z})) = \sum_{j=1}^{|\rho_i(\boldsymbol{z})|} \rho_i(\boldsymbol{z})_j \bmod 2$ denote the parity of the vector $\rho_i(\boldsymbol{z})$. Finally, we form our codeword $\boldsymbol{x}$ as follows:

$$\boldsymbol{x} = \Big(\boldsymbol{z}, \rho_1(\boldsymbol{z}), P(\rho_1(\boldsymbol{z})), \ldots, \rho_t(\boldsymbol{z}), P(\rho_t(\boldsymbol{z}))\Big) \in \{0,1\}^n. \tag{1}$$

We refer to the constructed codebooks as ***segmented reverse concatenation codes*** or ***t-SRC codes***. It will be proven in Theorem 1 that a t-SRC code has minimum distance $2t+1$. Notice from (1) that $n \approx ak + at\log(ak) + t$ where we have dropped the ceiling function for notational ease. We will show in Section IV that this result can be used to generate code constructions for many existing and well-studied setups which include: (i) Error-correcting (EC) WOM-codes, (ii) EC balanced codes, and (iii) EC run-length limited codes. Although we consider these three setups in detail, we believe that there may be many other areas where this construction can be useful as well. As mentioned earlier, our approach enjoys the benefit that it does not rely on brute force search methods. Furthermore, the number of redundant bits is at most approximately $ak + at\log(ak) + t - k$ irrespective of the particular coding constraint where $t \leq \mathcal{O}(\sqrt{n})$.

C. Background on Goppa Codes

The primary tool used in our construction is Goppa codes, which we briefly review now. In the following exposition, we adopt the notation from [1]. Let m be a positive integer and suppose $R = \{\alpha_1, \alpha_2, \ldots, \alpha_t\} \subseteq \mathbb{F}_{2^m}$. Define

$$g(\xi) = (\xi - \alpha_1) \cdot (\xi - \alpha_2) \cdots (\xi - \alpha_t) \in \mathbb{F}_{2^m}[\xi] \tag{2}$$

to be a polynomial over $\mathbb{F}_{2^m}$ of degree t with the indeterminate ξ. Let $L = \{\gamma_1, \ldots, \gamma_{|L|}\} \subseteq \mathbb{F}_{2^m} \setminus R$. Then, the Goppa code $C(L,R)$ of length $|L|$ over $\mathbb{F}_2$ is the code whose coordinates are indexed by elements of L where

$$C(L,R) = \Big\{\boldsymbol{x} = (x_{\gamma_1}, \ldots, x_{\gamma_{|L|}}) \in \mathbb{F}_2^{|L|} : \sum_{i=1}^{|L|} \frac{x_{\gamma_i}}{\xi - \gamma_i} \equiv 0 \bmod g(\xi)\Big\}. \tag{3}$$

We state a well-known result which can be found in [1]. For shorthand, let $d(C)$ denote the minimum Hamming distance for the code C.

Lemma 1. *(cf., [1]) Let $R \subset \mathbb{F}_{2^m}$ be the set of roots of $g(\xi)$ that are contained in $\mathbb{F}_{2^m}$. Let $L \subseteq \mathbb{F}_{2^m} \setminus R$. Then, $d(C(L,R)) \geq 2|R| + 1$.*

[1] Specifically, we need for the vector $\Big(\mathscr{E}(\boldsymbol{u}), f(R(\mathscr{E}(\boldsymbol{u})))\Big)$ to satisfy the coding constraint, but we make the relaxing assumption that the coding constraint is satisfied if each of the vectors $\mathscr{E}(\boldsymbol{u})$ and $f(R(\mathscr{E}(\boldsymbol{u})))$ satisfy the coding constraint.

Next, we state an implication of the previous lemma which will be used in our construction.

Lemma 2. *Let $R_1, R_2 \subset \mathbb{F}_{2^m}$ be two disjoint subsets and suppose that $L \subseteq \mathbb{F}_{2^m} \setminus \big(R_1 \cup R_2\big)$. Then,*
1) $d\big(C(L, R_1)\big) \geq 2|R_1| + 1$,
2) $d\big(C(L, R_2)\big) \geq 2|R_2| + 1$, and
3) $d\big(C(L, R_1 \cup R_2)\big) \geq 2(|R_1| + |R_2|) + 1$.

According to (2), the types of Goppa polynomials $g(\xi)$ we will be interested in can be written as a product of linear terms over $\mathbb{F}_{2^m}$. For a vector $\boldsymbol{x} = (x_{\gamma_1}, x_{\gamma_2}, \ldots, x_{\gamma_{|L|}}) \in \mathbb{F}_2^{|L|}$, we associate $f_{\boldsymbol{x},L}(\xi)$ with $\boldsymbol{x}$ where

$$f_{\boldsymbol{x},L}(\xi) = \sum_{i=1}^{|L|} \frac{x_{\gamma_i}}{\xi - \gamma_i} \in \mathbb{F}_{2^m}[\xi]. \tag{4}$$

When the set L is clear from the context, we will omit it from our notation and refer to $f_{\boldsymbol{x},L}(\xi)$ as $f_{\boldsymbol{x}}(\xi)$. Using this interpretation, since $f_{\boldsymbol{x},L}(\xi) \bmod (\xi - \alpha_i) \equiv 0$ if and only if $f_{\boldsymbol{x},L}(\alpha_i) = 0$, we have

$$C(L, R) = \Big\{\boldsymbol{x} \in \mathbb{F}_2^{|L|} : \forall \alpha \in R, f_{\boldsymbol{x},L}(\alpha) = 0\Big\}.$$

Consider the following set which is closely related to $C(L, R)$. Now suppose that $R = \{\alpha_1, \ldots, \alpha_t\} \subseteq \mathbb{F}_{2^m}$ and also that $\boldsymbol{V} = (V_{\alpha_1}, \ldots, V_{\alpha_t}) \in \mathbb{F}_{2^m}^t$. Then we will use the set R as a set of evaluation points and $\boldsymbol{V}$ as a set of evaluation values to define the following set

$$C(L, R, \boldsymbol{V}) = \Big\{\boldsymbol{x} \in \mathbb{F}_2^{|L|} : \forall \alpha \in R, f_{\boldsymbol{x},L}(\alpha) = V_\alpha\Big\}. \tag{5}$$

Since the vectors from $C(L, R, \boldsymbol{V})$ are contained in a coset of $C(L, R)$, the following corollary is straightforward.

Corollary 1. *The code $C(L, R, \boldsymbol{V})$ has minimum distance $2|R| + 1$.*

It also follows from the results in [1] that there exists a polynomial time (with complexity at most $|L|^3$) decoder for the code $C(L, R, \boldsymbol{V})$. For the vector $\boldsymbol{V} = (V_{\alpha_1}, V_{\alpha_2}, \ldots, V_{\alpha_t}) \in \mathbb{F}_{2^m}^t$ which is indexed by the elements in R, and a subset $R' \subseteq R$, let $\boldsymbol{V}_{R'}$ be the result of removing components of $\boldsymbol{V}$ that are not indexed by elements from R'. For example if $\boldsymbol{V} = (V_{\alpha_1}, V_{\alpha_2}, V_{\alpha_3})$ and $R' = \{\alpha_1, \alpha_3\}$, then $\boldsymbol{V}_{R'} = (V_{\alpha_1}, V_{\alpha_3})$. The next corollary follows immediately from the same logic as Lemma 2.

Corollary 2. *Suppose $R = R_1 \cup R_2 \subseteq \mathbb{F}_{2^m}$ where $R_1 \cap R_2 = \emptyset$ and let $L = \mathbb{F}_{2^m} \setminus R$, $|R| = t$, $\boldsymbol{V} \in \mathbb{F}_{2^m}^t$. Then,*
1) $d\big(C(L, R_1, \boldsymbol{V}_{R_1})\big) \geq 2|R_1| + 1$,
2) $d\big(C(L, R_2, \boldsymbol{V}_{R_2})\big) \geq 2|R_2| + 1$, and
3) $d\big(C(L, R, \boldsymbol{V}\big) \geq 2(|R_1| + |R_2|) + 1$.
Furthermore for each of these three codes, there exists an efficient polynomial time decoder that has complexity at most $\mathcal{O}(|L|^3)$.

III. Construction of SRC Codes

We now discuss our SRC construction in more detail and then we show that the resulting code has the desired minimum distance properties. Let $R = \{\alpha_1, \alpha_2, \ldots, \alpha_t\} \subseteq \mathbb{F}_{2^m}$, and let

$$L = \{\gamma_1, \ldots, \gamma_{|L|}\} \subseteq \mathbb{F}_{2^m} \setminus R.$$

Let $\boldsymbol{u} \in \mathbb{F}_2^k$ be a sequence of k information bits. For a vector $\boldsymbol{z} = \mathscr{E}(\boldsymbol{u}) \in \{0,1\}^{|L|}$ (where $|L| = ak$), we define the redundancy segment $\boldsymbol{r}_i(\boldsymbol{z})$ for $i \in [t]$ as follows:

$$\boldsymbol{r}_i(\boldsymbol{z}) = f_{\boldsymbol{z},L}(\alpha_i) \in \mathbb{F}_{2^m},$$

We will sometimes refer to elements of $\mathbb{F}_{2^m}$ also as binary vectors from $\mathbb{F}_2^m$ according to some fixed basis function. Let $\rho_i(\boldsymbol{z}) = \mathscr{E}(f_{\boldsymbol{z}}(\alpha_i))$ (where we drop the L subscript from f to avoid clutter) and let $P_i = P\left(\rho_i(\boldsymbol{z})\right)$. A t-segmented reverse concatenated code of length $n = ak + at\log(ak) + t$ and evaluation point set R is

$$SRC(n, R) = \Bigg\{\boldsymbol{x} \in \mathbb{F}_2^n : \boldsymbol{u} \in \mathbb{F}_2^k, \boldsymbol{z} = \mathscr{E}(\boldsymbol{u}), \boldsymbol{x} = \Big(\boldsymbol{z}, \rho_1(\boldsymbol{z}), P_1, \ldots, \rho_t(\boldsymbol{z}), P_t\Big)\Bigg\}. \tag{6}$$

For two vectors $\boldsymbol{x}, \boldsymbol{y}$, let $d(\boldsymbol{x}, \boldsymbol{y})$ denote their Hamming distance. The next lemma is the key result which will be used to prove the minimum distance of our t-SRC codes.

Lemma 3. *Suppose $\boldsymbol{z}_1 \neq \boldsymbol{z}_2$ where $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{F}_2^{|L|}$. Let $e = \lfloor \frac{d(\boldsymbol{z}_1, \boldsymbol{z}_2) - 1}{2} \rfloor$. Then,*

$$\Big|\big\{\alpha \in R : f_{\boldsymbol{z}_1}(\alpha) \neq f_{\boldsymbol{z}_2}(\alpha)\big\}\Big| \geq t - e.$$

Proof. Consider the vector $\boldsymbol{d} = \boldsymbol{z}_1 + \boldsymbol{z}_2 \in \mathbb{F}_2^{|L|}$. Let $Z \subseteq R$ be such that for any $\alpha \in Z$, $f_{\boldsymbol{z}_1}(\alpha) = f_{\boldsymbol{z}_2}(\alpha)$ and index $\boldsymbol{d}$ so that $\boldsymbol{d} = (d_{\gamma_1}, \ldots, d_{\gamma_{|L|}})$. Due to linearity, it follows that if $f_{\boldsymbol{z}_1}(\alpha) = f_{\boldsymbol{z}_2}(\alpha)$, then $f_{\boldsymbol{d}}(\alpha) = 0$, so that α is a root of $f_{\boldsymbol{d}}(\xi)$. From Lemma 2, it follows that $\boldsymbol{d}$ belongs to a code with minimum Hamming distance $2|Z|+1$, and since $\boldsymbol{d}$ is non-zero, we have $d(\boldsymbol{z}_1, \boldsymbol{z}_2) \geq 2|Z| + 1$. Since $e = \lfloor \frac{d(\boldsymbol{z}_1, \boldsymbol{z}_2) - 1}{2} \rfloor$, it follows that $e \geq |Z|$. Since $|R| = t$, it follows that $\Big|\big\{\alpha \in R : f_{\boldsymbol{z}_1}(\alpha) \neq f_{\boldsymbol{z}_2}(\alpha)\big\}\Big| = t - |Z| \geq t - e$, which completes the proof. ∎

We use the previous lemma to prove the following, where $SRC(n, R)$ refers to the set defined in (6).

Theorem 1. *For any n and $t \leq \mathcal{O}(n^{1/2})$, $d(SRC(n, R)) \geq 2t + 1$.*

Proof. Let $\boldsymbol{x} = (\boldsymbol{z}_1, \rho_1(\boldsymbol{z}_1), P_1, \ \ldots, \rho_t(\boldsymbol{z}_1), P_t)$, $\boldsymbol{y} = (\boldsymbol{z}_2, \rho_1(\boldsymbol{z}_2), P_1', \ \ldots, \rho_t(\boldsymbol{z}_2), P_t')$ be two distinct vectors from $SRC(n, R)$. Assume that $e = \lfloor \frac{d(\boldsymbol{z}_1, \boldsymbol{z}_2) - 1}{2} \rfloor$. Then, according to Lemma 3, $\Big|\big\{\alpha \in R : f_{\boldsymbol{z}_1}(\alpha) \neq f_{\boldsymbol{z}_2}(\alpha)\big\}\Big| \geq t - e$. Since $\mathscr{E}$ is an invertible mapping, it follows that for every $\alpha_i \in R$ where $f_{\boldsymbol{z}_1}(\alpha_i) \neq f_{\boldsymbol{z}_2}(\alpha_i)$, $\rho_i(\boldsymbol{z}_1) \neq \rho_i(\boldsymbol{z}_2)$. For every $i \in [t]$ we have $d\big((\rho_i(\boldsymbol{z}_1), P_i), (\rho_i(\boldsymbol{z}_2), P_i')\big) \geq 2$ when $f_{\boldsymbol{z}_1}(\alpha_i) \neq f_{\boldsymbol{z}_2}(\alpha_i)$, so that $d(\boldsymbol{x}, \boldsymbol{y}) \geq 2e + 1 + 2(t - e) = 2t + 1$, as desired. ∎

Assuming $\mathscr{E}$ has a polynomial-time encoding algorithm, it is straightforward to verify from (6), that $SRC(n, R)$ also has a polynomial-time encoding algorithm. A discussion of efficient decoding algorithms can be found in the longer version of this work [8].

Note that in our construction of SRC codes we only guarantee that each individual segment satisfies the constraint, which does not necessarily imply the entire vector satisfies the constraint. However, as we shall see in the next section, for many well-studied constraints, it is still possible to guarantee that the entire vector satisfies the constraint by adding a small number of additional redundant bits.

IV. Applications of SRC Codes

In this section we present a few applications of SRC codes.

A. WOM-Codes

An $[n, k, t]$ *WOM code* $\mathcal{C}(\mathcal{E}, \mathcal{D})$ is a coding scheme consisting of n binary cells and is defined by an encoding map $\mathcal{E}$ and a decoding map $\mathcal{D}$. The WOM code guarantees any t writes of k-bit message while the cells can only change their value irreversibly from zero to one. If the WOM code can also correct e errors, then it is called an $[n, k, t]$ *e-error-correcting WOM code*. Several constructions of error-correcting WOM codes were presented in [12], which take advantage of BCH-like cyclic ECCs. For example, for triple-error-correcting WOM codes, the idea was to encode the three syndromes which correspond to the three roots of the ECC using another WOM code such that during decoding, these three roots will be used to correct at most three errors. However, this set of three roots had to satisfy the property that each root generates a single ECC and every pair of roots generates a double ECC. This property was called a *strong ECC*. Strong ECCs were found in [12] only for at most triple error correction codes and thus the construction of multiple-error-correcting WOM codes required a larger number of cells.

Using Goppa codes and their properties stated in Lemma 2 and Corollary 2 we are able to construct e-error-correcting WOM codes as follows. Let $R = \{\alpha_1, \alpha_2, \ldots, \alpha_e\} \subseteq \mathbb{F}_{2^m}$, and let $L = \{\gamma_1, \ldots, \gamma_{|L|}\} \subseteq \mathbb{F}_{2^m} \setminus R$. We use two WOM codes. The first one $\mathcal{C}_1(\mathcal{E}_1, \mathcal{D}_1)$ is an $[n, k, t]$ WOM code that is used to encode the information bits on each write. The second one $\mathcal{C}_2(\mathcal{E}_2, \mathcal{D}_2)$ is an $[n_0, \lceil \log(n) \rceil, t]$ WOM code that is used to encode each of the e segments on each write. In the encoding part we receive a message $\boldsymbol{u}$ of k bits that is encoded into n cells based upon $\boldsymbol{u}$ and the current memory state of the n cells. Denote the n encoded bits to the cells by $\boldsymbol{z} \in \mathbb{F}_2^n$. Next we generate the e redundancy segments $\boldsymbol{r}_i(\boldsymbol{z})$ for $i \in [e]$ by $\boldsymbol{r}_i(\boldsymbol{z}) = f_{\boldsymbol{z}}(\alpha_i) \in \mathbb{F}_{2^m}$, where $m = \lceil \log(n) \rceil$. Then, each of the e segments is encoded using the WOM code $\mathcal{C}_2(\mathcal{E}_2, \mathcal{D}_2)$ in order to receive the WOM encoded segment which we denote by $\rho_i(\boldsymbol{z})$. Lastly, we use one more cell for each segment to encode the parity $P_i = P(\rho_i(\boldsymbol{z}))$.

The number of cells which one has to use for this construction will be $n + e(n_0 + t)$. This construction significantly improves upon the construction of multiple (more than three)-error-correcting WOM codes from [12]. For example, the number of cells required by the construction from Theorem 14 in [12] is at least $\lceil e/2 \rceil n$ where n is the number of cells of a triple-error-correcting WOM codes. Note that in our construction the number of cells is significantly smaller than $2n$ for all constant values of e.

B. Balanced Codes

A balanced codebook of length n (where n is even) is one where each codeword in the codebook has $\frac{n}{2}$ ones. The best known construction in terms of redundancy for an error-correcting balanced code with an efficient encoder/decoder can be found in [3]. We summarize their result in the following theorem.

Theorem 2. *(cf., [3]) For $t \geq 1$, there exists a family of t-error-correcting balanced codes with at most $(t+1)\log n + 1$ bits of redundancy.*

In the following, we show the existence of t-error-correcting balanced codes with roughly $(t+\frac{1}{2})\log n + 2t + 2\log(t \log n) + 2$ redundant bits. This represents an improvement in the redundancy of roughly $\frac{1}{2}\log n$ bits for the case where t is a constant with respect to n. The complexity of our encoder is linear whereas our decoder has complexity which is polynomial with respect to n, since the decoder for our code relies on the decoder of an underlying Goppa code. One of the attractive features of our approach is that like [3], it does not require searching for a short balanced code.

To describe the construction, we need some tools that have been shown to exist from prior art. To begin, we require the use of a balanced encoder $\mathscr{E}_B : \mathbb{F}_2^N \to \mathbb{F}_2^{N + r_B(N)}$, where $r_B(N) = \frac{1}{2}\log N + \mathcal{O}(1)$ and $\mathcal{O}(1)$ denotes a constant at most equal to two for N large enough [11]. The input to $\mathscr{E}_B$ is an information vector of length N and the output is a balanced vector of length $N + r_B(N)$ which has an equal number of zeros and ones. An encoder can be constructed by inverting the role of the encoder/decoder of a classical arithmetic coding scheme and it has been shown that the resulting scheme has linear encoding complexity [11].

Let $\mathcal{Z} : \mathbb{F}_2^N \to \mathbb{F}_2^{N+1}$ be such that $\mathcal{Z}(\boldsymbol{v}) = (0, \boldsymbol{v})$ so that $\mathcal{Z}(\boldsymbol{v})$ is simply the result of inserting a zero into the first position of the output vector. Clearly, for any $\boldsymbol{v}' \in \mathbb{F}_2^N$, $\boldsymbol{v}' \neq \boldsymbol{v}$, we have

$$\overline{\mathcal{Z}(\boldsymbol{v})} \notin \left\{ \mathcal{Z}(\boldsymbol{v}'), \overline{\mathcal{Z}(\boldsymbol{v}')} \right\}, \tag{7}$$

where $\overline{(\mathcal{Z}(\boldsymbol{v}))}$ denotes the complement of $\mathcal{Z}(\boldsymbol{v})$. This property follows immediately since the first bit of $\mathcal{Z}(\boldsymbol{v})$ is always equal to zero and $\overline{\mathcal{Z}(\boldsymbol{v})} = \overline{\mathcal{Z}(\boldsymbol{v}')}$ if and only if $\boldsymbol{v} = \boldsymbol{v}'$.

Our scheme also makes use of a well-known technique by Knuth [9]. Suppose $\boldsymbol{v}$ is a binary vector of length N, and suppose $\sigma_j(\boldsymbol{v})$ is the result of inverting (or flipping) the first j bits of $\boldsymbol{v}$. Then there exists an index $b_K(\boldsymbol{v}) \in [N]$ such that $\sigma_{b_K(\boldsymbol{v})}(\boldsymbol{v})$ is balanced [9].

Our construction works by first using the encoder $\mathscr{E}_B$ to encode the information vector $\boldsymbol{u} \in \mathbb{F}_2^k$ and we denote the output of the encoder as $\boldsymbol{z} = \mathscr{E}_B(\boldsymbol{u})$ where $\boldsymbol{z}$ is a balanced vector. Next we generate our evaluations $f_{\boldsymbol{z}}(\alpha_1), f_{\boldsymbol{z}}(\alpha_2), \ldots, f_{\boldsymbol{z}}(\alpha_t)$, and we pass each of these evaluations through $\mathcal{Z}$ to obtain $\mathcal{Z}(f_{\boldsymbol{z}}(\alpha_1)), \mathcal{Z}(f_{\boldsymbol{z}}(\alpha_2)), \ldots, \mathcal{Z}(f_{\boldsymbol{z}}(\alpha_t))$. Recall that we can represent the output of the function $f_{\boldsymbol{z}}$ using m bits according to (4) where $m = \log(k + r_B(k) + t)$,

and so $\mathcal{Z}(f_{\boldsymbol{z}}(\alpha_i)) \in \mathbb{F}_2^{m+1}$. Let P_i be the same as before where $P_i = P\Big(\mathcal{Z}(f_{\boldsymbol{z}}(\alpha_i))\Big) \in \mathbb{F}_2$ is the parity of the vector $\mathcal{Z}(f_{\boldsymbol{z}}(\alpha_i))$. Note that at this point we cannot simply append the information $\boldsymbol{R}(\boldsymbol{z})$ to $\boldsymbol{z}$ where $\boldsymbol{R}(\boldsymbol{z}) = \Big(\mathcal{Z}(f_{\boldsymbol{z}}(\alpha_1)), P_1, \ldots, \mathcal{Z}(f_{\boldsymbol{z}}(\alpha_t)), P_t\Big) \in \mathbb{F}_2^{t(m+2)}$ since the resulting vector may not be balanced. In order to ensure our codewords are balanced, the idea is to use Knuth's technique (and in particular the function b_K) to maintain the balancing property. As will be detailed in the proof of Theorem 3 (available at [8]), despite the fact that we are flipping the bits in $\boldsymbol{R}(\boldsymbol{z})$ according to Knuth's technique, the resulting code will still maintain its minimum distance.

Our construction proceeds as follows. Let $\boldsymbol{u} \in \mathbb{F}_2^k$ be an information sequence and suppose $\boldsymbol{z} = \mathscr{E}_B(\boldsymbol{u})$ and let $f_{\boldsymbol{z}}(\alpha_1), f_{\boldsymbol{z}}(\alpha_2), \ldots, f_{\boldsymbol{z}}(\alpha_t)$, $P_1, \ldots, P_t$ and $\boldsymbol{R}(\boldsymbol{z})$ be as defined in the previous paragraph. Suppose $b_K\Big(\boldsymbol{R}(\boldsymbol{z})\Big) = \ell$ where $\ell \in [t(m+2)]$. Let $B : [t(m+2)] \to \mathbb{F}_2^{\lceil \log t(m+2) \rceil}$ be an invertible function which gives the binary image.

Then our error-correcting balanced code is defined:

$$C_B(n,t) := \Bigg\{ \boldsymbol{x} \in \mathbb{F}_2^n : \boldsymbol{u} \in \mathbb{F}_2^k, \boldsymbol{z} = \mathscr{E}_B(\boldsymbol{u}), \quad \boldsymbol{x} = \Big(\boldsymbol{z}, \sigma_\ell(\boldsymbol{R}(\boldsymbol{z})), B(\ell), \overline{B(\ell)}, P_\ell, \overline{P_\ell}\Big) \Bigg\}, \tag{8}$$

where in the previous equation $P_\ell = P(B(\ell)) \in \mathbb{F}_2$.

Note that since $|\boldsymbol{z}| = k + r_B(k)$, $|\boldsymbol{R}(\boldsymbol{z})| = t(m+2)$, $|B(\ell)| = \lceil \log t(m+2) \rceil$, and $m = \log(|\boldsymbol{z}| + t)$, it follows that $n = k + r_B(k) + t(m+2) + 2\lceil \log t(m+2) \rceil + 2$. Since the number of information bits is k, it follows that the code $C_B(n,t)$ has $(t+\frac{1}{2})\log n + 2\log\log n + \mathcal{O}(1)$ redundant bits when $t = \mathcal{O}(1)$. It is straightforward to verify $\boldsymbol{x}$ is balanced since $\boldsymbol{z}$, $\sigma_\ell(\boldsymbol{R}(\boldsymbol{z}))$, $(B(\ell), \overline{B(\ell)})$, and $(P_\ell, \overline{P_\ell})$ are each balanced. We are now have the following result.

Theorem 3. *Suppose $m+1$ is odd. Then the code $C_B(n,t)$ is a t-error-correcting balanced code.*

C. Run-Length Limited Codes

In the following, we consider the design of a special type of error-correcting run-length limited codebook. We will be interested in the setup where our codewords do not include runs of ones of length L. For shorthand, we will refer to such codes as t-error-correcting L-RLL codes or t-error-correcting RLL codes when the meaning of L is clear. Motivated by applications in DNA storage [10], we will be interested in the setup where $L = \mathcal{O}(\log n)$.

In [10], the authors describe a linear-time encoding/decoding algorithm for an L-RLL code where $L = \lceil \log n \rceil + 1$. The codes from [10] require only a single bit of redundancy. In the following, we introduce a family of t-error-correcting RLL codes that require $t\log n + t + 2$ bits of redundancy where $L = 2(1 + \log n)$. Our codes have linear-time encoding complexity and polynomial-time decoding complexity. We compare this approach with an application of the traditional reverse concatenation method and we show that for the case where t is a constant, our codes require approximately $\mathcal{O}(\log\log n)$ less bits of redundancy than the traditional reverse concatenation method. Furthermore, our approach, unlike traditional reverse concatenation, does not require searching for a short error-correcting RLL code.

We need some notation and existing results to describe our code construction. Let $\mathscr{E}_{LY} : \mathbb{F}_2^N \to \mathbb{F}_2^{N+1}$ be the encoder from [10] which takes as input an arbitrary data vector of length N and it outputs a sequence of length $N+1$ that does not contain the substring $\mathbf{1}^{\lceil \log N \rceil + 1}$.

Our t-error correcting RLL code $C_{RL}(n,t)$ is defined

$$C_{RL}(n,t) := \Big\{ \boldsymbol{x} \in \mathbb{F}_2^n : \boldsymbol{u} \in \mathbb{F}_2^k, \boldsymbol{z} = \mathscr{E}_{LY}(\boldsymbol{u}), \quad \boldsymbol{x} = \Big(\boldsymbol{z}, 0, f_{\boldsymbol{z}}(\alpha_1), P_1, \ldots, f_{\boldsymbol{z}}(\alpha_t), P_t\Big). \tag{9}$$

As in the previous subsection, we assume that $f_{\boldsymbol{z}}(\alpha_i) \in \mathbb{F}_2^m$ can be expressed using m bits where $m = \log(k+1+t) < \log n$ and that $P_i = P(f_{\boldsymbol{z}}(\alpha_i))$.

Theorem 4. *Suppose m is an even integer. The code $C_{RL}(n,t)$ is a t-error-correcting L-RLL code for $L = 2m+1$.*

Next, we consider a construction for a t-error-correcting L-RLL code using the traditional reverse concatenation method. The code is constructed by applying the encoder $\mathscr{E}_{LY}$ on the systematic portion of the systematic ECC. Next we need to generate a codeword to encode $t\log n$ bits of information from the non-systematic portion of the ECC and the codeword is taken from a short L-RLL code which also has minimum distance $2t+1$. As mentioned before, such a code is oftentimes generated using a brute-force search. The redundancy of the resulting construction is at least $1 + t\log n + t\log(t\log n))$ which in many instances is more than the redundancy required for $C_{RL}(n,t)$. In addition, our construction enjoys the benefit of not requiring a short error-correcting RLL code.

REFERENCES

[1] E. Berklekamp, "Goppa codes," *IEEE Transactions on Information Theory*, vol. 19, no. 5, pp. 590-592, 1973.

[2] W.G. Bliss, "Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation," *IBM Tech. Discl. Bull.*, vol. 23, pp. 4633-4634, 1981.

[3] Y.M. Chee, H.M. Kiah, and H. Wei, "Efficient and explicit balanced primer codes," *IEEE Symposium on Information Theory*, Paris, 2019.

[4] J. Fan and R. Calderbank, "A modified concatenated coding scheme with applications to magnetic data storage," *IEEE Trans. on Inform. Theory*, vol. 44, no. 4, pp. 1565-1574, 1998.

[5] K.A.S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inform. Theory*, vol. 43, no. 5, pp. 1389-1399, 1997.

[6] M. Mansuripur, "Enumerative modulation coding with arbitrary constraints and post-modulation error correction coding and data storage systems," *Proc. SPIE*, vol. 1499, pp. 72-86, 1991.

[7] B.H. Marcus, R.M. Roth,and P.H. Siegel, An introduction to coding for constrained systems, 2001.

[8] R. Gabrys, P.H. Siegel, and E. Yaakobi, "Segmented reverse concatenation: a new approach to constrained ecc," available on *arXiv*.

[9] D.E. Knuth, "Efficient balanced codes," *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 51-53, 1986.

[10] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Transactions on Information Theory*, vol. 6, no. 65, 2019.

[11] T. V. Ramabadran, "A coding scheme for m-out-of-n codes," *IEEE Transactions on Communications*, vol. 38, no. 8, pp. 1156-1163, 1990.

[12] E. Yaakobi, P.H. Siegel, A. Vardy, J.K. Wolf, "Multiple error-correcting WOM-codes," vol. 58, no. 4, pp. 2220-2230, 2012.