On Coding Over Sliced Information

Jin Sima, **Netanel Raviv**, and **Jehoshua Bruck**
Electrical Engineering, California Institute of Technology, U.S.A., {jsima,bruck}@caltech.edu

***Abstract*—The interest in channel models in which the data is sent as an unordered set of binary strings has increased lately, due to emerging applications in DNA storage, among others. In this paper we analyze the minimal redundancy of binary codes for this channel under substitution errors, and provide a code construction for a single substitution that is shown to be asymptotically optimal up to constants. The surprising result in this paper is that while the information vector is sliced into a set of unordered strings, the amount of redundant bits that are required to correct errors is orderwise equivalent to the amount required in the classical error correcting paradigm.**

I. Introduction

Data storage in synthetic DNA molecules suggests unprecedented advances in density and durability. The interest in DNA storage has increased dramatically in recent years, following a few successful prototype implementations [1], [4], [11]. However, due to biochemical restrictions in synthesis (i.e., writing) and sequencing (i.e., reading), the underlying channel model of DNA storage systems is fundamentally different from its digital-media counterpart.

Typically, the data in a DNA storage system is stored as a pool of short strings that are dissolved inside a solution, and consequently, these strings are obtained at the decoder in an *unordered* fashion. Furthermore, current technology does not allow the decoder to count the exact number of appearances of each string in the solution, but merely to estimate relative concentrations. These restrictions have re-ignited the interest in *coding over sets*, a model that also finds applications in transmission over asynchronous networks (see Section III).

In this model, the data to be stored is encoded as a set of M strings of length L over a certain alphabet Σ, for some integers M and L such that $M < |\Sigma|^L$; typical values for M and L are currently within the order of magnitude of 10^7 and 10^2, respectively [11], and each individual string is subject to deletions, insertions, and substitutions.

In the context of DNA storage, after encoding the data as a set of strings over a four-symbol alphabet, the corresponding DNA molecules are synthesized and dissolved inside a solution. Then, a chemical process called *Polymerase Chain Reaction* (PCR) is applied, which drastically amplifies the number of copies of each string. In the reading process, strings whose length is either shorter or longer than L are discarded, and the remaining ones are clustered according to their respective edit-distance. Then, a majority vote is held within each cluster in order to come up with the most likely origin of the reads in that cluster, and all majority winners are included in the *output set* of the decoding algorithm.

The work was supported in part by NSF grants CCF-1816965 and CCF-1717884 .

One of the caveats of this approach is that errors in *synthesis* might cause the PCR process to amplify a string that was written erroneously, and hence the decoder might include this erroneous string in the output set. In this context, deletions and insertions are easier to handle since they result in a string of length different from[1] L. Substitution errors, however, are more challenging to combat, and are discussed next.

A substitution error that occurs prior to amplification by PCR can induce either one of two possible error patterns. In one, the newly created string already exists in the set of strings, and hence, the decoder will output a set of $M-1$ strings. In the other, which is undetectable by counting the size of the output set, the substitution generates a string which is not equal to any other string in the set. In this case the output set has the same size as the error free one. These error patterns, which are referred to simply as *substitutions*, are the main focus of this paper.

A formal definition of the channel model and a summary of our contributions are given in Section II, and previous work is discussed in Section III. Upper and lower bounds on the amount of redundant bits that are required to combat substitutions are given in Section IV. In Section V we provide a construction of a code that can correct a *single* substitution, which is shown to be optimal up to some constant.

Remark 1. *An improvement of the code construction in Section V, as well as its generalization to multiple substitutions and many omitted details, are readily available online [6].*

Remark 2. *It follows from the sphere-packing bound [12, Sec. 4.2] that without the slicing operation, one must introduce at least $K\log(N)$ redundant bits at the encoder in order to combat K substitutions in a code of length N. The surprising result of this paper, is that the slicing operation* does not *incur a substantial increase in the amount of redundant bits that are required to correct K substitutions.*

II. Preliminaries and Contributions

For integers M and L such that $M \leq 2^L$ we denote by $\binom{\{0,1\}^L}{M}$ the family of all subsets of size M of $\{0,1\}^L$. In our channel model, a *word* is an element $W \in \binom{\{0,1\}^L}{M}$, and a *code* $\mathcal{C} \subseteq \binom{\{0,1\}^L}{M}$ is a set of words (for clarity, we refer to words in a given code as *codewords*). To prevent ambiguity with classical coding theoretic terms, the elements in a word $W = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ are referred to as *strings*. We emphasize that the indexing in W is merely a notational convenience, e.g., by the lexicographic order of the strings, and this information is not available at the decoder.

[1] As long as the number of insertions is not equal to the number of deletions, an event that occurs in negligible probability.

A *substitution error* (substitution, in short), is an operation that changes the value of a position in a word. Notice that the result of a substitution is not necessarily an element of $\binom{\{0,1\}^L}{M}$, and might be an element of $\binom{\{0,1\}^L}{M-1}$. Similarly, K substitutions may result in an element of $\binom{\{0,1\}^L}{T}$ for some $M-K \leq T \leq M$. This gives rise to the following definition.

Definition 1. *For a word $W \in \binom{\{0,1\}^L}{M}$, a ball $\mathcal{B}_K(W) \subseteq \bigcup_{j=M-K}^{M} \binom{\{0,1\}^L}{j}$ centered at W is the collection of all subsets of $\{0,1\}^L$ that can be obtained by at most K-substitutions in W.*

Example 1. *For $M = 2$, $L = 3$, $K = 1$, and $W = \{001, 011\}$, we have that*

$$\mathcal{B}_K(W) = \{\{001, 101\}, \{101, 011\}, \{011\}, \{000, 011\}, \{001, 111\}, \{001\}, \{001, 010\}\}.$$

In this paper, we discuss bounds and constructions of codes in $\binom{\{0,1\}^L}{M}$ that can correct K substitutions (K-substitution codes, for short), for various values of K. The *size* of a code, which is denoted by $|\mathcal{C}|$, is the number of codewords (that is, sets) in it. The *redundancy* of the code, a quantity that measures the amount of redundant information that is to be added to the data to guarantee successful decoding, is defined as $r(\mathcal{C}) \triangleq \log \binom{2^L}{M} - \log(|\mathcal{C}|)$, where the logarithms are in base 2.

A code $\mathcal{C}$ is used in our channel as follows. First, the data to be stored (or transmitted) is mapped by a bijective *encoding function* to a codeword $C \in \mathcal{C}$. This codeword passes through a channel that might introduce up to K substitutions, and as a result a word $W \in \mathcal{B}_K(C)$ is obtained at the decoder. In turn, the decoder applies some *decoding function* to extract the original data. The code $\mathcal{C}$ is called a K-substitution code if the decoding process always recovers the original data successfully. Having settled the channel model, we are now in a position to formally state our contribution.

Theorem 1. *(Main) For any integers M, L, and K such that $M \leq 2^{L/(4K+2)}$, there exists an explicit code construction with redundancy $O(K^2 \log(ML))$ ([6, Sec. VI]). For $K = 1$, the redundancy of this construction is at most six times larger than the optimal one (Section V). Furthermore, an improved construction for $K = 1$ achieves redundancy which is at most three times the optimal one ([6, App. C]).*

A few auxiliary notions are used throughout the paper, and are introduced herein. For two strings $\mathbf{s}, \mathbf{t} \in \{0,1\}^L$, the *Hamming distance* $d_H(\mathbf{s}, \mathbf{t})$ is the number of entries in which they differ. To prevent confusion with common terms, a subset of $\{0,1\}^L$ is called *a vector-code*, and the set $\mathcal{B}_D^H(\mathbf{s})$ of all strings within Hamming distance D or less of a given string $\mathbf{s}$ is called the *Hamming ball* of radius D centered at $\mathbf{s}$. The well-known Hamming code is used in the sequel; this is a $[2^t-1, 2^t-t-1]_2$ code (i.e., a linear subspace of $GF(2)^{2^t-1}$ of dimension $2^t - t - 1$) of minimum distance 3. Generally, we denote scalars by lower-case letters $x, y, \ldots$, vectors by bold symbols $\mathbf{x}, \mathbf{y}, \ldots$, integers by capital letters $K, L, \ldots$, and $[K] \triangleq \{1, 2, \ldots, K\}$.

III. Previous Work

The channel model in this work has been studied by several authors in the past. The work of [5] addressed this channel model under the restriction that individual strings are read in an error free manner, and some strings might get lost as a result of random sampling of the DNA pool. In their techniques, the strings in a codeword are appended with an indexing prefix, a solution which already incurs $\Theta(M \log M)$ redundant bits, or $\log(e)M - o(1)$ redundancy [9, Remark 1], and will be shown to be strictly sub-optimal in our case.

The recent work of [9] addressed this model under substitutions, deletions, and insertions. When discussing substitutions only, [9] suggested a code construction for $K = 1$ with $2L+1$ bits of redundancy. Furthermore, by using a reduction to constant Hamming weight vector-codes, it is shown that there exists a code that can correct e errors in each one of the M sequences with redundancy $Me \log(L+1)$.

The work of [7] addressed a similar model, where *multisets* are received at the decoder, rather than sets. In addition, errors in the stored strings are not seen in a fine-grained manner. That is, any set of errors in an individual string is counted as a single error, regardless of how many substitutions, insertions, or deletions it contains. As a result, the specific structure of $\{0,1\}^L$ is immaterial, and the problem reduces to decoding *histograms* over an alphabet of a certain size.

The specialized reader might suggest the use of *fountain codes*, such as the LT [10] codes. However, we stress that these solutions rely on randomness at much higher redundancy, whereas this work aims for a deterministic and rigorous solution at redundancy which is close to optimal.

Finally, we also mention the *permutation channel* (e.g., [8], [13]), which is similar to our setting, and yet it is farther away in spirit than the aforementioned works. In that channel, a vector over a certain alphabet is transmitted, and its symbols are received at the decoder under a certain permutation. If no restriction is applied over the possible permutations, then this channel reduces to *multiset decoding*, as in [7]. This channel is applicable in networks in which different packets are routed along different paths of varying lengths, and are obtained in an unordered and possibly erroneous form at the decoder. Yet, this line of works is less relevant to ours, and to DNA storage in general, since the specific error pattern in each "symbol" (which corresponds to a string in $\{0,1\}^L$ in our case) is not addressed, and perfect knowledge of the number of appearances of each "symbol" is assumed.

IV. Bounds

In this section we use sphere packing arguments in order to establish an existence result of codes with low redundancy, and a lower bound on the redundancy for any single substitution code. The latter bound demonstrates the asymptotic optimality of the construction in Section V. Both of these bounds rely on upper and lower bounds on the size of the ball $\mathcal{B}_K$ (Definition 1), which are given below.

Lemma 1. *For every word $W = \{\mathbf{x}_i\}_{i=1}^M \in \binom{\{0,1\}^L}{M}$ and every positive integer $K \leq ML$, we have that $|\mathcal{B}_K(W)| \leq \sum_{\ell=0}^{K} \binom{ML}{\ell}$.*

Proof. Every word in $\mathcal{B}_K(W)$ is obtained by flipping the bits in $\mathbf{x}_i$ that are indexed by some $J_i \subseteq [L]$, for every $i \in [M]$, where $\sum_{i=1}^{M} |J_i| \leq K$. Clearly, there are at most $\sum_{\ell=0}^{K} \binom{ML}{\ell}$ ways to choose the index sets $\{J_i\}_{i=1}^{M}$. □

A careful application of sphere-packing arguments that follow from Lemma 1 results in the following [6, Sec. IV.A].

Corollary 1. *For every K, M, and L such that $M < 2^L$ and $K \leq ML$, there exists a code $\mathcal{C} \subseteq \binom{\{0,1\}^L}{M}$ whose redundancy is at most $2K\log(ML) + K\log(M)$.*

Notice that the bound in Lemma 1 is tight, e.g., in cases where $d_H(\mathbf{x}_i, \mathbf{x}_j) \geq 2K+1$ for all distinct $i, j \in [M]$. This might occur only if M is less than the maximum size of a K-substitution correcting vector-code, i.e., when $M \leq 2^L / (\sum_{i=0}^{K} \binom{L}{i})$ [12, Sec. 4.2]. When the minimum Hamming distance between the strings in a codeword is not large enough, different substitution errors might result in identical words, and the size of the ball is smaller than the given upper bound.

Example 2. *For $L = 4$ and $M = 2$, consider the word $W = \{\underline{0}110, \mathbf{0}11\underline{\mathbf{1}}\}$. By flipping either the two underlined symbols, or the two bold symbols, the word $W' = \{0110, 1110\}$ is obtained. Hence, different substitution operation might result in identical words.*

However, in some cases it is possible to bound the size of $\mathcal{B}_K$ from below by using tools from Fourier analysis of Boolean functions. In the following it is assumed that $M \leq 2^{(1-\epsilon)L}$ for some $0 < \epsilon < 1$, and that $K = 1$. A word $W \in \binom{\{0,1\}^L}{M}$ corresponds to a Boolean function $f_W : \{\pm 1\}^L \to \{\pm 1\}$ in a natural way. For $\mathbf{x} \in \{0,1\}^L$ let $\overline{\mathbf{x}} \in \{\pm 1\}^L$ be the vector which is obtained from $\mathbf{x}$ be replacing every 0 by 1 and every 1 by -1. Then, we define $f_W(\overline{\mathbf{x}}) = -1$ if $\mathbf{x} \in W$, and 1 otherwise. Considering the set $\{\pm 1\}^L$ as the *hypercube graph*[2], the *boundary* of f_W is the set of all edges $\{\mathbf{x}_1, \mathbf{x}_2\} \in \binom{\{\pm 1\}^L}{2}$ in this graph such that $f_W(\mathbf{x}_1) \neq f_W(\mathbf{x}_2)$. The following lemma is easy to prove, and its proof is omitted.

Lemma 2. *The size of $\mathcal{B}_1(W)$ is at least as the size of the boundary of f_W.*

Notice that the bound in Lemma 2 is tight, e.g., in cases where the minimum Hamming distance between the strings of W is at least 2. This implies the tightness of the bound which is given below in these cases. The following Fourier analytic claims will aid in proving a lower bound. Let the *total influence* of f_W be $I(f_W) \triangleq \sum_{i=1}^{L} \Pr_{\mathbf{x}}(f_W(\mathbf{x}) \neq f_W(\mathbf{x}^{\oplus i}))$, where $\mathbf{x}^{\oplus i}$ is obtained from $\mathbf{x}$ by changing the sign of the i-th entry, and $\mathbf{x} \in \{\pm 1\}^L$ is chosen uniformly at random.

Lemma 3. *[2, Theorem 2.39] For every function $f : \{\pm 1\}^L \to \mathbb{R}$, we have that $I(f) \geq 2\alpha\log(1/\alpha)$, where $\alpha = \alpha(f) \triangleq \min\{\Pr_{\mathbf{x}}(f(\mathbf{x}) = 1), \Pr_{\mathbf{x}}(f(\mathbf{x}) = -1)\}$, and $\mathbf{x} \in \{\pm 1\}^L$ is chosen uniformly at random.*

Lemma 4. *For every word $W \in \binom{\{0,1\}^L}{M}$ we have that $|\mathcal{B}_1(W)| \geq \epsilon ML$.*

[2]The nodes of the hypercube graph of dimension L are identified by $\{\pm 1\}^L$, and every two nodes are connected if and only if the Hamming distance between them is 1.

Proof. Since $M \leq 2^{(1-\epsilon)L}$ and $\alpha = \alpha(f_W) = \min\{(2^L - M)/2^L, M/2^L\}$, it follows that $\alpha = M/2^L$ for reasonably large values of L. In addition, since $\Pr_{\mathbf{x}}(f_W(\mathbf{x}) \neq f_W(\mathbf{x}^{\oplus i}))$ equals the fraction of dimension i edges that lie on the boundary of f_W ([2, Fact 2.14]), Lemma 2 implies that

$$I(f_W) = \frac{\text{the size of the boundary of } f_W}{2^{L-1}} \leq \frac{|\mathcal{B}_1(W)|}{2^{L-1}}.$$

Therefore, from Lemma 3 we have that $|\mathcal{B}_1(W)| \geq 2\alpha\log(1/\alpha) \cdot 2^{L-1} = M(L - \log(M)) \geq \epsilon ML$. □

Corollary 2. *For integers L and M and a constant $0 < \epsilon < 1$ such that $M \leq 2^{(1-\epsilon)L}$, a 1-substitution code $\mathcal{C} \subseteq \binom{\{0,1\}^L}{M}$ satisfies that $r(\mathcal{C}) = \log(ML) - O(1)$.*

Proof. According to Lemma 4, every codeword of every $\mathcal{C}$ excludes at least ϵML other words from belonging to $\mathcal{C}$. Hence, we have that $|\mathcal{C}| \leq \binom{2^L}{M} / \epsilon ML$, and by the definition of redundancy, it follows that

$$\begin{aligned} r(\mathcal{C}) &= \log\binom{2^L}{M} - \log(|\mathcal{C}|) \\ &\geq \log(\epsilon ML) = \log(ML) - O(1). \end{aligned}$$ □

V. Codes for a Single Substitution

In this section we present a 1-substitution code construction that applies whenever $M \leq 2^{L/6}$, whose redundancy is at most $3\log ML + 3\log M + O(1)$. For simplicity of illustration, we restrict our attention to values of M and L such that $\log ML + \log M \leq M$. In the remaining values, a similar construction of comparable redundancy exists [6].

Theorem 2. *For $\mathcal{D} = \{1, \ldots, \binom{2^{L/3-1}}{M}^3 \cdot (M!)^2 \cdot 2^{3M - 3\log ML - 3\log M - 6}\}$, there exist an encoding function $E : D \to \binom{\{0,1\}^L}{M}$ whose image is a single substitution correcting code.*

The idea behind Theorem 2 is to concatenate the strings in a codeword $C = \{\mathbf{x}_i\}_{i=1}^{M}$ in a certain order, so that classic 1-substitution error correction techniques can be applied over the concatenated string. Since a substitution error may affect any particular order of the $\mathbf{x}_i$'s, we consider the lexicographic orders of several different parts of the $\mathbf{x}_i$'s, instead of the lexicographic order of the whole strings. Specifically, we partition the $\mathbf{x}_i$'s to three parts, and place distinct strings in each of them. Since a substitution operation can scramble the order in at most one part, the correct order will be inferred by a majority vote, so that classic substitution error correction can be applied.

Consider a message $d \in D$ as a tuple $d = (d_1, \ldots, d_6)$, where $d_1 \in [\binom{2^{L/3-1}}{M}]$, $d_3, d_5 \in [\binom{2^{L/3-1}}{M} M!]$, and $d_2, d_4, d_6 \in [2^{M - \log ML - \log M - 2}]$. Let F_1 (resp. F_2) be any injective function that maps $[\binom{2^{L/3-1}}{M}]$ (resp. $[\binom{2^{L/3-1}}{M} M!]$) to $\binom{\{0,1\}^{L/3-1}}{M}$ (resp. $\binom{\{0,1\}^{L/3-1}}{M} \times S_M$, where S_M is the symmetric group on M elements). In addition, let

$$\begin{aligned} F_1(d_1) &= \{\mathbf{a}_1, \ldots, \mathbf{a}_M\}, \\ F_2(d_3) &= (\{\mathbf{b}_1, \ldots, \mathbf{b}_M\}, \sigma), \\ F_2(d_5) &= (\{\mathbf{c}_1, \ldots, \mathbf{c}_M\}, \pi), \end{aligned} \tag{1}$$

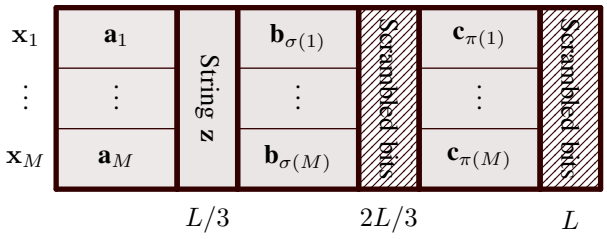

Fig. 1. An illustration of the encoding scheme in Section V, where the string $\mathbf{z}$ equals $(\mathbf{d}_2, E_H(\mathbf{d}_2), E_H(\mathbf{s}_1))$. The strings $\mathbf{x}_1, \ldots, \mathbf{x}_M$ are shown sorted according to their $\mathbf{a}_i$ entries, and hence the content of column $L/3$ is $\mathbf{z}$, and columns $2L/3$ and L are scrambled. If the strings $\{\mathbf{x}_i\}_{i=1}^M$ were to be sorted by their $\mathbf{b}_i$ entries, then column $2L/3$ would contain $(\mathbf{d}_4, E_H(\mathbf{d}_4), E_H(\mathbf{s}_2))$, and if they were to be sorted by their $\mathbf{c}_i$ entries, then column L would contain $(\mathbf{d}_6, E_H(\mathbf{d}_6), E_H(\mathbf{s}_3))$.

where $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i \in \{0,1\}^{L/3-1}$ for every $i \in [M]$, the permutations σ and π are in S_M, and the indexing of $\{\mathbf{a}_i\}_{i=1}^M$, $\{\mathbf{b}_i\}_{i=1}^M$, and $\{\mathbf{c}_i\}_{i=1}^M$ is lexicographic. Finally, let $\mathbf{d}_2, \mathbf{d}_4$, and $\mathbf{d}_6$ be the binary strings that correspond to d_2, d_4, and d_6, respectively, and define

$$\begin{aligned}
\mathbf{s}_1 &= (\mathbf{a}_i)_{i=1}^M \circ (\mathbf{b}_{\sigma(i)})_{i=1}^M \circ (\mathbf{c}_{\pi(i)})_{i=1}^M \\
\mathbf{s}_2 &= (\mathbf{a}_{\sigma^{-1}(i)})_{i=1}^M \circ (\mathbf{b}_i)_{i=1}^M \circ (\mathbf{c}_{\sigma^{-1}\pi(i)})_{i=1}^M \\
\mathbf{s}_3 &= (\mathbf{a}_{\pi^{-1}(i)})_{i=1}^M \circ (\mathbf{b}_{\pi^{-1}\sigma(i)})_{i=1}^M \circ (\mathbf{c}_i)_{i=1}^M,
\end{aligned} \tag{2}$$

where $\circ$ denotes concatenation.

Without loss of generality assume that there exists an integer t for which $|\mathbf{s}_i| = (L-3)M = 2^t - t - 1$ for all $i \in [3]$. Then, each $\mathbf{s}_i$ can be encoded by using a *systematic* $[2^t - 1, 2^t - t - 1]_2$ Hamming code, by introducing t redundant bits. That is, the encoding function is of the form $\mathbf{s}_i \mapsto (\mathbf{s}_i, E_H(\mathbf{s}_i))$, where $E_H(\mathbf{s}_i)$ are the t redundant bits, and $t \leq \log(ML) + 1$. Similarly, we assume that there exists an integer h for which $|\mathbf{d}_i| = 2^h - h - 1$ for $i \in \{2,4,6\}$, and let $E_H(\mathbf{d}_i)$ be the corresponding h bits of redundancy, that result from encoding $\mathbf{d}_i$ by using a $[2^h - 1, 2^h - h - 1]_2$ Hamming code. By the properties of the Hamming code we have that $h \leq \log(M) + 1$. The data $d \in D$ is mapped to a codeword $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ as follows (see Fig. 1). First, we place $\{\mathbf{a}_i\}_{i=1}^M$, $\{\mathbf{b}_i\}_{i=1}^M$, and $\{\mathbf{c}_i\}_{i=1}^M$ in the different thirds of the $\mathbf{x}_i$'s, sorted by σ and π. That is, denoting $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,L})$, we define

$$\begin{aligned}
(x_{i,1}, \ldots, x_{i,L/3-1}) &= \mathbf{a}_i, \\
(x_{i,L/3+1}, \ldots, x_{i,2L/3-1}) &= \mathbf{b}_{\sigma(i)}, \text{ and} \\
(x_{i,2L/3+1}, \ldots, x_{i,L-1}) &= \mathbf{c}_{\pi(i)}.
\end{aligned} \tag{3}$$

The remaining bits $\{x_{i,L/3}\}_{i=1}^M$, $\{x_{i,2L/3}\}_{i=1}^M$, and $\{x_{i,L}\}_{i=1}^M$ are used to accommodate the information bits of $\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_6$, and the redundancy bits $\{E_H(\mathbf{s}_i)\}_{i=1}^3$ and $\{E_H(\mathbf{d}_i)\}_{i\in\{2,4,6\}}$, in the following manner.

Consider the codeword as an $M \times L$ matrix, in which rows correspond to the $\mathbf{x}_i$'s. The remaining bits $\{x_{i,L/3}\}_{i=1}^M$, $\{x_{i,2L/3}\}_{i=1}^M$, and $\{x_{i,L}\}_{i=1}^M$ correspond to columns $L/3$, $2L/3$, and L in this matrix. We begin by sorting the rows only according to the $\mathbf{a}_i$ values, i.e., the values in columns $1, \ldots, L/3-1$, and placing $(\mathbf{d}_2, E_H(\mathbf{d}_2), E_H(\mathbf{s}_1))^\top$ in column $L/3$. Then, we sort by the $\mathbf{b}_i$ values, i.e., the values in columns $L/3+1, \ldots, 2L/3-1$, and place $(\mathbf{d}_4, E_H(\mathbf{d}_4), E_H(\mathbf{s}_2))^\top$ in column $2L/3$. Finally, we sort by the $\mathbf{c}_i$ values, and place $(\mathbf{d}_6, E_H(\mathbf{d}_6), E_H(\mathbf{s}_3))^\top$ in column L.

That is, if the strings $\{\mathbf{x}_i\}_{i=1}^M$ are sorted according to the content of the bits $(x_{i,1}, \ldots, x_{i,L/3-1}) = \mathbf{a}_i$, then the top $M - \log ML \log M - 2$ bits of the $(L/3)$'th column contain $\mathbf{d}_2$, the middle $\log M + 1$ bits contain $E_H(\mathbf{d}_2)$, and the bottom $\log ML + 1$ bits contain $E_H(\mathbf{s}_1)$. Similarly, if the strings are sorted according to $(x_{i,L/3+1}, \ldots, x_{i,2L/3-1}) = \mathbf{b}_i$, then the top $M - \log ML \log M - 2$ bits of the $(2L/3)$'th column contain $\mathbf{d}_4$, the middle $\log M + 1$ bits contain $E_H(\mathbf{d}_4)$, and the bottom $\log ML + 1$ bits contain $E_H(\mathbf{s}_2)$, and so on. This concludes the encoding function E of Theorem 2. It can be readily verified that E is injective since different messages result in either different $(\{\mathbf{a}_i\}_{i=1}^M, \{\mathbf{b}_i\}_{i=1}^M, \{\mathbf{c}_i\}_{i=1}^M)$ or the same $(\{\mathbf{a}_i\}_{i=1}^M, \{\mathbf{b}_i\}_{i=1}^M, \{\mathbf{c}_i\}_{i=1}^M)$ with different $(\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_6)$. In either case, the resulting codewords $\{\mathbf{x}_i\}_{i=1}^M$ of the two messages are different.

To verify that the image of E is a 1-substitution code, observe first that since $\{\mathbf{a}_i\}_{i=1}^M$, $\{\mathbf{b}_i\}_{i=1}^M$, and $\{\mathbf{c}_i\}_{i=1}^M$ are *sets*, it follows that any two strings in the same set are distinct. Hence, according to (3), it follows that $d_H(\mathbf{x}_i, \mathbf{x}_j) \geq 3$ for every distinct i and j in $[M]$. Therefore, no 1-substitution error can cause one $\mathbf{x}_i$ to be equal to another, and consequently, the result of a 1-substitution error is always in $\binom{\{0,1\}^L}{M}$. In what follows a decoding algorithm is presented, whose input is a codeword that was distorted by at most a single substitution, and its output is d.

Upon receiving a word $C' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_M\} \in \mathcal{B}_1(C)$ for some codeword C we define

$$\begin{aligned}
\hat{\mathbf{a}}_i &= (x'_{i,1}, \ldots, x'_{i,L/3-1}) \\
\hat{\mathbf{b}}_i &= (x'_{\tau^{-1}(i),L/3+1}, \ldots, x'_{\tau^{-1}(i),2L/3-1}) \\
\hat{\mathbf{c}}_i &= (x'_{\rho^{-1}(i),2L/3+1}, \ldots, x'_{\rho^{-1}(i),L-1}),
\end{aligned} \tag{4}$$

where τ is the permutation by which $\{\mathbf{x}'_i\}_{i=1}^M$ are sorted according to their $L/3+1, \ldots, 2L/3-1$ entries, and ρ is the permutation by which they are sorted according to their $2L/3+1, \ldots, L-1$ entries (we emphasize that τ and ρ are unrelated to the original π and σ, and those will be decoded later). Further, when ordering $\{\mathbf{x}'_i\}_{i=1}^M$ by either the lexicographic ordering, by τ, or by ρ, we obtain *candidates* for each one of $\mathbf{d}_2$, $\mathbf{d}_4$, $\mathbf{d}_6$, $E_H(\mathbf{d}_2)$, $E_H(\mathbf{d}_4)$, $E_H(\mathbf{d}_6)$, $E_H(\mathbf{s}_1)$, $E_H(\mathbf{s}_2)$, and $E_H(\mathbf{s}_3)$, that we similarly denote with an additional apostrophe[3]. For example, if we order $\{\mathbf{x}'_i\}_{i=1}^M$ according to τ, then the bottom $\log(ML)+1$ bits of the $(2L/3)$-th column are $E_H(\mathbf{s}_2)'$, the middle $\log M + 1$ bits are $E_H(\mathbf{d}_4)'$, and the top $M - \log ML - \log M - 2$ bits are $\mathbf{d}'_4$. Now, let

$$\begin{aligned}
\mathbf{s}'_1 &= (\hat{\mathbf{a}}_i)_{i=1}^M \circ (\hat{\mathbf{b}}_{\tau(i)})_{i=1}^M \circ (\hat{\mathbf{c}}_{\rho(i)})_{i=1}^M \\
\mathbf{s}'_2 &= (\hat{\mathbf{a}}_{\tau^{-1}(i)})_{i=1}^M \circ (\hat{\mathbf{b}}_i)_{i=1}^M \circ (\hat{\mathbf{c}}_{\tau^{-1}\rho(i)})_{i=1}^M \\
\mathbf{s}'_3 &= (\hat{\mathbf{a}}_{\rho^{-1}(i)})_{i=1}^M \circ (\hat{\mathbf{b}}_{\rho^{-1}\tau(i)})_{i=1}^M \circ (\hat{\mathbf{c}}_i)_{i=1}^M
\end{aligned}$$

[3]That is, each one of $\mathbf{d}'_2$, $\mathbf{d}'_4$, etc., is obtained from $\mathbf{d}_2$, $\mathbf{d}_4$, etc., by at most a single substitution.

The following lemma shows that at least two of the above $\mathbf{s}'_i$ are close in Hamming distance to their encoded counterpart $(\mathbf{s}_i, E_H(\mathbf{s}_i))$.

Lemma 5. *There exist distinct integers $k, \ell \in [3]$ such that*

$$d_H((\mathbf{s}'_k, E_H(\mathbf{s}_k)'), (\mathbf{s}_k, E_H(\mathbf{s}_k)) \leq 1, \text{ and}$$
$$d_H((\mathbf{s}'_\ell, E_H(\mathbf{s}_\ell)'), (\mathbf{s}_\ell, E_H(\mathbf{s}_k))) \leq 1.$$

Proof. If the substitution did not occur at either of index sets $\{1, \ldots, L/3-1\}$, $\{L/3+1, \ldots, 2L/3-1\}$, or $\{2L/3+1, \ldots, L-1\}$ (which correspond to the values of the $\mathbf{a}_i$'s, $\mathbf{b}_i$'s, and $\mathbf{c}_i$'s, respectively), then the order among the $\mathbf{a}_i$'s, $\mathbf{b}_i$'s and $\mathbf{c}_i$'s is maintained. That is, we have that $\mathbf{s}'_i = \mathbf{s}_i$ for $i \in [3]$, and the claim is clear. It remains to show the other cases, and due to symmetry, assume without loss of generality that the substitution occurred in one of the $\mathbf{a}_i$'s, i.e., in an entry which is indexed by an integer in $\{1, \ldots, L/3-1\}$.

Let $A \in \{0,1\}^{M \times L}$ be a matrix whose rows are the $\mathbf{x}_i$'s, in any order. Let $A_{\textit{left}}$ be the result of ordering the rows of A according to the lexicographic order of their $1, \ldots, L/3-1$ entries. Similarly, let $A_{\textit{mid}}$ and $A_{\textit{right}}$ be the results of ordering the rows of A by their $L/3+1, \ldots, 2L/3-1$ and $2L/3+1, \ldots, L-1$ entries, respectively, and let $A'_{\textit{left}}, A'_{\textit{mid}}$, and $A'_{\textit{right}}$ be defined analogously with $\{\mathbf{x}'_i\}_{i=1}^M$ instead of $\{\mathbf{x}_i\}_{i=1}^M$.

It is readily verified that there exist permutation matrices P_1 and P_2 such that $A_{\textit{mid}} = P_1 A_{\textit{left}}$ and $A_{\textit{right}} = P_2 A_{\textit{left}}$. Moreover, since $\{\mathbf{b}_i\}_{i=1}^M = \{\hat{\mathbf{b}}_i\}_{i=1}^M$, and $\{\mathbf{c}_i\}_{i=1}^M = \{\hat{\mathbf{c}}_i\}_{i=1}^M$, it follows that $A'_{\textit{mid}} = P_1(A_{\textit{left}} + R)$ and $A'_{\textit{right}} = P_2(A_{\textit{left}} + R)$, where $R \in \{0,1\}^{M \times L}$ is a matrix of Hamming weight 1; this clearly implies that $A'_{\textit{mid}} = A_{\textit{mid}} + P_1 R$ and that $A'_{\textit{right}} = A_{\textit{right}} + P_2 R$. Now, notice that $\mathbf{s}_2$ results from vectorizing some submatrix M_2 of $A_{\textit{mid}}$, and $\mathbf{s}'_2$ results from vectorizing some submatrix M'_2 of $A'_{\textit{mid}}$. Moreover, the matrices M_2 and M'_2 are taken from their mother matrix by omitting the same rows and columns, and both vectorizing operations consider the entries of M_2 and M'_2 in the same order. In addition, the redundancies $E_H(\mathbf{s}_2)$ and $E_H(\mathbf{s}_3)$ can be identified similarly, and have at most a single substitution with respect to the corresponding entries in the noiseless codeword. Therefore, it follows from $A'_{\textit{mid}} = A_{\textit{mid}} + P_1 R$ that $d_H((\mathbf{s}'_2, E_H(\mathbf{s}'_2)), (\mathbf{s}_2, E_H(\mathbf{s}_2))) \leq 1$. The claim for $\mathbf{s}_3$ is similar. □

By applying a Hamming decoder on either one of the $\mathbf{s}_i$'s, the decoder obtains possible candidates for $\{\mathbf{a}_i\}_{i=1}^M$, $\{\mathbf{b}_i\}_{i=1}^M$, and $\{\mathbf{c}_i\}_{i=1}^M$, and by Lemma 5, it follows that these sets of candidates will coincide in at least two cases. Therefore, the decoder can apply a majority vote of the candidates from the decoding of each $\mathbf{s}'_i$, and the winning values are $\{\mathbf{a}_i\}_{i=1}^M$, $\{\mathbf{b}_i\}_{i=1}^M$, and $\{\mathbf{c}_i\}_{i=1}^M$. Having these correct values, the decoder can sort $\{\mathbf{x}'_i\}_{i=1}^M$ according to their $\mathbf{a}_i$ columns, and deduce the values of σ and π by observing the resulting permutations in the $\mathbf{b}_i$ and $\mathbf{c}_i$ columns, with respect to their lexicographic ordering. This concludes the decoding of the values d_1, d_3, and d_5 of the data d.

We are left to extract d_2, d_4, and d_6. To this end, observe that since the correct values of $\{\mathbf{a}_i\}_{i=1}^M$, $\{\mathbf{b}_i\}_{i=1}^M$, and $\{\mathbf{c}_i\}_{i=1}^M$ are known at this point, the decoder can extract the *true* positions of $\mathbf{d}_2, \mathbf{d}_4$, and $\mathbf{d}_6$, as well as their respective redundancy bits $E_H(\mathbf{d}_2)$, $E_H(\mathbf{d}_4)$, $E_H(\mathbf{d}_6)$. Hence, the decoding algorithm is complete by applying a Hamming decoder.

We now turn to compute the redundancy of the above code $\mathcal{C}$. Note that there are two sources of redundancy—the Hamming code redundancy, which is at most $3(\log ML + \log M + 2)$ and the fact that the sets $\{\mathbf{a}_i\}_{i=1}^M$, $\{\mathbf{b}_i\}_{i=1}^M$, and $\{\mathbf{c}_i\}_{i=1}^M$ contain distinct strings. By a straightforward computation, for $4 \leq M \leq 2^{L/6}$ we have

$$\begin{aligned} r(\mathcal{C}) &= \log \binom{2^L}{M} \\ &\quad - \log \left(\binom{2^{L/3-1}}{M}^3 \cdot (M!)^2 \cdot 2^{3(M - \log ML - \log M - 2)} \right) \\ &\leq 12 \log e + 3 \log ML + 3 \log M + 6 \end{aligned} \tag{5}$$

where step-by-step computation is given in [6, App. B].

Remark 3. *This construction can be extended to one that corrects K substitutions, by partitioning the $\mathbf{x}_i$'s to $2K+1$ parts, and following a similar outline. The resulting redundancy is $O(K^2 \log(ML))$ [6, Sec. VI].*

VI. Conclusions and Future Work

For $K = 1$, the construction in Section V is a constant away from the lower bound in Corollary 2. For larger values of K, a code construction whose redundancy is asymptotically K times the one in Corollary 1 is given in the full version of this paper [6]. Closing this gap is an interesting open problem. Furthermore, it is intriguing to find a lower bound on the redundancy for larger values of K.

References

[1] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, no. 6102, pp. 1628–1628, 2012.

[2] R. O'Donnell. Analysis of Boolean functions. Cambridge University Press, 2014.

[3] R. P. Feynman, "There's plenty of room at the bottom: An invitation to enter a new field of physics," In *Handbook of Nanoscience, Engineering, and Technology*, Third Edition, pp. 26–35, CRC Press, 2012.

[4] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, no. 7435, pp. 77–80, 2013.

[5] R. Heckel, I. Shomorony, K. Ramchandran, and N. C. David, "Fundamental limits of DNA storage systems," *IEEE International Symposium on Information Theory* (ISIT), pp. 3130–3134, 2017.

[6] J. Sima, N. Raviv, and J. Bruck, "On coding over sliced information," *arXiv:1809.02716* [cs.IT], 2018.

[7] M. Kovačević and V. Y. F Tan, "Codes in the space of multisets–Coding for permutation channels with impairments," *IEEE Transactions on Information Theory*, vol. 64, no. 7, pp. 5156–5169, 2018.

[8] M. Langberg, M. Schwartz, and E. Yaakobi, "Coding for the ℓ_∞-limited permutation channel," *IEEE Transactions on Information Theory*, vol. 63, no. 12, pp. 7676–7686, 2017.

[9] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE International Symposium on Information Theory* (ISIT), pp. 2411–2415, 2018.

[10] M. Luby, "LT codes," *The 43rd Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 2002.

[11] L. Organick *et al.*, "Scaling up DNA data storage and random access retrieval," *bioRxiv*, 2017.

[12] R. Roth, Introduction to coding theory, Cambridge University Press, 2006.

[13] J. M. Walsh and S. Weber, "Capacity region of the permutation channel," *46th Annual Allerton Conference on Communication, Control, and Computing*, pp. 646–652, 2008.

[14] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Scientific reports*, vol. 5, p. 14138, 2015.