

# On Coding over Sliced Information

Jin Sima, Netanel Raviv, *Member, IEEE*, and Jehoshua Bruck, *Fellow, IEEE*

**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 several constructions, some of which are 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 order-wise equivalent to the amount required in the classical error correcting paradigm.

**Index Terms**—Sliced information, DNA storage, Boolean functions.

## 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 [2], [7], [4], [17]. 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* [14], [15], [20], [22], [24], 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, for some

integers  $M$  and  $L$  such that  $M < 2^L$ ; typical values for  $M$  and  $L$  are currently within the order of magnitude of  $10^7$  and  $10^2$ , respectively [17]. Each individual string is subject to various types of errors, such as deletions (i.e., omissions of symbols, which result in a shorter string), insertions (which result in a longer string), and substitutions (i.e., replacements of one symbol by another). In the context of DNA storage, the data are stored following a reading/writing process, the details of which can be found in [17]. In the writing process, 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, the amplified strings are sequenced, which results in multiple (possibly erroneous) reads of each string. These reads, whose length can be shorter or longer than  $L$ , are clustered according to their respective edit-distance<sup>1</sup>. Then, sequence reconstruction algorithms are used within each cluster in order to come up with the most likely origin of the reads in that cluster (Figure 1), and all reconstructed origins are included in the input set of an error correction decoding algorithm.

One of the reasons for error correction in the decoding algorithm is that errors in synthesis might cause the PCR process to amplify a string that was written erroneously, and hence the reconstructed origins might include this erroneous string. In some cases, error correction after synthesis is possible, and yet our model is most suitable for substitution errors that were amplified by the PCR process. In this context, deletions and insertions are easier to handle since they result in a string of length different from<sup>2</sup>  $L$ . The work in [14] provided a scheme that efficiently corrects a single deletion. 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 reconstructed origins will constitute a set of  $M - 1$  strings. In the other, which is undetectable by counting the size of the set of reconstructed origins, 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. Parts of this work have been presented in [23].

Following a formal definition of the channel model in Section II, previous work is discussed in Section III. Upper

This work was supported in part by NSF grants CCF-1816965 and CCF-1717884. The work of Netanel Raviv was supported in part by the postdoctoral fellowship of the Center for the Mathematics of Information (CMI), Caltech, and in part by the Lester-Deutsch postdoctoral fellowship. This paper was presented in part at the IEEE International Symposium on Information Theory, Paris, France, July 2019.

J. Sima and J. Bruck are with the Electrical Engineering Department, California Institute of Technology, Pasadena, CA, 91125 USA (e-mail: jsima@caltech.edu ;bruck@caltech.edu).

N. Raviv was with the Electrical Engineering Department, California Institute of Technology, Pasadena, CA, 91125 USA. He is now with the Department of Computer Science and Engineering, McKelvey School of Engineering, Washington University in Saint Louis, St. Louis, MO, 63130 USA (email: netanel.raviv@gmail.com).

Copyright (c) 2017 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

This work was supported in part by NSF grant CCF- 1816965 and NSF grant CCF-1717884.

<sup>1</sup>The edit distance between two strings is the minimum number of deletions, insertions, and substitutions that turn one to another.

<sup>2</sup>As long as the number of insertions is not equal to the number of deletions.

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. The redundancy of this construction is shown to be optimal up to some constant, which is later improved in Appendix C. In Section VI the construction for a single substitution is generalized to multiple substitutions, and is shown to be order-wise optimal whenever the number of substitutions is a constant. Finally, open problems for future research are discussed in Section VII.

**Remark 1.** *The channel which is discussed in this paper can essentially be seen as taking a string of a certain length  $N$  as input. Then, during transmission, the string is sliced into substrings of equal length, and each substring is subject to substitution errors in the usual sense. Moreover, the order between the slices is lost during transmission, and they arrive as an unordered set.*

*It follows from the sphere-packing bound [19, 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. 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 these  $K$  substitutions. In the case of a single substitution, our codes attain an amount of redundancy that is asymptotically equivalent to the ordinary (i.e., unsliced) channel, whereas for a larger number of substitutions we come close to that, but prove that a comparable amount of redundancy is achievable.*

## II. PRELIMINARIES

To discuss the problem in its most general form and illustrate the ideas, we restrict our attention to binary strings. Most of the techniques in this paper can be extended to non-binary cases. Generally, we denote scalars by lower-case letters  $x, y, \dots$ , vectors by bold symbols  $\mathbf{x}, \mathbf{y}, \dots$ , integers by capital letters  $K, L, \dots$ , and  $[K] \triangleq \{1, 2, \dots, K\}$ . For integers  $M$  and  $L$  such that<sup>3</sup>  $M \leq 2^L$  we denote by  $\binom{\{0,1\}^L}{M}$  the family of all subsets of size  $M$  of  $\{0, 1\}^L$ , and by  $\binom{\{0,1\}^L}{\leq M}$  the family of subsets of size at most  $M$  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  $W = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$  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.

For  $K \leq ML$ , a  $K$ -substitution error ( $K$ -substitution, in short), is an operation that changes the values of at most  $K$  different positions in a word. Notice that the result of a  $K$ -substitution is not necessarily an element of  $\binom{\{0,1\}^L}{M}$ , and might be an element of  $\binom{\{0,1\}^L}{T}$  for some  $M - K \leq T \leq M$ . This gives rise to the following definition.

<sup>3</sup>We occasionally also assume that  $M \leq 2^{cL}$  for some  $0 < c < 1$ . This is in accordance with typical values of  $M$  and  $L$  in contemporary DNA storage prototypes (see Section I).

**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 a  $K$ -substitution in  $W$ .*

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

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

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  $O(K^2 \log(ML))^4$  (Section VI). For  $K = 1$ , the redundancy of this construction is asymptotically at most six times larger than the optimal one (Section V), when  $L$  goes to infinity and  $M \geq 4$ . Furthermore, an improved construction for  $K = 1$  achieves redundancy which is asymptotically at most three times the optimal one (Appendix C), when  $L$  goes to infinity and  $4 \leq M \leq 2^{L/4}$ .*

**Remark 2.** *We note that under the current technology restriction, our code constructions apply to a limited parameter range. For example, when  $K = 1$  and  $L = 100$ , our code requires that  $M \leq 10^5$ . The upper bound constraint on  $M$  is more strict as  $K$  increases. Yet, our constructions provide the following insights.*

*First, as the technology advances, it is reasonable to hope that the error rate will decrease and the synthesis length  $L$  will increase, in which case we have smaller  $K$  and larger  $L$ . As  $K$  gets smaller, the range of  $M$  increases exponentially with  $L$ .*

*Second, while  $L$  is limited under current technology, the number of strings  $M$  can be more freely chosen. One can get smaller  $M$  by storing information in more DNA pools.*

<sup>4</sup>Throughout the paper, we write  $g(n) = O(f(n))$  for any functions  $f, g : \mathbb{N} \rightarrow \mathbb{R}$  and integer  $n$ , if  $\limsup_{n \rightarrow \infty} \frac{g(n)}{f(n)} < \infty$ .

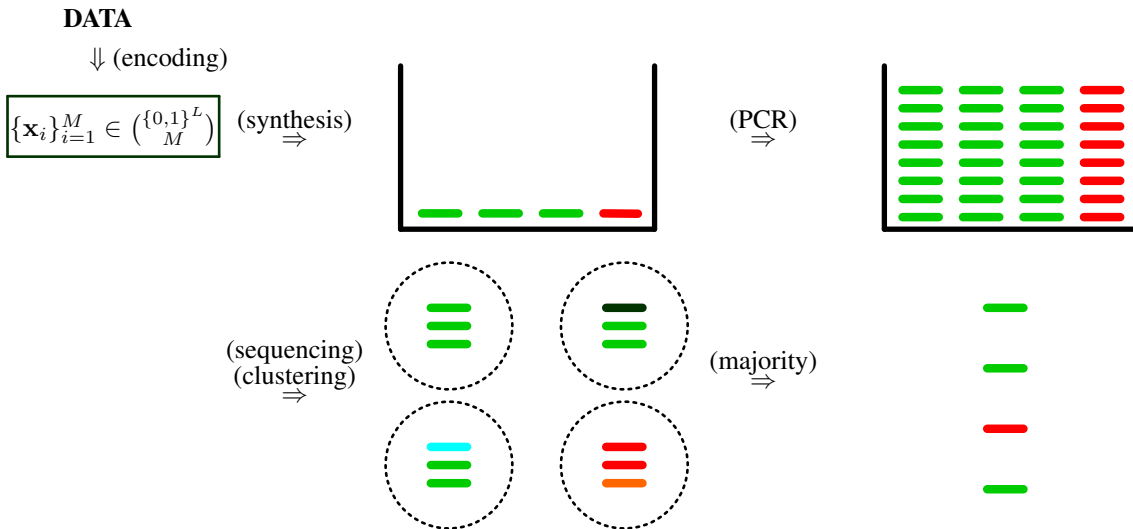


Fig. 1. An illustration of a typical operation of a DNA storage system. The data at hand is encoded to a set of  $M$  binary strings of length  $L$  each. These strings are then synthesized, possibly with errors, into DNA sequences, that are placed in a solution and amplified by a PCR process. Then, the DNA sequences are read, clustered by similarity, and the output set is decided by a majority vote. In the illustrated example, one string is synthesized in error, which causes the output set to be in error. If the erroneous string happens to be equal to another existing string, the output set is of size  $M - 1$ , and otherwise, it is of size  $M$ .

Third, the application of this work is not limited to DNA storage. Our techniques apply to other cases such as network packet transmission.

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  $\mathbf{s}$ . A *linear* vector code is called an  $[n, k]_q$  code if the strings in it form a subspace of dimension  $k$  in  $\mathbb{F}_q^n$ , where  $\mathbb{F}_q$  is the finite field with  $q$  elements.

Several well-known vector-codes are used in the sequel, such as Reed-Solomon codes or Hamming codes. For an integer  $t$ , the Hamming code is an  $[2^t - 1, 2^t - t - 1]_2$  code (i.e., there are  $t$  redundant bits in every codeword), and its minimum Hamming distance is 3. Reed-Solomon (RS) codes over  $\mathbb{F}_q$  exist for every length  $n$  and dimension  $k$ , as long as  $q \geq n - 1$  [19, Sec. 5], and require  $n - k$  redundant symbols in  $\mathbb{F}_q$ . Whenever  $q$  is a power of two, RS codes can be made binary by representing each element of  $\mathbb{F}_q$  as a binary string of length  $\log_2(q)$ . In the sequel we use this form of RS code, which requires  $\log(n)(n - k)$  redundant bits.

Finally, our encoding algorithms make use of *combinatorial numbering maps* [10], that are functions that map a number to an element in some structured set. Specifically,  $F_{com} : \left[ \binom{N}{M} \right] \rightarrow \{S : S \subset [N], |S| = M\}$  maps a number to a set of distinct elements, and  $F_{perm} : [N!] \rightarrow S_N$  maps a number to a permutation in the symmetric group  $S_N$ . The function  $F_{com}$  can be computed using a greedy algorithm

with complexity  $O(MN \log N)$ , and the function  $F_{perm}$  can be computed in a straightforward manner with complexity  $O(N \log N)$ . Using  $F_{com}$  and  $F_{perm}$  together, we define a map  $F : \left[ \binom{N}{M} M! \right] \rightarrow \{S : S \subset [N], |S| = M\} \times S_M$  that maps a number into an unordered set of size  $M$  together with a permutation.

### III. PREVIOUS WORK

The idea of manipulating atomic particles for engineering applications dates back to the 1950's, with R. Feynman's famous citation "there's plenty of room at the bottom" [5]. The specific idea of manipulating DNA molecules for data storage as been circulating the scientific community for a few decades, and yet it was not until 2012-2013 where two prototypes have been implemented [2], [7]. These prototypes have ignited the imagination of practitioners and theoreticians alike, and many works followed suit with various implementations and channel models [1], [6], [8], [9], [18], [26].

By and large, all practical implementations to this day follows the aforementioned channel model, in which multiple short strings are stored inside a solution. Normally, deletions and insertions are also taken into account, but substitutions were found to be the most common form of errors [17, Fig. 3.b], and strings that were subject to insertions and deletions are scarcer, and some of them can be corrected using clustering and sequence reconstruction algorithms.

The channel model in this work has been studied by several authors in the past. The work of [8] 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 already incurs  $\Theta(M \log M)$  redundant bits, or  $\log(e)M - o(1)$  redundancy [14, Remark 1], and will be shown to be strictly sub-optimal in our case. In a different setting, where substitution errors occur probabilistically and the decoding error fades with  $M$  and  $L$ , using indexing prefix was proved asymptotically optimal in terms of coding rate [22]. The indexing prefix technique was also studied in an adversarial setting under substitution errors [15], [20], where codes correcting errors in the indexing prefix were proposed.

The recent work of [14] addressed this model under a bounded number of substitutions, deletions, and insertions per string. When discussing substitutions only, [14] 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 [24] studied a more generalized model, where in addition to substitution errors, insertions/deletions of strings were considered. A distance called sequence-subset distance was defined, and upper bounds and constructions were presented based on the sequence-subset distance. When considering only substitution errors, the upper bounds deal with cases where the number of errors is at least a fraction of  $L$ .

The work of [11] 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 [16] codes or Raptor [21] codes. However, we stress that these solutions rely on randomness at much higher redundancy rates, whereas this work aims for a deterministic and rigorous solution at redundancy which is close to optimal.

Finally, we also mention the *permutation channel* [12], [13], [25], 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, than this channel reduces to *multiset decoding*, as in [11]. 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 of any  $K$ -substitution code. The latter bound demonstrates the asymptotic optimality of the construction in Section V for  $K = 1$ , up to constants. Our techniques rely on upper and lower bounds on the size of the ball  $\mathcal{B}_K$  (Definition 1), which are given below. However, since our measure for distance is not a metric, extra care is needed when applying sphere-packing arguments. We begin with the existential upper bound in Subsection IV-A, continue to provide a lower bound for  $K = 1$  in Subsection IV-B, and extend this bound to larger values of  $K$  in Subsection IV-C.

### A. Existential upper bound

In this subsection, let  $K$ ,  $M$ , and  $L$  be positive integers such that  $K \leq ML$  and  $M \leq 2^L$ . The subsequent series of lemmas will eventually lead to the following upper bound.

**Theorem 2.** *There exists a  $K$ -substitution code  $\mathcal{C} \subseteq \binom{\{0,1\}^L}{M}$  such that  $r(\mathcal{C}) \leq 2K \log(ML) + 2$ .*

We begin with a simple upper bound on the size of the ball  $\mathcal{B}_K$ .

**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  $\{J_i\}_{i=1}^M$ .  $\square$

For  $W \in \binom{\{0,1\}^L}{\leq M}$  let  $\mathcal{R}_K(W)$  be the set of all words  $U \in \binom{\{0,1\}^L}{M}$  such that  $W \in \mathcal{B}_K(U)$ . That is, for a channel output  $W$ , the set  $\mathcal{R}_K(W)$  contains all potential codewords  $U$  whose transmission through the channel can result in  $W$ , given that at most  $K$  substitutions occur. Further, for  $W \in \binom{\{0,1\}^L}{M}$  define the *confusable set* of  $W$  as  $\mathcal{D}_K(W) \triangleq \cup_{W' \in \mathcal{B}_K(W)} \mathcal{R}_K(W')$ . It is readily seen that the words in the confusable set  $\mathcal{D}_K(W)$  of a word  $W$  cannot reside in the same  $K$ -substitution code as  $W$ , and therefore we have the following lemma.

**Lemma 2.** *For every  $K$ ,  $M$ , and  $L$  such that  $K \leq ML$  and  $M \leq 2^L$  there exists a  $K$ -substitution code  $\mathcal{C}$  such that*

$$|\mathcal{C}| \geq \left\lfloor \frac{\binom{2^L}{M}}{D} \right\rfloor, \text{ where}$$

$$D \triangleq \max_{W \in \binom{\{0,1\}^L}{M}} |\mathcal{D}_K(W)|.$$

*Proof.* Initialize a list  $\mathcal{P} = \binom{\{0,1\}^L}{M}$ , and repeat the following process.

- 1) Choose  $W \in \mathcal{P}$ .
- 2) Remove  $\mathcal{D}_K(W)$  from  $\mathcal{P}$ .

Clearly, the resulting code  $\mathcal{C}$  is of the aforementioned size. It remains to show that  $\mathcal{C}$  corrects  $K$  substitutions, i.e.,

that  $\mathcal{B}_K(C) \cap \mathcal{B}_K(C') = \emptyset$  for every distinct  $C, C' \in \mathcal{C}$ . Recall Definition 1 that  $\mathcal{B}_K(C)$  can be obtained by substitution any  $K$  symbols in strings of word  $C$ .

Assume for contradiction that there exist distinct  $C, C' \in \mathcal{C}$  and  $V \in \binom{\{0,1\}^L}{\leq M}$  such that  $V \in \mathcal{B}_K(C) \cap \mathcal{B}_K(C')$ , and w.l.o.g assume that  $C$  was chosen earlier than  $C'$  in the above process. Since  $V \in \mathcal{B}_K(C)$ , it follows that  $\mathcal{R}_K(V) \subseteq \mathcal{D}_K(C)$ . In addition, since  $V \in \mathcal{B}_K(C')$ , it follows that  $C' \in \mathcal{R}_K(V)$ . Therefore, a contradiction is obtained, since  $C'$  is in  $\mathcal{D}_K(C)$ , that was removed from the list  $\mathcal{P}$  when  $C$  was chosen.  $\square$

**Lemma 3.** For an nonnegative integer  $T \leq K$  and  $W \in \binom{\{0,1\}^L}{M-T}$  we have that  $|\mathcal{R}_K(W)| \leq 2(2ML)^K$ .

*Proof.* Denote  $W = \{\mathbf{y}_1, \dots, \mathbf{y}_{M-T}\}$  and let  $U \in \mathcal{R}_K(W)$ . Notice that by the definition of  $\mathcal{R}_K(W)$ , there exists a  $K$ -substitution operation which turns  $U$  to  $W$ . Therefore, every  $\mathbf{y}_i$  in  $W$  is a result of a certain nonnegative number of substitutions in one or more strings in  $U$ . Hence, we denote by  $\mathbf{z}_1^1, \dots, \mathbf{z}_{i_1}^1$  the strings in  $U$  that resulted in  $\mathbf{y}_1$  after the  $K$ -substitution operation, we denote by  $\mathbf{z}_1^2, \dots, \mathbf{z}_{i_2}^2$  the strings which resulted in  $\mathbf{y}_2$ , and so on, up to  $\mathbf{z}_1^{M-T}, \dots, \mathbf{z}_{i_{M-T}}^{M-T}$ , which resulted in  $\mathbf{y}_{M-T}$ . Therefore, since  $U = \cup_{j=1}^{M-T} \{\mathbf{z}_1^j, \dots, \mathbf{z}_{i_j}^j\}$ , it follows that there exists a set  $\mathcal{L} \subseteq [M] \times [L]$ , of size at most  $K$ , such that

$$\begin{pmatrix} \mathbf{z}_1^1 \\ \vdots \\ \mathbf{z}_{i_1}^1 \\ \mathbf{z}_1^2 \\ \vdots \\ \mathbf{z}_{i_{M-T-1}}^{M-T-1} \\ \mathbf{z}_1^{M-T} \\ \vdots \\ \mathbf{z}_{i_{M-T}}^{M-T} \end{pmatrix} \stackrel{(\mathcal{L})}{=} \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{M-T-1} \\ \mathbf{y}_{M-T} \\ \vdots \\ \mathbf{y}_{M-T} \end{pmatrix}, \quad (1)$$

where  $(\cdot)^{(\mathcal{L})}$  is a matrix operator, which corresponds to flipping the bits that are indexed by  $\mathcal{L}$  in the matrix on which it operates. In what follows, we bound the number of ways to choose  $\mathcal{L}$ , which will consequently provide a bound on  $|\mathcal{R}_K(W)|$ . Note that the number of ways to choose  $\mathcal{L}$  is summed over all possible tuples  $(i_1, \dots, i_{M-T})$ .

First, define  $\mathcal{P} = \{p : i_p > 1\}$ , and denote  $P \triangleq |\mathcal{P}|$ . Therefore, since  $\sum_{j=1}^{M-T} i_j = M$ , it follows that

$$\begin{aligned} \sum_{p \in \mathcal{P}} i_p &= \sum_{j=1}^{M-T} i_j - \sum_{j \notin \mathcal{P}} i_j \\ &= M - (M - T - P) \\ &= T + P. \end{aligned} \quad (2)$$

Second, notice that for every  $p \in \mathcal{P}$ , the set  $\{\mathbf{z}_1^p, \dots, \mathbf{z}_{i_p}^p\}$  contains  $i_p$  different strings. Hence, since after the  $K$ -substitution operation they are all equal to  $\mathbf{y}_p$ , it follows that at least  $i_p - 1$  of them must undergo at least one substitution. Clearly, there are  $\binom{i_p}{i_p - 1} = i_p$  different ways to choose who will these  $i_p - 1$  strings be, and additional  $L^{i_p - 1}$  different ways to determine

the locations of the substitutions, and therefore  $i_p \cdot L^{i_p - 1}$  ways to choose these  $i_p - 1$  substitutions.

Third, notice that

$$\begin{aligned} K - \sum_{p \in \mathcal{P}} (i_p - 1) &= K - \sum_{p \in \mathcal{P}} i_p + P \\ &\stackrel{(2)}{=} K - T, \end{aligned} \quad (3)$$

and hence, there are at most  $K - T$  remaining positions to be chosen to  $\mathcal{L}$ , after choosing the  $i_p - 1$  positions for every  $p \in \mathcal{P}$  as described above.

Now, let  $\mathcal{I}$  be the set of all tuples  $(i_1, \dots, i_{M-T})$  of positive integers that sum to  $M$  (whose size is  $\binom{M-1}{M-T-1}$  by the famous stars and bars theorem). Let  $N : \mathcal{I} \rightarrow \mathbb{N}$  be a function which maps  $(i_1, \dots, i_{M-T}) \in \mathcal{I}$  to the number of different  $U \in \mathcal{R}_K(W)$  for which there exist  $\mathcal{L} \subseteq [M] \times [L]$  of size at most  $K$  such that (1) is satisfied. Since this quantity is at most the number of ways to choose a suitable  $\mathcal{L}$ , the above arguments demonstrate that

$$N(i_1, \dots, i_{M-T}) \leq \binom{ML}{K-T} \prod_{p \in \mathcal{P}} i_p L^{i_p - 1}.$$

Then, we have

$$\begin{aligned} |\mathcal{R}_K(W)| &\leq \sum_{\mathcal{I}} N(i_1, \dots, i_{M-T}) \\ &\leq \sum_{\mathcal{I}} \binom{ML}{K-T} \prod_{p \in \mathcal{P}} i_p L^{i_p - 1} \\ &\leq \sum_{\mathcal{I}} (ML)^{K-T} L^{\sum_p (i_p - 1)} \prod_{p \in \mathcal{P}} i_p \\ &\stackrel{(3)}{\leq} \sum_{\mathcal{I}} (ML)^{K-T} L^T \prod_{p \in \mathcal{P}} i_p. \end{aligned} \quad (4)$$

Since the geometric mean of positive numbers is always less than the arithmetic one, we have  $(\prod_{p \in \mathcal{P}} i_p)^{1/P} \leq \frac{1}{P} \sum_{p \in \mathcal{P}} i_p$ , and hence,

$$\begin{aligned} (4) &\leq \sum_{\mathcal{I}} (ML)^{K-T} L^T \left( \frac{\sum_p i_p}{P} \right)^P \\ &\stackrel{(2)}{\leq} \sum_{\mathcal{I}} (ML)^{K-T} L^T ((T+P)/P)^P \\ &\leq \binom{M-1}{M-T-1} (ML)^{K-T} L^T ((T+P)/P)^P \\ &\leq \binom{M}{T} (ML)^{K-T} L^T ((T+P)/P)^P \\ &\leq (ML)^{K-T} (ML)^T ((T+P)/P)^P \\ &\leq (ML)^K ((T+P)/P)^P \\ &\stackrel{(a)}{\leq} (ML)^K 2^T \\ &\leq (2ML)^K, \end{aligned} \quad (5)$$

where (a) will be proved in Appendix D.  $\square$

*Proof.* (of Theorem 2) It follows from Lemma 1, Lemma 3, and from the definition of  $D$  that

$$D \leq \max_{W \in \binom{\{0,1\}^L}{M}} |\mathcal{B}_K(W)| \cdot \max_{W \in \binom{\{0,1\}^L}{M}} |\mathcal{R}_K(W)|$$

$$\leq \left( \sum_{\ell=0}^K \binom{ML}{\ell} \right) (2ML)^K.$$

Therefore, the code  $\mathcal{C}$  that is constructed in Lemma 2 satisfies

$$\begin{aligned} r(\mathcal{C}) &\leq \log \binom{2^L}{M} - \log |\mathcal{C}| \\ &\leq \log \left( \left( \sum_{\ell=0}^K \binom{ML}{\ell} \right) (2ML)^K \right) \\ &= \log \left( \sum_{\ell=0}^K \binom{ML}{\ell} \right) + K(\log(ML) + 1) \\ &\leq \log \left( K \binom{ML}{K} \right) + K(\log(ML) + 1) \\ &\leq (\log(K) - \log(K!) + \log(ML^K)) \\ &\quad + K(\log(ML) + 1) \\ &\leq 2K \log(ML) + 2. \end{aligned} \quad \square$$

### B. Lower bound for a single substitution code

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  $M \leq 2^L / (\sum_{i=0}^K \binom{L}{i})$  [19, 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. This is illustrated in the following example.

**Example 2.** For  $L = 3$  and  $M = 2$ , consider the word  $W = \{010, \underline{011}\}$ . By flipping either the two underlined symbols, or the two bold symbols, the word  $W' = \{010, 110\}$  is obtained. Hence, different substitution operation might result in identical words. As a result, the size of the ball

$$\begin{aligned} \mathcal{B}_2(W) &= \{\{010, 011\}, \{110, 011\}, \{000, 011\}, \{011\}, \\ &\quad \{010, 111\}, \{010, 001\}, \{010\}, \{100, 011\}, \\ &\quad \{111, 011\}, \{110, 111\}, \{110, 001\}, \{110, 010\}, \\ &\quad \{001, 011\}, \{000, 111\}, \{000, 001\}, \{000, 010\}, \\ &\quad \{010, 101\}\} \end{aligned}$$

is 17, which is smaller than the upper bound of the  $\mathcal{B}_2(W)$  ball size  $\binom{ML}{0} + \binom{ML}{1} + \binom{ML}{2} = 22$ .

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  $K = 1$ . A word  $W \in \binom{\{0,1\}^L}{M}$  corresponds to a Boolean function  $f_W : \{\pm 1\}^L \rightarrow \{\pm 1\}$  as follows. For  $\mathbf{x} \in \{0, 1\}^L$  let  $\bar{\mathbf{x}} \in \{\pm 1\}^L$  be the vector which is obtained from  $\mathbf{x}$  by replacing every 0 by 1 and every 1 by  $-1$ . Then, we define  $f_W(\bar{\mathbf{x}}) = -1$  if  $\mathbf{x} \in W$ , and 1 otherwise. Considering the set  $\{\pm 1\}^L$  as the hypercube graph<sup>5</sup>, the boundary  $\partial f_W$  of  $f_W$

<sup>5</sup>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.

is the set of all edges  $\{\mathbf{x}_1, \mathbf{x}_2\} \in \binom{\{\pm 1\}^L}{2}$  in this graph such that  $f_W(\mathbf{x}_1) \neq f_W(\mathbf{x}_2)$ .

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

*Proof.* Let  $W = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$  be a word. Every edge  $e = \{\mathbf{x}_i, \mathbf{x}'\}$  on the boundary of  $f_W$  corresponds to a substitution operation in  $\mathbf{x}_i$  from  $W$  that results in a word  $W_e = \{\mathbf{x}_1, \dots, \mathbf{x}_{i-1}, \mathbf{x}', \mathbf{x}_{i+1}, \dots, \mathbf{x}_M\} \in (\mathcal{B}_1(W) \setminus \{W\}) \cap \binom{\{0,1\}^L}{M}$ . To show that every edge  $e$  on the boundary corresponds to a *unique* word  $W_e$  in  $\mathcal{B}_1(W)$ , assume for contradiction that  $W_e = W_{e'}$  for two distinct edges  $e = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2\}$  and  $e' = \{\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2\}$ , where  $\mathbf{x}_1, \mathbf{y}_1 \in W$  and  $\mathbf{x}_2, \mathbf{y}_2 \notin W$ . Since both  $W_e$  and  $W_{e'}$  contain precisely one element which is not in  $W$ , and are missing one element which is in  $W$ , it follows that  $\mathbf{x}_1 = \mathbf{y}_1$  and  $\mathbf{x}_2 = \mathbf{y}_2$ , a contradiction. Therefore, there exists an injective mapping between the boundary of  $f_W$  and  $\mathcal{B}_1(W) \setminus \{W\}$ , and the claim follows.  $\square$

Notice that the bound in Lemma 4 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. Having established the connection between  $\mathcal{B}_1(W)$  and the boundary of  $f_W$ , the following Fourier analytic claim 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  $\mathbf{x}$  is chosen uniformly at random.

**Lemma 5.** [3, Theorem 2.39] For every function  $f : \{\pm 1\}^L \rightarrow \mathbb{R}$ , 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 6.** Let  $L$  and  $M$  be integers such that  $M \leq 2^{(1-\epsilon)L}$  and  $L > \frac{1}{\epsilon}$  for some  $0 < \epsilon < 1$ . For every word  $W \in \binom{\{0,1\}^L}{M}$  we have that  $|\partial f_W| \geq \epsilon ML$ .

*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  $\alpha = M/2^L$  whenever  $L > \frac{1}{\epsilon}$ , which holds for every non-constant  $L$ . In addition, since  $\Pr_{\mathbf{x}}(f_W(\bar{\mathbf{x}}) \neq f_W(\bar{\mathbf{x}}^{\oplus i}))$  equals the fraction of dimension  $i$  edges that lie on the boundary of  $f_W$  ([3, Fact 2.14]), Lemma 4 implies that

$$I(f_W) = \frac{|\partial f_W|}{2^{L-1}}.$$

Therefore, since  $M \leq 2^{(1-\epsilon)L}$  and from Lemma 5 we have that  $|\partial f_W| = 2^{L-1} I(f_W) \geq 2^L \alpha \log(1/\alpha) = M \log(2^L/M) \geq \epsilon ML$ .  $\square$

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

*Proof.* According to Lemma 4 and Lemma 6, every codeword of every  $\mathcal{C}$  excludes at least  $\epsilon ML$  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

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

**Remark 3.** A similar lower bound was presented in [14], where it was shown that for a code correcting  $s$  loss of strings,  $q$  substitution errors in each of  $t$  strings, the redundancy is lower bounded by

$$sL + t \log M + tq \log L - \log(s!t!q!^t) + o(1),$$

when  $M = 2^{\beta L}$  for some  $0 < \beta < 1$ . Taking  $s = 0$  and  $q = t = 1$  results in the lower bound  $\log(ML) + o(1)$ , which is order-wise the same as, and stronger by a constant  $\log 1/\epsilon$  than the lower bound  $\log(ML) - \log 1/\epsilon$  in Corollary 1. Compared to the bound in [14], the bound in Corollary 1 does not require  $M$  to be exponential in  $L$ , and thus applies to broader parameter range for  $M$  and  $L$ .

### C. Lower bound for more than one substitution

Similar techniques to the ones in Subsection IV-B can be used to obtain a lower bound for larger values of  $K$ . Specifically, we have the following theorem.

**Theorem 3.** For integers  $L, M, K$ , and positive constants  $\epsilon, c < 1$  such that  $M \leq 2^{(1-\epsilon)L}$ ,  $L > \frac{1}{\epsilon}$ , and  $K \leq c\epsilon\sqrt{M}$ , a  $K$ -substitution code  $\mathcal{C} \subseteq \binom{\{0,1\}^L}{M}$  satisfies that  $r(\mathcal{C}) \geq K(\log(ML) - 2\log(K)) - O(1)$ .

To prove this theorem, it is shown that certain special  $K$ -subsets of  $\partial f_W$  correspond to words in  $\mathcal{B}_K(W)$ , and by bounding the number of these special subsets from below, the lower bound is attained. A subset of  $K$  boundary edges is called *special*, if it does not contain two edges that intersect on a node (i.e., a string) in  $W$ . Formally, a subset  $\mathcal{S} \subseteq \partial f_W$  is special if  $|\mathcal{S}| = K$ , and for every  $\{\mathbf{x}_1, \mathbf{y}_1\}, \{\mathbf{x}_2, \mathbf{y}_2\} \in \mathcal{S}$  with  $f_W(\bar{\mathbf{x}}_1) = f_W(\bar{\mathbf{x}}_2) = -1$  and  $f_W(\bar{\mathbf{y}}_1) = f_W(\bar{\mathbf{y}}_2) = 1$  we have that  $\mathbf{x}_1 \neq \mathbf{x}_2$ . We begin by showing how special sets are relevant to proving Theorem 3.

**Lemma 7.** For every word  $W \in \binom{\{0,1\}^L}{M}$  we have that  $|\mathcal{B}_K(W)| \geq \frac{|\{\mathcal{S} \subseteq \partial f_W \mid \mathcal{S} \text{ is special}\}|}{K^K}$ .

*Proof.* It is shown that every special set corresponds to a word in  $\mathcal{B}_K(W)$ , and at most  $K^K$  different special sets can correspond to the same word (namely, there exists a mapping from the family of special sets to  $\mathcal{B}_K(W)$ , which is at most  $K^K$  to 1). Let  $\mathcal{S} = \{\{\mathbf{x}_i, \mathbf{y}_i\}\}_{i=1}^K$  be special, where  $f_W(\bar{\mathbf{x}}_i) = -1$  and  $f_W(\bar{\mathbf{y}}_i) = 1$  for every  $i \in [K]$ . Let  $W_{\mathcal{S}} \in \binom{\{0,1\}^L}{\leq M}$  be obtained from  $W$  by removing the  $\mathbf{x}_i$ 's and adding the  $\mathbf{y}_i$ 's, i.e.,  $W_{\mathcal{S}} \triangleq (W \setminus \{\mathbf{x}_i\}_{i=1}^K) \cup \{\mathbf{y}_i\}_{i=1}^K$  for some  $T \leq K$ ; notice that there are exactly  $K$  distinct  $\mathbf{x}_i$ 's but at most  $K$  distinct  $\mathbf{y}_i$ 's, since  $\mathcal{S}$  is special, and therefore we assume w.l.o.g that  $\mathbf{y}_1, \dots, \mathbf{y}_T$  are the distinct  $\mathbf{y}_i$ 's. It is readily verified that  $W_{\mathcal{S}} \in \mathcal{B}_K(W)$ , since  $W_{\mathcal{S}}$  can be obtained from  $W$  by performing  $K$  substitution operations in  $W$ , each

of which corresponds to an edge in  $\mathcal{S}$ . Moreover, every  $\mathcal{S}$  corresponds to a unique surjective function  $f_{\mathcal{S}} : [K] \rightarrow [T]$  such that  $f_{\mathcal{S}}(i) = j$  if there exists  $j \leq T$  such that  $\{\mathbf{x}_i, \mathbf{y}_j\} \in \mathcal{S}$ , and hence at most  $K^T \leq K^K$  different special sets  $\mathcal{S}$  can correspond to the same word in  $\mathcal{B}_K(W)$ .  $\square$

We now turn to prove a lower bound on the number of special sets.

**Lemma 8.** Let  $L$  and  $M$  be integers such that  $M \leq 2^{(1-\epsilon)L}$  and  $L > \frac{1}{\epsilon}$  for some  $0 < \epsilon < 1$ . If there exists a positive constant  $c < 1$  such that  $K \leq c \cdot \epsilon\sqrt{M}$ , then there are at least  $(1 - c^2) \binom{|\partial f_W|}{K}$  special sets  $\mathcal{S} \subseteq \partial f_W$ .

*Proof.* Clearly, the number of ways to choose a  $K$ -subset of  $\partial f_W$  which is *not* special, i.e., contains  $K$  distinct edges of  $\partial f_W$  but at least two of those are adjacent to the same  $\mathbf{x} \in W$ , is at most

$$\begin{aligned} &M \cdot \binom{L}{2} \cdot \binom{|\partial f_W|}{K-2} \\ &= M \cdot \binom{L}{2} \cdot \frac{K(K-1)}{(|\partial f_W| - K + 2)(|\partial f_W| - K + 1)} \cdot \binom{|\partial f_W|}{K}. \end{aligned}$$

Observe that the multiplier of  $\binom{|\partial f_W|}{K}$  in the above expression can be bounded as follows.

$$\begin{aligned} &M \cdot \binom{L}{2} \cdot \frac{K(K-1)}{(|\partial f_W| - K + 2)(|\partial f_W| - K + 1)} \\ &\leq M \cdot \binom{L}{2} \cdot \frac{K(K-1)}{(\epsilon ML - K + 2)(\epsilon ML - K + 1)} \\ &\leq M \cdot L^2 \cdot \frac{K^2}{\epsilon^2 M^2 L^2}, \end{aligned}$$

where the former inequality follows since  $|\partial f_W| \geq \epsilon ML$  by Lemma 6; the latter inequality follows since  $K \leq c\epsilon\sqrt{M}$  implies that  $\epsilon ML - K + 2 \geq \epsilon ML - K + 1 \geq \frac{1}{\sqrt{2}} \cdot \epsilon ML$  whenever  $\frac{c\sqrt{2}}{\sqrt{2}-1} \leq L\sqrt{M}$ , which holds for every non-constant  $M$  and  $L$ . Therefore, since

$$M \cdot L^2 \cdot \frac{K^2}{\epsilon^2 M^2 L^2} = \frac{K^2}{\epsilon^2 M} \leq c^2,$$

it follows that these are at least  $(1 - c^2) \cdot \binom{|\partial f_W|}{K}$  special subsets in  $\partial f_W$ .  $\square$

Lemma 7 and Lemma 8 readily imply that  $|\mathcal{B}_K(W)| \geq \frac{(1-c^2)}{K^K} \binom{\epsilon ML}{K}$  for every  $W \in \binom{\{0,1\}^L}{M}$ , from which we can prove Theorem 3.

*Proof.* (of Theorem 3) Clearly, no two  $K$ -balls around code-words in  $\mathcal{C}$  can intersect, and therefore we must have  $|\mathcal{C}| \leq \binom{2^L}{M} / \min_{W \in \mathcal{C}} |\mathcal{B}_K(W)|$ . Therefore,

$$\begin{aligned} r(\mathcal{C}) &= \log \binom{2^L}{M} - \log |\mathcal{C}| \\ &\geq \log \left( \frac{(1-c^2)}{K^K} \binom{\epsilon ML}{K} \right) \\ &= \log \binom{\epsilon ML}{K} - K \log(K) - O(1) \\ &\geq \log \left( \frac{(\epsilon ML - K + 1)^K}{K^K} \right) - K \log(K) - O(1) \end{aligned}$$

$$\begin{aligned}
&\geq \log \left( \frac{\left( \frac{1}{\sqrt{2}} \epsilon ML \right)^K}{K^K} \right) - K \log(K) - O(1) \\
&= K \left( \log\left(\frac{\epsilon}{\sqrt{2}}\right) + \log(ML) \right) - 2K \log(K) - O(1) \\
&\geq K \log(ML) - 2K \log(K) - O(K) \quad \square
\end{aligned}$$

## 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  $3 \log ML + 3 \log M + O(1)$ . For simplicity of illustration, we restrict our attention to values of  $M$  and  $L$  such that  $\lceil \log ML \rceil + \lceil \log M \rceil \leq M$ . In the remaining values, a similar construction of comparable redundancy exists.

**Theorem 4.** For  $D = \left[ \binom{2^{L/3-1}}{M} \right]^3 \cdot (M!)^2 \cdot 2^{3M-3 \log ML-3 \log M-6}$ , there exist an encoding function  $E : D \rightarrow \left( \{0,1\}^L \right)$  whose image is a single substitution correcting code.

The idea behind Theorem 4 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  $d = (d_1, \dots, d_6)$ , where

$$\begin{aligned}
d_1 &\in \left[ \binom{2^{L/3-1}}{M} \right], \\
d_3, d_5 &\in \left[ \binom{2^{L/3-1}}{M} M! \right],
\end{aligned}$$

and

$$d_2, d_4, d_6 \in [2^{M-\log ML-\log M-2}].$$

Apply the functions  $F_{com}$ ,  $F_{perm}$ , and  $F$  (see Section II) to obtain

$$\begin{aligned}
F_{com}(d_1) &= \{\mathbf{a}_1, \dots, \mathbf{a}_M\}, \\
F(d_3) &= (\{\mathbf{b}_1, \dots, \mathbf{b}_M\}, \sigma), \\
F(d_5) &= (\{\mathbf{c}_1, \dots, \mathbf{c}_M\}, \pi),
\end{aligned} \quad (6)$$

where  $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i \in \{0,1\}^{L/3-1}$  for every  $i \in [M]$ , the permutations  $\sigma$  and  $\pi$  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. Further, 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 let

$$\begin{aligned}
\mathbf{s}_1 &= (\mathbf{a}_1, \dots, \mathbf{a}_M, \mathbf{b}_{\sigma(1)}, \dots, \mathbf{b}_{\sigma(M)}, \\
&\quad \mathbf{c}_{\pi(1)}, \dots, \mathbf{c}_{\pi(M)}), \\
\mathbf{s}_2 &= (\mathbf{a}_{\sigma^{-1}(1)}, \dots, \mathbf{a}_{\sigma^{-1}(M)}, \mathbf{b}_1, \dots, \mathbf{b}_M,
\end{aligned}$$

$$\begin{aligned}
&\quad \mathbf{c}_{\sigma^{-1}\pi(1)}, \dots, \mathbf{c}_{\sigma^{-1}\pi(M)}), \text{ and} \\
\mathbf{s}_3 &= (\mathbf{a}_{\pi^{-1}(1)}, \dots, \mathbf{a}_{\pi^{-1}(M)}, \mathbf{b}_{\pi^{-1}\sigma(1)}, \dots, \mathbf{b}_{\pi^{-1}\sigma(M)}, \\
&\quad \mathbf{c}_1, \dots, \mathbf{c}_M).
\end{aligned} \quad (7)$$

Without loss of generality<sup>6</sup> assume that there exists an integer  $t$  for which the length  $|\mathbf{s}_i| = (L-3)M = 2^t - t - 1$  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  $t \leq \lceil \log(ML) \rceil$ . Similarly, we assume that there exists an integer  $h$  for which the length  $|\mathbf{d}_i| = 2^h - h - 1$  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  $[2^h - 1, 2^h - h - 1]$  Hamming code. By the properties of a Hamming code, and by the definition of  $h$ , we have that  $h \leq \lceil \log(M) \rceil$ .

The data  $d \in D$  is mapped to a codeword  $C = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$  as follows, and the reader is encouraged to refer to Figure 2 for clarifications. 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  $\sigma$  and  $\pi$ . That is, denoting  $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,L})$ , we define

$$\begin{aligned}
(x_{i,1}, \dots, x_{i,L/3-1}) &= \mathbf{a}_i, \\
(x_{i,L/3+1}, \dots, x_{i,2L/3-1}) &= \mathbf{b}_{\sigma(i)}, \text{ and} \\
(x_{i,2L/3+1}, \dots, x_{i,L-1}) &= \mathbf{c}_{\pi(i)}.
\end{aligned} \quad (8)$$

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.

$$x_{i,L/3} = \begin{cases} d_{2,i}, & \text{if } i \in [M - \lceil \log ML \rceil - \lceil \log M \rceil], \\ E_H(\mathbf{d}_2)_{i - (M - \lceil \log ML \rceil - \lceil \log M \rceil)}, & \text{if } i \in [M - \lceil \log ML \rceil - \lceil \log M \rceil + 1, \\ & M - \lceil \log ML \rceil], h \\ E_H(\mathbf{s}_1)_{i - (M - \lceil \log ML \rceil)}, & \text{if } i \in [M - \lceil \log ML \rceil + 1, M], \end{cases}$$

$$x_{i,2L/3} = \begin{cases} d_{4,\sigma^{-1}(i)}, & \text{if } \sigma^{-1}(i) \in [M - \lceil \log ML \rceil - \lceil \log M \rceil], \\ E_H(\mathbf{d}_4)_{\sigma^{-1}(i) - (M - \lceil \log ML \rceil - \lceil \log M \rceil)}, & \text{if } \sigma^{-1}(i) \in [M - \lceil \log ML \rceil - \lceil \log M \rceil \\ & + 1, M - \lceil \log ML \rceil], \\ E_H(\mathbf{s}_2)_{\sigma^{-1}(i) - (M - \lceil \log ML \rceil)}, & \text{if } \sigma^{-1}(i) \in [M - \lceil \log ML \rceil + 1, M], \end{cases}$$

<sup>6</sup>Every string can be padded with zeros to extend its length to  $2^t - t - 1$  for some  $t$ . It is readily verified that this operation extends the string by at most a factor of two, and by the properties of the Hamming code, this will increase the number of redundant bits by at most 1.



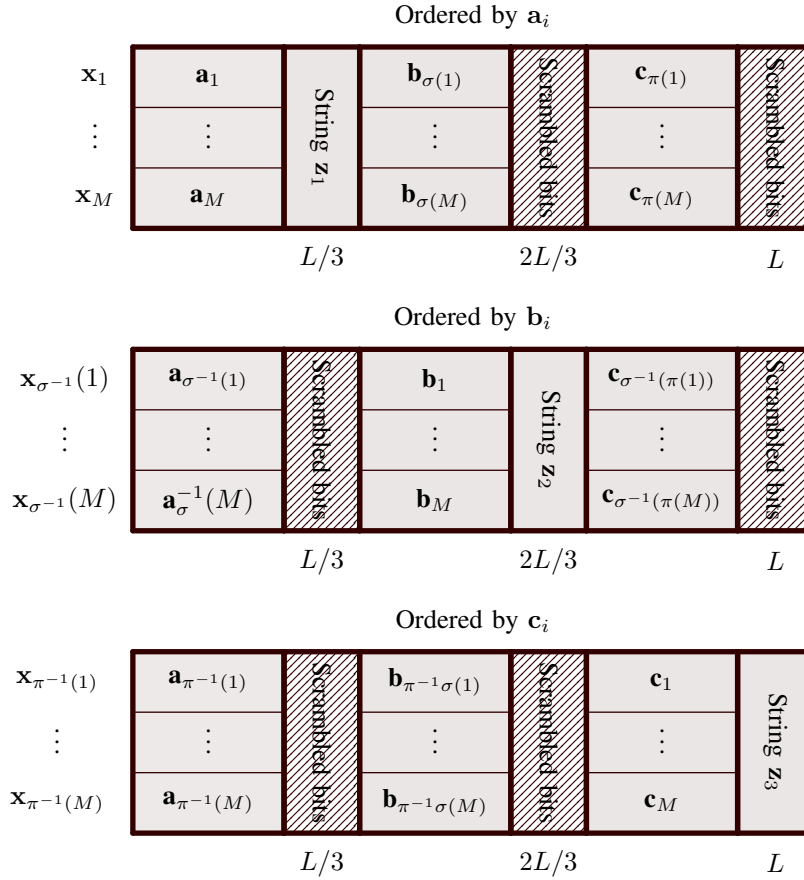


Fig. 2. This figure illustrates the three different  $M \times L$  binary matrices which results from placing the strings  $\{\mathbf{x}_i\}_{i=1}^M$  on top of one another in various orders. That is, each of the three matrices represents the same codeword set  $\{x_1, x_2, \dots, x_M\}$ . Notice that the strings  $\mathbf{z}_1$ ,  $\mathbf{z}_2$ , and  $\mathbf{z}_3$  constitute three  $M \times 1$  columns, that contain the bits of  $(\mathbf{d}_2, E_H(\mathbf{d}_2), E_H(\mathbf{s}_1))$ ,  $(\mathbf{d}_4, E_H(\mathbf{d}_4), E_H(\mathbf{s}_2))$ , and  $(\mathbf{d}_6, E_H(\mathbf{d}_6), E_H(\mathbf{s}_3))$  respectively. For example, when sorting the  $\mathbf{x}_i$ 's according to the  $\mathbf{a}_i$ 's (top figure), the bits of  $\mathbf{d}_2$ ,  $E_H(\mathbf{d}_2)$ , and  $E_H(\mathbf{s}_1)$  appear consecutively.

$$x_{i,L} = \begin{cases} d_{6,\pi^{-1}(i)}, & \text{if } \pi^{-1}(i) \in [M - \lceil \log ML \rceil - \lceil \log M \rceil], \\ E_H(\mathbf{d}_6)^{\pi^{-1}(i) - (M - \lceil \log ML \rceil - \lceil \log M \rceil)}, & \text{if } \pi^{-1}(i) \in [M - \lceil \log ML \rceil - \lceil \log M \rceil \\ + 1, M - \lceil \log ML \rceil], \\ E_H(\mathbf{s}_3)^{\pi^{-1}(i) - (M - \lceil \log ML \rceil)}, & \text{if } \pi^{-1}(i) \in [M - \lceil \log ML \rceil + 1, M], \end{cases} \quad (9)$$

That is, if the strings  $\{\mathbf{x}_i\}_{i=1}^M$  are sorted according to the content of the bits  $(x_{i,1}, \dots, x_{i,L/3-1}) = \mathbf{a}_i$ , then the top  $M - \lceil \log ML \rceil - \lceil \log M \rceil$  bits of the  $(L/3)$ 'th column<sup>7</sup> contain  $\mathbf{d}_2$ , the middle  $\lceil \log M \rceil$  bits contain  $E_H(\mathbf{d}_2)$ , and the bottom  $\lceil \log ML \rceil$  bits contain  $E_H(\mathbf{s}_1)$ . Similarly, if the strings are sorted according to  $(x_{i,L/3+1}, \dots, x_{i,2L/3-1}) = \mathbf{b}_i$ , then the top  $M - \lceil \log ML \rceil - \lceil \log M \rceil$  bits of the  $(2L/3)$ 'th column contain  $\mathbf{d}_4$ , the middle  $\lceil \log M \rceil$  bits contain  $E_H(\mathbf{d}_4)$ , and the bottom  $\lceil \log ML \rceil$  bits contain  $E_H(\mathbf{s}_2)$ , and so on. Equations (8) and (9) conclude the encoding function  $E$  of Theorem 4. It can be readily

<sup>7</sup>Sorting the strings  $\{\mathbf{x}_i\}_{i=1}^M$  by any ordering method provides a matrix in a natural way, and can consider columns in this matrix.

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 (8), 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  $d$ . The algorithm is summarized in Algorithm 1.

Upon receiving a word  $C' = \{\mathbf{x}'_1, \dots, \mathbf{x}'_M\} \in \mathcal{B}_1(C)$  for some codeword  $C$  (once again, the indexing of the elements of  $C'$  is lexicographic), we define

$$\begin{aligned} \hat{\mathbf{a}}_i &= (x'_{i,1}, \dots, x'_{i,L/3-1}) \\ \hat{\mathbf{b}}_i &= (x'_{\tau^{-1}(i),L/3+1}, \dots, x'_{\tau^{-1}(i),2L/3-1}) \\ \hat{\mathbf{c}}_i &= (x'_{\rho^{-1}(i),2L/3+1}, \dots, x'_{\rho^{-1}(i),L-1}), \end{aligned} \quad (10)$$

**Algorithm 1: Decoding**


---

**Input:** A word  $C' \in \mathcal{B}_1(C)$  for some codeword  $C$ .  
**Output:** The message  $d$  encoded as  $C$ .  
Sort and index the strings in  $C' = \{\mathbf{x}'_1, \dots, \mathbf{x}'_M\}$  lexicographically;  
Compute the strings  $\hat{\mathbf{a}}_i$ ,  $\hat{\mathbf{b}}_i$ , and  $\hat{\mathbf{c}}_i$  for  $i \in [M]$ , according to (10);  
Compute the strings  $\mathbf{s}'_1$ ,  $\mathbf{s}'_2$ , and  $\mathbf{s}'_3$  according to (12);  
Compute the strings  $E_H(\mathbf{s}_1)'$ ,  $E_H(\mathbf{s}_2)'$ , and  $E_H(\mathbf{s}_3)'$  according to (11);  
Use Hamming decoder to decode  $(\mathbf{s}'_i, E_H(\mathbf{s}_i))$  and obtain  $\mathbf{s}_i$  for  $i \in [3]$ ;  
According to Lemma 9, we can apply majority vote on the recovered  $\{\mathbf{s}_i\}_{i=1}^3$  to obtain the correct strings  $\{\mathbf{a}_i\}_{i=1}^M$ ,  $\{\mathbf{b}_i\}_{i=1}^M$ , and  $\{\mathbf{c}_i\}_{i=1}^M$ , and the permutations  $\sigma$  and  $\pi$ . Then determine  $d_1, d_3, d_5$  using combinatorial map (6);  
Compute  $(\mathbf{d}'_i, E_H(\mathbf{d}_i)')$   $i \in \{2, 4, 6\}$  according to (11) and use Hamming decoder to decode  $(\mathbf{d}'_i, E_H(\mathbf{d}_i)')$  and obtain  $\mathbf{d}_i$  for  $i \in \{2, 4, 6\}$ ;  
Output  $d = (d_1, d_2, d_3, d_4, d_5, d_6)$ .

---

where  $\tau$  is the permutation by which  $\{\mathbf{x}'_i\}_{i=1}^M$  are sorted according to their  $L/3 + 1, \dots, 2L/3 - 1$  entries, and  $\rho$  is the permutation by which they are sorted according to their  $2L/3 + 1, \dots, L - 1$  entries (we emphasize that  $\tau$  and  $\rho$  are unrelated to the original  $\pi$  and  $\sigma$ , and those will be decoded later). Further, when ordering  $\{\mathbf{x}'_i\}_{i=1}^M$  by either the lexicographic ordering, by  $\tau$ , or by  $\rho$ , 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<sup>8</sup>, as follows for  $i \in [M]$ .

$$\begin{aligned}
d'_{2,i} &= x'_{i,L/3}, \\
&\quad \text{for } i \in [M - \lceil \log ML \rceil - \lceil \log M \rceil], \\
E_H(\mathbf{d}_2)'_i &= x'_{i+(M-\lceil \log ML \rceil - \lceil \log M \rceil), L/3}, \\
&\quad \text{for } i \in [\lceil \log M \rceil], \\
E_H(\mathbf{s}_1)'_i &= x'_{i+(M-\lceil \log ML \rceil), L/3}, \\
&\quad \text{for } i \in [\lceil \log ML \rceil], \\
d'_{4,i} &= x'_{\tau(i), 2L/3}, \\
&\quad \text{for } i \in [M - \lceil \log ML \rceil - \lceil \log M \rceil], \\
E_H(\mathbf{d}_4)'_i &= x'_{\tau(i+(M-\lceil \log ML \rceil - \lceil \log M \rceil)), 2L/3}, \\
&\quad \text{for } i \in [\lceil \log M \rceil], \\
E_H(\mathbf{s}_2)'_i &= x'_{\tau(i+(M-\lceil \log ML \rceil)), 2L/3}, \\
&\quad \text{for } i \in [\lceil \log ML \rceil], \\
d'_{6,i} &= x'_{\rho(i), L}, \\
&\quad \text{for } i \in [M - \lceil \log ML \rceil - \lceil \log M \rceil], \\
E_H(\mathbf{d}_6)'_i &= x'_{\rho(i+(M-\lceil \log ML \rceil - \lceil \log M \rceil)), L}, \\
&\quad \text{for } i \in [\lceil \log M \rceil], \\
E_H(\mathbf{s}_3)'_i &= x'_{\rho(i+(M-\lceil \log ML \rceil)), L},
\end{aligned}$$

<sup>8</sup>That is, each one of  $\mathbf{d}'_2, \mathbf{d}'_4, \dots$ , is obtained from  $\mathbf{d}_2, \mathbf{d}_4, \dots$ , by at most a single substitution.

$$\text{for } i \in [\lceil \log ML \rceil], \quad (11)$$

For example, if we order  $\{\mathbf{x}'_i\}_{i=1}^M$  according to  $\tau$ , then the bottom  $\lceil \log(ML) \rceil$  bits of the  $(2L/3)$ -th column are  $E_H(\mathbf{s}_2)'$ , the middle  $\lceil \log M \rceil$  bits are  $E_H(\mathbf{d}_4)'$ , and the top  $M - \lceil \log ML \rceil - \lceil \log M \rceil$  bits are  $\mathbf{d}'_4$  (see Eq. (9)). Now, let

$$\begin{aligned}
\mathbf{s}'_1 &= (\hat{\mathbf{a}}_1, \dots, \hat{\mathbf{a}}_M, \hat{\mathbf{b}}_{\tau(1)}, \dots, \hat{\mathbf{b}}_{\tau(M)}, \\
&\quad \hat{\mathbf{c}}_{\rho(1)}, \dots, \hat{\mathbf{c}}_{\rho(M)}), \\
\mathbf{s}'_2 &= (\hat{\mathbf{a}}_{\tau^{-1}(1)}, \dots, \hat{\mathbf{a}}_{\tau^{-1}(M)}, \hat{\mathbf{b}}_1, \dots, \hat{\mathbf{b}}_M, \\
&\quad \hat{\mathbf{c}}_{\tau^{-1}\rho(1)}, \dots, \hat{\mathbf{c}}_{\tau^{-1}\rho(M)}), \quad \text{and} \quad (12) \\
\mathbf{s}'_3 &= (\hat{\mathbf{a}}_{\rho^{-1}(1)}, \dots, \hat{\mathbf{a}}_{\rho^{-1}(M)}, \hat{\mathbf{b}}_{\rho^{-1}\tau(1)}, \dots, \hat{\mathbf{b}}_{\rho^{-1}\tau(M)}, \\
&\quad \hat{\mathbf{c}}_1, \dots, \hat{\mathbf{c}}_M).
\end{aligned}$$

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

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

*Proof.* If the substitution did not occur at either of index sets  $\{1, \dots, L/3 - 1\}$ ,  $\{L/3 + 1, \dots, 2L/3 - 1\}$ , or  $\{2L/3 + 1, \dots, L - 1\}$  (which correspond to the values of the strings  $\{\mathbf{a}_i\}_{i=1}^M$ ,  $\{\mathbf{b}_i\}_{i=1}^M$ , and  $\{\mathbf{c}_i\}_{i=1}^M$ , respectively), then the orders among the strings  $\{\mathbf{a}_i\}_{i=1}^M$ ,  $\{\mathbf{b}_i\}_{i=1}^M$ , and  $\{\mathbf{c}_i\}_{i=1}^M$  are maintained, respectively. That is, we have that  $\tau = \sigma$  and  $\rho = \pi$ . This implies that

$$\begin{aligned}
\mathbf{s}'_1 &= (\mathbf{a}_1, \dots, \mathbf{a}_M, \mathbf{b}_{\sigma(1)}, \dots, \mathbf{b}_{\sigma(M)}, \\
&\quad \mathbf{c}_{\pi(1)}, \dots, \mathbf{c}_{\pi(M)}), \\
\mathbf{s}'_2 &= (\mathbf{a}_{\sigma^{-1}(1)}, \dots, \mathbf{a}_{\sigma^{-1}(M)}, \mathbf{b}_1, \dots, \mathbf{b}_M, \\
&\quad \mathbf{c}_{\sigma^{-1}\pi(1)}, \dots, \mathbf{c}_{\sigma^{-1}\pi(M)}), \\
\mathbf{s}'_3 &= (\mathbf{a}_{\pi^{-1}(1)}, \dots, \mathbf{a}_{\pi^{-1}(M)}, \mathbf{b}_{\pi^{-1}\sigma(1)}, \dots, \mathbf{b}_{\pi^{-1}\sigma(M)}, \\
&\quad \mathbf{c}_1, \dots, \mathbf{c}_M),
\end{aligned}$$

and that  $d_H(E_H(\mathbf{s}_i), E_H(\mathbf{s}'_i)) \leq 1$ ,  $i \in [3]$ , according to (9) and (11). In this case, 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  $[L/3 - 1]$ .

Let  $A \in \{0, 1\}^{M \times L}$  be a matrix whose rows are the  $\mathbf{x}_i$ 's, in any order. Let  $A_{\text{left}}$  be the result of ordering the rows of  $A$  according to the lexicographic order of their  $1, \dots, L/3 - 1$  entries. Similarly, let  $A_{\text{mid}}$  and  $A_{\text{right}}$  be the results of ordering the rows of  $A$  by their  $L/3 + 1, \dots, 2L/3 - 1$  and  $2L/3 + 1, \dots, L - 1$  entries, respectively, and let  $A'_{\text{left}}, A'_{\text{mid}},$  and  $A'_{\text{right}}$  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  $A_{\text{mid}} = P_1 A_{\text{left}}$  and  $A_{\text{right}} = P_2 A_{\text{left}}$ . 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  $A'_{\text{mid}} = P_1(A_{\text{left}} + R)$  and  $A'_{\text{right}} = P_2(A_{\text{left}} + R)$ , where  $R \in \{0, 1\}^{M \times L}$  is a matrix of Hamming weight 1; this clearly implies that  $A'_{\text{mid}} = A_{\text{mid}} + P_1 R$  and that  $A'_{\text{right}} = A_{\text{right}} + P_2 R$ . Now, notice that  $\mathbf{s}_2$  results from vectorizing some submatrix  $M_2$  of  $A_{\text{mid}}$ , and  $\mathbf{s}'_2$  results from vectorizing

some submatrix  $M'_2$  of  $A'_{mid}$ . 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, no substitution occurs in the  $L/3, \dots, L$  entries in the  $\mathbf{x}_i$ 's, which implies that  $x'_{\tau(i), 2L/3} = x_{\pi(i), 2L/3}$ . Then, the redundancies  $E_H(\mathbf{s}_2)' = E_H(\mathbf{s}_2)$  and  $E_H(\mathbf{s}_3)' = E_H(\mathbf{s}_3)$  can be identified from (11). Therefore, it follows from  $A'_{mid} = A_{mid} + P_1R$  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.  $\square$

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 9, 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  $\sigma$  and  $\pi$  by observing the resulting permutation 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, we have that

$$d_H((\mathbf{d}'_i, E_H(\mathbf{d}'_i)), (\mathbf{d}_i, E_H(\mathbf{d}_i))) \leq 1$$

for  $i \in \{2, 4, 6\}$ , and thus that 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

$$\begin{aligned} r(\mathcal{C}) &= \log \binom{2^L}{M} \\ &\quad - \log \left( \binom{2^{L/3-1}}{M} \right)^3 \cdot (M!)^2 \cdot 2^{3(M - \log ML - \log M - 2)} \\ &= \log \prod_{i=0}^{M-1} (2^L - i) - \log \prod_{i=0}^{M-1} (2^{L/3-1} - i)^3 \\ &\quad - 3M + 3 \log ML + 3 \log M + 6 \\ &= \log \prod_{i=0}^{M-1} \frac{(2^L - i)}{(2^{L/3} - 2i)^3} + 3 \log ML + 3 \log M + 6 \\ &\leq 3M \log \frac{2^{L/3}}{2^{L/3} - 2M} + 3 \log ML + 3 \log M + 6. \\ &\stackrel{(a)}{\leq} 12 \log e + 3 \log ML + 3 \log M + 6 \end{aligned} \quad (13)$$

where inequality (a) is derived in Appendix B.

For the case when  $M < \log ML + \log M$ , we generate  $\{\mathbf{a}_i\}_{i=1}^M$ ,  $\{\mathbf{b}_i\}_{i=1}^M$ , and  $\{\mathbf{c}_i\}_{i=1}^M$  with length  $L/3 -$

$\lceil \frac{\log ML + \log M}{M} \rceil$ . As a result, we have  $\lceil \frac{\log ML + \log M}{M} \rceil$  bits  $x_{i,j}$ ,  $i \in [M]$ ,  $j \in \{L/3 - \lceil \frac{\log ML + \log M}{M} \rceil + 1, \dots, L/3\} \cup \{2L/3 - \lceil \frac{\log ML + \log M}{M} \rceil + 1, \dots, 2L/3\} \cup \{L - \lceil \frac{\log ML + \log M}{M} \rceil + 1, \dots, L\}$  to accommodate the information bits  $\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_6$  and the redundancy bits  $\{E_H(\mathbf{s}_i)\}_{i=1}^3$  and  $\{E_H(\mathbf{d}_i)\}_{i \in \{2, 4, 6\}}$  in each part.

**Remark 4.** *The above construction is valid whenever  $M \leq 2^{L/3-1}$ . However, asymptotically optimal amount of redundancy is achieved for  $M \leq 2^{L/6}$ .*

**Remark 5.** *In this construction, the separate storage of the Hamming code redundancies  $E_H(\mathbf{d}_2), E_H(\mathbf{d}_4)$ , and  $E_H(\mathbf{d}_6)$  is not necessary. Instead, storing  $E_H(\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_6)$  is sufficient, since the true position of those can be inferred after  $\{\mathbf{a}_i\}_{i=1}^M$ ,  $\{\mathbf{b}_i\}_{i=1}^M$ , and  $\{\mathbf{c}_i\}_{i=1}^M$  were successfully decoded. This approach results in redundancy of  $3 \log ML + \log 3M + O(1)$ , and a similar approach can be utilized in the next section as well.*

## VI. CODES FOR MULTIPLE SUBSTITUTIONS

In this section we extend the 1-substitution correcting code from Section V to multiple substitutions whenever the number of substitutions  $K$  is at most  $L/(4 \log M) - 1/2$ . In particular, we obtain the following result.

**Theorem 5.** *For integers  $M, L$ , and  $K$  such that  $M \leq 2^{\frac{L}{2(2K+1)}}$  there exists a  $K$ -substitution code with redundancy*

$$2K(2K+1) \log ML + 2K(2K+1) \log M + O(K).$$

We restrict our attention to values of  $M, L$ , and  $K$  for which  $2K \lceil \log ML \rceil + 2K \lceil \log M \rceil \leq M$ . For the remaining values, i.e., when  $2K \lceil \log ML \rceil + 2K \lceil \log M \rceil > M$ , a similar code can be constructed. The construction of a  $K$ -substitution correcting code is similar in spirit to the single substitution case, except that we partition the strings to  $2K+1$  parts instead of 3. In addition, we use a Reed-Solomon code in its binary representation (see Section II) to combat  $K$ -substitutions in the classic sense. The motivation behind considering  $2K+1$  parts is that  $K$  substitutions can affect at most  $K$  of them. As a result, at least  $K+1$  parts retain their original order; and that enables a classic RS decoding algorithm to succeed. In turn, the true values of the parts are decided by a majority vote, which is applied over a set of  $2K+1$  values,  $K+1$  of whom are guaranteed to be correct.

For parameters  $M, L$ , and  $K$  as above, let

$$D = \left[ \binom{2^{L/(2K+1)-1}}{M} \right]^{2K+1} \cdot (M!)^{2K} \cdot 2^{(2K+1)(M-2K \log ML - 2K \log M)}$$

be the information set. We split a message  $d \in D$  into  $d = (d_1, \dots, d_{4K+2})$ , where  $d_1 \in \left[ \binom{2^{L/(2K+1)-1}}{M} \right]$ ,  $d_j \in \left[ \binom{2^{L/(2K+1)-1}}{M} M! \right]$  for  $j \in \{2, \dots, 2K+1\}$ , and  $d_j \in \left[ 2^{(2K+1)(M-2K \log ML - 2K \log M)} \right]$  for  $j \in \{2K+2, \dots, 4K+2\}$ . As in (6), we apply  $F_{com}$  and  $F$  to obtain

$$F_{com}(d_1) = \{\mathbf{a}_{1,1}, \dots, \mathbf{a}_{M,1}\},$$

where  $\mathbf{a}_{i,1} \in \{0, 1\}^{L/(2K+1)-1}$  for all  $i$ , and

$F(d_j) = (\{\mathbf{a}_{1,j}, \dots, \mathbf{a}_{M,j}\}, \pi_j)$  for all  $j \in \{2, \dots, 2K+1\}$ ,  
where  $\mathbf{a}_{i,j} \in \{0, 1\}^{L/(2K+1)-1}$  and  $\pi_j \in S_M$ .

As usual, the sets  $\{\mathbf{a}_{i,j}\}_{i=1}^M$  are indexed lexicographically according to  $i$ , i.e.,  $\mathbf{a}_{1,j} < \dots < \mathbf{a}_{M,j}$  for all  $j$ . Similar to (8), let

$$\begin{aligned} & (x_{i,(j-1)L/(2K+1)+1}, \dots, x_{i,jL/(2K+1)-1}) \\ & = \mathbf{a}_{\pi_j(i),j}, \quad i \in [M], \quad j \in [2K+1], \end{aligned}$$

where  $\pi_1(i) = i$  is the identity permutation. In addition, define the equivalents of (7) as

$$\begin{aligned} \mathbf{s}_1 &= (\mathbf{a}_{1,1}, && \dots, && \mathbf{a}_{M,1}, \\ & \mathbf{a}_{\pi_2(1),2}, && \dots, && \mathbf{a}_{\pi_2(M),2}, \dots \\ & \mathbf{a}_{\pi_{2K+1}(1),2K+1}, && \dots, && \mathbf{a}_{\pi_{2K+1}(M),2K+1}), \\ \mathbf{s}_2 &= (\mathbf{a}_{\pi_2^{-1}(1),1}, && \dots, && \mathbf{a}_{\pi_2^{-1}(M),1}, \\ & \mathbf{a}_{1,2}, && \dots, && \mathbf{a}_{M,2}, \dots \\ & \mathbf{a}_{\pi_2^{-1}\pi_{2K+1}(1),2K+1}, \dots, && \mathbf{a}_{\pi_2^{-1}\pi_{2K+1}(M),2K+1}), \\ & \vdots \\ \mathbf{s}_{2K+1} &= (\mathbf{a}_{\pi_{2K+1}^{-1}(1),1}, && \dots, && \mathbf{a}_{\pi_{2K+1}^{-1}(M),1}, \\ & \mathbf{a}_{\pi_{2K+1}^{-1}\pi_2(1),2}, && \dots, && \mathbf{a}_{\pi_{2K+1}^{-1}\pi_2(M),2}, \dots \\ & \mathbf{a}_{1,2K+1}, && \dots, && \mathbf{a}_{M,2K+1}). \end{aligned}$$

Namely, for every  $i \in [2K+1]$ , the elements  $\{\mathbf{a}_{i,j}\}_{j=1}^M$  appear in  $s_i$  by their lexicographic order, and the remaining ones are sorted accordingly.

To state the equivalent of (9), for a binary string  $\mathbf{t}$  let  $RS_K(\mathbf{t})$  be the redundancy bits that result from  $K$ -substitution correcting RS encoding of  $\mathbf{t}$ , in its binary representation<sup>9</sup>. In particular, we employ an RS code which corrects  $K$  substitutions, and incurs  $2K \log(|\mathbf{t}|)$  bits of redundancy. Then, the remaining bits  $\{x_{i, \frac{L}{2K+1}}\}_{i=1}^M$ ,  $\{x_{i, \frac{2L}{2K+1}}\}_{i=1}^M$ ,  $\dots$ ,  $\{x_{i, L}\}_{i=1}^M$  are defined as follows for  $i \in [M]$  and  $j \in [1, 2K+1]$ .

$$\begin{aligned} & x_{i, \frac{jL}{2K+1}} \\ & = \begin{cases} d_{j+2K+1, \pi_j^{-1}(i)} \\ \text{if } \pi_j^{-1}(i) \in [M - 2K \lceil \log M \rceil - 2K \lceil \log ML \rceil] \\ RS_K(\mathbf{d}_{j+2K+1})_{\pi_j^{-1}(i) - M - 2K \lceil \log M \rceil - 2K \lceil \log ML \rceil}, \\ \text{if } \pi_j^{-1}(i) \in [M - 2K \lceil \log M \rceil - 2K \lceil \log ML \rceil + 1, \\ \quad \quad \quad M - 2K \lceil \log ML \rceil] \\ RS_K(\mathbf{s}_j)_{\pi_j^{-1}(i) - M - 2K \lceil \log ML \rceil}, \\ \text{if } \pi_j^{-1}(i) \in [M - 2K \lceil \log ML \rceil + 1, M] \end{cases} \end{aligned} \quad (14)$$

In this expression, notice that  $|\mathbf{s}_i| = M(L - 2K - 1)$  for every  $i$  and  $|\mathbf{d}_j| \leq M$  for every  $j$ . As a result, it follows that  $|RS_K(\mathbf{d}_j)| \leq 2K \lceil \log M \rceil$  for every  $j \in$

<sup>9</sup>To avoid uninteresting technical details, it is assumed henceforth that RS encoding in its binary form is possible, i.e., that  $\log(|\mathbf{t}|)$  is an integer that divides  $\mathbf{t}$ ; this can always be attained by padding with zeros. Furthermore, the existence of an RS code is guaranteed, since  $q = 2^{\lceil \log(|\mathbf{t}|) \rceil}$  is larger than the length of the code, which is  $|\mathbf{t}| / \log(|\mathbf{t}|)$ .

$\{2K+2, \dots, 4K+2\}$ , and  $|RS_K(\mathbf{s}_i)| \leq 2K \lceil \log ML \rceil$  for every  $i \in [2K+1]$ .

To verify that the above construction provides a  $K$ -substitution code, observe first that  $\{\mathbf{a}_{i,j}\}_{j=1}^M$  is a set of distinct integers for all  $i \in [2K+1]$ , and hence  $d_H(\mathbf{x}_i, \mathbf{x}_j) \geq 2K+1$  for all distinct  $i$  and  $j$  in  $[M]$ . Thus, a  $K$ -substitution error cannot turn one  $\mathbf{x}_i$  into another, and the result is always in  $\binom{\{0,1\}^L}{M}$ .

The decoding procedure also resembles the one in Section V. Upon receiving a word  $C' = \{\mathbf{x}'_1, \dots, \mathbf{x}'_M\} \in \mathcal{B}_K(C)$  for some codeword  $C$ , where the indexing of the elements of  $C'$  is lexicographic, we define

$$\begin{aligned} \hat{\mathbf{a}}_{i,j} &= (x'_{\tau_j^{-1}(i), \frac{(j-1)L}{2K+1}+1}, \dots, x'_{\tau_j^{-1}(i), \frac{jL}{2K+1}-1}), \\ & \text{for } j \in [2K+1], \text{ and } i \in [M] \end{aligned}$$

where  $\tau_j$  is the permutation by which  $\{\mathbf{x}'_i\}_{i=1}^M$  are sorted according to their  $\frac{(j-1)L}{2K+1} + 1, \dots, \frac{jL}{2K+1} - 1$  entries ( $\tau_1$  is the identity permutation, compare with (10)). In addition, sorting  $\{\mathbf{x}'_i\}_{i=1}^M$  by either one of  $\tau_j$  yields *candidates* for  $\{RS_K(\mathbf{s}_i)\}_{i=1}^{2K+1}$ , for  $\{\mathbf{d}_j\}_{j=2K+2}^{4K+2}$ , and for  $\{RS_K(\mathbf{d}_j)\}_{j=2K+2}^{4K+2}$ , as follows.

$$\begin{aligned} d'_{j+2K+1,i} &= x'_{\pi_j(i), \frac{jL}{2K+1}}, \\ & \text{for } i \in [M - 2K \lceil \log ML \rceil - 2K \lceil \log M \rceil] \\ RS_K(\mathbf{d}_{j+2K+1})'_i &= x'_{\pi_j(i + (M - 2K \lceil \log ML \rceil - 2K \lceil \log M \rceil)), \frac{jL}{2K+1}}, \\ & \text{for } i \in [2K \lceil \log M \rceil] \\ RS_K(\mathbf{s}_j)'_i &= x'_{\pi_j(i + (M - 2K \lceil \log ML \rceil)), \frac{jL}{2K+1}}, \\ & \text{for } i \in [2K \lceil \log ML \rceil] \end{aligned} \quad (15)$$

The respective  $\{\mathbf{s}'_i\}_{i=1}^{2K+1}$  are defined as

$$\begin{aligned} \mathbf{s}'_1 &= (\hat{\mathbf{a}}_{1,1}, && \dots, && \hat{\mathbf{a}}_{M,1}, \\ & \hat{\mathbf{a}}_{\tau_2(1),2}, && \dots, && \hat{\mathbf{a}}_{\tau_2(M),2}, \dots \\ & \hat{\mathbf{a}}_{2K+1, \tau_{2K+1}(1)}, && \dots, && \hat{\mathbf{a}}_{2K+1, \tau_{2K+1}(M)}, \\ & RS_K(\mathbf{s}_1)', \\ & \hat{\mathbf{a}}_{\tau_{2K+1}(1), 2K+1}, && \dots, && \hat{\mathbf{a}}_{\tau_{2K+1}(M), 2K+1}), \\ \mathbf{s}'_2 &= (\hat{\mathbf{a}}_{\tau_2^{-1}(1),1}, && \dots, && \hat{\mathbf{a}}_{\tau_2^{-1}(M),1}, \\ & \hat{\mathbf{a}}_{1,2}, && \dots, && \hat{\mathbf{a}}_{M,2}, \dots \\ & \hat{\mathbf{a}}_{2K+1, \tau_2^{-1}\tau_{2K+1}(1)}, \dots, && \hat{\mathbf{a}}_{2K+1, \tau_2^{-1}\tau_{2K+1}(M)}, \\ & RS_K(\mathbf{s}_2)', \\ & \hat{\mathbf{a}}_{\tau_2^{-1}\tau_{2K+1}(1), 2K+1}, \dots, && \hat{\mathbf{a}}_{\tau_2^{-1}\tau_{2K+1}(M), 2K+1}), \\ & \vdots \\ \mathbf{s}'_{2K+1} &= (\hat{\mathbf{a}}_{\tau_{2K+1}^{-1}(1),1}, && \dots, && \hat{\mathbf{a}}_{\tau_{2K+1}^{-1}(M),1}, \\ & \hat{\mathbf{a}}_{\tau_{2K+1}^{-1}\tau_2(1),2}, && \dots, && \hat{\mathbf{a}}_{\tau_{2K+1}^{-1}\tau_2(M),2}, \dots \\ & \hat{\mathbf{a}}_{2K+1,1}, && \dots, && \hat{\mathbf{a}}_{2K+1,M}, \\ & RS_K(\mathbf{s}_{2K+1})', \\ & \hat{\mathbf{a}}_{1,2K+1}, && \dots, && \hat{\mathbf{a}}_{M,2K+1}). \end{aligned}$$

**Lemma 10.** *There exist  $K + 1$  distinct integers  $\ell_1, \dots, \ell_{K+1}$  such that  $d_H((s'_{\ell_j}, RS_K(s_{\ell_j})), (s_{\ell_j}, RS_K(s_{\ell_j}))) \leq K$  for every  $j \in [K + 1]$ .*

*Proof.* Analogous to the proof of Lemma 9. See Appendix A for additional details.  $\square$

By applying an RS decoding algorithm on each of  $\{s'_i\}_{i=1}^{2K+1}$  we obtain candidates for the true values of  $\{a_{i,j}\}_{j=1}^M$  for every  $i \in [2K + 1]$ . According to Lemma 10, at least  $K + 1$  of these candidate coincide, and hence the true value of  $\{a_{i,j}\}_{j=1}^M$  can be deduced by a majority vote. Once these true values are known, the decoder can sort  $\{x'_i\}_{i=1}^M$  by its  $a_{i,j}$  entries (i.e., the entries indexed by  $1, \dots, \frac{L}{2K+1} - 1$ ), and deduce the values of each  $\pi_t$ ,  $t \in \{2, \dots, 2K + 1\}$  according to the resulting permutation of  $\{a_{t,\ell}\}_{\ell=1}^M$  in comparison to their lexicographic one. Having all the permutations  $\{\pi_j\}_{j=2}^{2K+1}$ , the decoder can extract the true positions of  $\{d_j\}_{j=2K+2}^{4K+2}$  and  $\{RS_K(d_j)\}_{j=2K+2}^{4K+2}$ , and apply an RS decoder to correct any substitutions that might have occurred.

**Remark 6.** *Notice that the above RS code in its binary representation consists of binary substrings that represent elements in a larger field. As a result, this code is capable of correcting any set of substitutions that are confined to at most  $K$  of these substrings. Therefore, our code can correct more than  $K$  substitutions in many cases.*

For  $4 \leq M \leq 2^{L/2(2K+1)}$ , the total redundancy of the above construction  $\mathcal{C}$  is given by

$$\begin{aligned} r(\mathcal{C}) &= \log \binom{2^L}{M} - \log \binom{2^{L/(2K+1)-1}}{M}^{2K+1} \\ &\quad - \log(M!^{2K} 2^{(2K+1)(M-2K \log ML - 2K \log M)}) \\ &\stackrel{(b)}{\leq} (2K + 1) \log e + 2K(2K + 1) \log ML \\ &\quad + 2K(2K + 1) \log M. \end{aligned} \quad (16)$$

where the proof of inequality (b) is given in Appendix B.

**Remark 7.** *As mentioned in Remark 5, storing  $RS_K(d_j)$  separately in each part  $j \in \{2K + 2, \dots, 4K + 2\}$  is not necessary. Instead, we store  $RS_K(d_{2K+2}, \dots, d_{4K+2})$  in a single part  $j = 2K + 1$ , since the position of the binary strings  $d_j$  for  $j \in \{2K + 2, \dots, 4K + 2\}$  and the redundancy  $RS_K(d_{2K+2}, \dots, d_{4K+2})$  can be identified once  $\{a_{i,j}\}_{i \leq M, j \leq 2K+1}$  are determined. The redundancy of the resulting code is  $2K(2K + 1) \log ML + 2K \log(2K + 1)M$ .*

For the case when  $M < 2K \log ML + 2K \log M$ , we generate sequences  $a_{i,j}$ ,  $i \in [M]$ ,  $j \in [2K + 1]$  with length  $L/(2K + 1) - \lceil \frac{2K \log ML + 2K \log M}{M} \rceil$ . Then, the length  $\lceil \frac{2K \log ML + 2K \log M}{M} \rceil$  sequences  $x_{i,j}$ ,  $i \in [M]$ ,  $j \in \cup_{l=1}^{2K+1} \{(l-1)L/(2K+1) - \lceil \frac{2K \log ML + 2K \log M}{M} \rceil + 1, \dots, lL/(2K+1)\}$  are used to accommodate the information bits  $\{d_j\}_{j=2K+2}^{4K+2}$  and the redundancy bits  $\{RS_K(s_i)\}_{i=1}^{2K+1}$  and  $\{RS_K(d_j)\}_{j=2K+2}^{4K+2}$  in each part.

## VII. CONCLUSIONS AND FUTURE WORK

Motivated by novel applications in coding for DNA storage, this paper presented a channel model in which the data is sent

as a set of unordered strings, that are distorted by substitutions. Respective sphere packing arguments were applied in order to establish an existence result of codes with low redundancy for this channel, and a corresponding lower bound on the redundancy for  $K = 1$  was given by using Fourier analysis. For  $K = 1$ , a code construction was given which asymptotically achieves the lower bound. For larger values of  $K$ , a code construction whose redundancy is asymptotically  $K$  times the aforementioned upper bound was given; closing this gap is an interesting open problem. Furthermore, it is intriguing to find a lower bound on the redundancy for larger values of  $K$  as well.

## REFERENCES

- [1] Z. Chang, J. Chrisnata, M. F. Ezerman, and H. M. Kiah, "Rates of DNA string profiles for practical values of read lengths," *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7166–7177, 2017.
- [2] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, no. 6102, pp. 1628–1628, 2012.
- [3] R. O'Donnell. Analysis of boolean functions. Cambridge University Press, 2014.
- [4] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, no. 6328, pp. 950–954, 2017.
- [5] 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.
- [6] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric Lee distance codes for DNA-based storage," *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 4982–4995, 2017.
- [7] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipes, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, no. 7435, pp. 77–80, 2013.
- [8] 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.
- [9] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE International Symposium on Information Theory (ISIT)*, pp. 814–818, 2015.
- [10] D. E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions (Art of Computer Programming)*, Addison-Wesley Professional, 2005.
- [11] 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.
- [12] M. Kovačević and D. Vukobratović, "Perfect codes in the discrete simplex," *Designs, Codes and Cryptography*, vol. 75, no. 1, pp. 81–95, 2015.
- [13] M. Langberg, M. Schwartz, and E. Yaakobi, "Coding for the  $\ell_\infty$ -limited permutation channel," *IEEE Transactions on Information Theory*, vol. 63, no. 12, pp. 7676–7686, 2017.
- [14] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2331–2351, 2020.
- [15] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Anchor-based correction of substitutions in indexed sets," *IEEE International Symposium on Information Theory (ISIT)*, Paris, France, 2019.
- [16] M. Luby, "LT codes," *The 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.
- [17] L. Organick, S. D. Ang, Y. J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. Takahashi, S. Newman, H. Y. Parker, C. Rashtchian, G. G. K. Stewart, R. Carlson, J. Mulligan, D. Carmean, G. Seelig, L. Ceze, and K. Strauss, "Scaling up DNA data storage and random access retrieval," *bioRxiv*, 2017.
- [18] N. Raviv, M. Schwartz, and E. Yaakobi, "Rank-modulation codes for DNA storage with shotgun sequencing," *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 50–64, 2019.
- [19] R. Roth, *Introduction to coding theory*, Cambridge University Press, 2006.
- [20] T. Shinkar, E. Yaakobi, A. Lenz, and A. Wachter-Zeh, "Clustering-correcting codes," *IEEE International Symposium on Information Theory (ISIT)*, Paris, France, 2019.

- [21] A. Shokrollahi, "Raptor codes," *IEEE/ACM Transactions on Networking*, vol. 14(SI), pp. 2551-2567, 2006.
- [22] I. Shomorony and R. Heckel, "Capacity results for the noisy shuffling channel," *IEEE International Symposium on Information Theory (ISIT)*, Paris, France, 2019.
- [23] J. Sima, N. Raviv and J. Bruck, "On coding over sliced information," *IEEE International Symposium on Information Theory (ISIT)*, Paris, France, 2019.
- [24] W. Song, K. Cai, and K. A. S. Immink, "Sequence-subset distance and coding for error control for DNA-based data storage," *IEEE International Symposium on Information Theory (ISIT)*, Paris, France, 2019.
- [25] 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.
- [26] 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.

## APPENDIX A PROOF OF LEMMA 10

*Proof.* (of Lemma 10) Similarly to the proof of Lemma 9, we consider a matrix  $A \in \{0, 1\}^{M \times L}$  whose rows are the  $\mathbf{x}_i$ 's, in any order. Let  $A_j$  be the result of ordering the rows of  $A$  according to the lexicographic order of their  $(j-1)L/(2K+1) + 1, \dots, jL/(2K+1) - 1$  bits for  $j \in [2K+1]$ . The matrices  $A'_j$  for  $j \in [2K+1]$  can 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  $2K+1$  permutation matrices  $P_j$  such that  $A_j = P_j A$  (Here  $P_1$  is the identity matrix). Moreover, since  $K$  substitution spoils at most  $K$  parts, there exist at least  $j_l \in [2K+1], l \in [K+1]$  such that  $\{\mathbf{a}_{i,j_l}\}_{i=1}^M = \{\hat{\mathbf{a}}_{i,j_l}\}_{i=1}^M$ , for  $l \in [K+1]$ , it follows that  $A'_{j_l} = P_{j_l}(A+R)$  for  $l \in [K+1]$ , where  $R \in \{0, 1\}^{M \times L}$  is a matrix of Hamming weight at most  $K$ ; this clearly implies that  $A'_{j_l} = A_{j_l} + P_{j_l}R$  for  $l \in [K+1]$ . Since  $\mathbf{s}_{j_l}$  results from vectorizing some submatrix  $M_l$  of  $A_{j_l}$ , and  $\hat{\mathbf{s}}_{j_l}$  results from vectorizing some submatrix  $M'_l$  of  $A'_{j_l}$ . Moreover, the matrices  $M_l$  and  $M'_l$  are taken from their mother matrix by omitting the same rows and columns, and both vectorizing operations consider the entries of  $M_l$  and  $M'_l$  in the same order. In addition, the redundancies  $E_H(\mathbf{s}_{j_l})$  for  $l \in [K+1]$  can be identified similarly, and have at most  $K$  substitution with respect to the corresponding entries in the noiseless codeword. Therefore, it follows from  $A_{j_l} = A_{j_l} + P_l R$  that  $d_H((\mathbf{s}'_{j_l}, E_H(\mathbf{s}_{j_l}')), (\mathbf{s}_{j_l}, E_H(\mathbf{s}_{j_l}))) \leq K$ .  $\square$

## APPENDIX B PROOF OF REDUNDANCY BOUNDS

*Proof of (a) in (13):*

$$\begin{aligned} r(\mathcal{C}) &\leq 3 \log\left(1 + \frac{2M}{2^{L/3} - 2M}\right)^M + 3 \log ML + 3 \log M + 6 \\ &\leq 3 \log\left(1 + \frac{4}{M}\right)^M + 3 \log ML + 3 \log M + 6 \\ &= 12 \log\left(\left(1 + \frac{4}{M}\right)^{M/4}\right) + 3 \log ML + 3 \log M + 6 \\ &\leq 12 \log e + 3 \log ML + 3 \log M + 6. \end{aligned}$$

*Proof of (b) in (16):*

$$r(\mathcal{C}) = \log \prod_{i=0}^{M-1} (2^L - i) - \log \prod_{i=0}^{M-1} (2^{L/(2K+1)-1} - i)^{2K+1}$$

$$\begin{aligned} & - \log 2^{(2K+1)M} + 2K(2K+1) \log ML \\ & + 2K(2K+1) \log M \\ & = \log \prod_{i=0}^{M-1} \frac{(2^L - i)}{(2^{L/(2K+1)} - 2i)^{2K+1}} \\ & + 2K(2K+1) \log ML + 2K(2K+1) \log M \\ & \leq (2K+1)M \log \frac{2^{L/(2K+1)}}{2^{L/(2K+1)} - 2M} \\ & + 2K(2K+1) \log ML + 2K(2K+1) \log M \\ & \leq (2K+1) \log\left(1 + \frac{2M}{2^{L/(2K+1)} - 2M}\right)^M \\ & + 2K(2K+1) \log ML + 2K(2K+1) \log M \\ & \leq (2K+1) \log\left(1 + \frac{4}{M}\right)^M + 2K(2K+1) \log ML \\ & + 2K(2K+1) \log M \\ & = (2K+1) \log\left(\left(1 + \frac{4}{M}\right)^{M/4}\right) + 2K(2K+1) \log ML \\ & + 2K(2K+1) \log M \\ & \leq (2K+1) \log e + 2K(2K+1) \log ML \\ & + 2K(2K+1) \log M. \end{aligned}$$

## APPENDIX C IMPROVED CODES FOR A SINGLE SUBSTITUTION

We briefly present an improved construction of a single substitution code, which achieves  $2 \log ML + \log 2M + O(1)$  redundancy.

**Theorem 6.** *Let  $M$  and  $L$  be numbers that satisfy  $M \leq 2^{L/4}$ . Then there exists a single substitution correcting code with redundancy  $2 \log ML + \log 2M + O(1)$ .*

The construction is based on the single substitution code as shown in Section V. The difference is that instead of using three parts and the majority rule, it suffices to use two parts (two halves) and an extra bit to indicate which part has the correct order. To compute this bit, let

$$\mathbf{x}_{\oplus} = \bigoplus_{i=1}^M \mathbf{x}_i$$

be the bitwise XOR of all strings  $\mathbf{x}_i$  and  $\mathbf{e} \in \{0, 1\}^L$  be a vector of  $L/2$  zeros followed by  $L/2$  ones. We use the bit  $b_e = \mathbf{e} \cdot \mathbf{x}_{\oplus} \bmod 2$  to indicate in which part the substitution error occurs. If a substitution error happens at the first half ( $x_i^1, \dots, x_i^{L/2}$ ), the bit  $b_e$  does not change. Otherwise the bit  $b_e$  is flipped. Moreover, as mentioned in Remark 5, we store the redundancy of all the binary strings in a single part, instead of storing the redundancy separately for each binary string in each part. The data to encode is regarded as  $d = (d_1, d_2, d_3, d_4)$ , where  $d_1 \in [{}^M_{(2^{L/2-1})}]$ ,  $d_2 \in [{}^{(2^{L/2-1})}_{M!}]$ ,  $d_3 \in [2^{M-\log ML-1}]$  and  $d_4 \in [2^{M-\log ML-2}]$ . That is,  $d_1$  represents a set of  $M$  strings of length  $L/2 - 1$ ,  $d_2$  represents a set of  $M$  strings of length  $L/2 - 1$  and a permutation  $\pi$ . Let  $\mathbf{d}_3 \in \{0, 1\}^{M-\log ML-1}$ ,  $\mathbf{d}_4 \in \{0, 1\}^{M-\log ML-2}$  be the binary strings corresponds to  $d_3$  and  $d_4$  respectively.

We now address the problem of inserting the bit  $b_e$  into the codeword. We consider the four bits  $x_{i_1, L/2}$ ,  $x_{i_2, L/2}$ ,  $x_{i_3, L}$ , and  $x_{i_4, L}$ , where  $i_1$  and  $i_2$  are the indices of the two largest strings among  $\{\mathbf{a}_i\}_{i=1}^M$  in lexicographic order, and  $i_3$  and  $i_4$  are the indices of the two largest strings among  $\{\mathbf{b}_i\}_{i=1}^M$  in lexicographic order. Then, we compute  $b_e$  and set

$$x_{i_1, L/2} = x_{i_2, L/2} = x_{i_3, L} = x_{i_4, L} = b_e.$$

Note that after a single substitution, at most one of  $i_1$ ,  $i_2$ ,  $i_3$ , and  $i_4$  will not be among the indices of the largest two strings in their corresponding part. Hence, upon receiving a word  $C' = \{\mathbf{x}'_1, \dots, \mathbf{x}'_M\} \in \mathcal{B}_1(C)$  for some codeword  $C$ , we find the two largest strings among  $\{\mathbf{a}_i\}_{i=1}^M$  and the two largest strings among  $\{\mathbf{b}_i\}_{i=1}^M$ , and use majority to determine the bit  $b_e$ . The rest of the encoding and decoding procedures are similar to the corresponding ones in Section V. We define  $\mathbf{s}_1$  and  $\mathbf{s}_2$  to the two possible concatenations of  $\{\mathbf{a}_i\}_{i=1}^M$  and  $\{\mathbf{b}_i\}_{i=1}^M$ ,

$$\begin{aligned} \mathbf{s}_1 &= (\mathbf{a}_1, \dots, \mathbf{a}_M, \mathbf{b}_{\pi(1)}, \dots, \mathbf{b}_{\pi(M)}) \\ \mathbf{s}_2 &= (\mathbf{a}_{\pi^{-1}(1)}, \dots, \mathbf{a}_{\pi^{-1}(M)}, \mathbf{b}_1, \dots, \mathbf{b}_M). \end{aligned}$$

We compute their Hamming redundancies and place them in columns  $L/2$  and  $L$ , alongside the strings  $d_3, d_4$  and their Hamming redundancy  $E_H(\mathbf{d}_3, \mathbf{d}_4)$  in column  $L$ , similar to (9).

In order to decode, we compute the value of  $b_e$  by a majority vote, which locates the substitution, and consequently, we find  $\pi$  by ordering  $\{\mathbf{x}'_i\}_{i=1}^M$  according to the error-free part. Knowing  $\pi$ , we extract the  $d_i$ 's and their redundancy  $E_H(\mathbf{d}_3, \mathbf{d}_4)$ , and complete the decoding procedure by applying a Hamming decoder. The resulting redundancy is  $2 \log ML + \log 2M + 3$ .

#### APPENDIX D PROOF OF (a) IN EQ. (5)

Note that  $P \leq T$ , it suffices to show that the function  $g(P) \triangleq ((T+P)/P)^P = (1+T/P)^P$  is increasing in  $P$  for  $P > 0$ . We now show that the derivative  $\partial g(P)/\partial P = (1+T/P)^P (\ln(1+T/P) - T/(T+P))$  is greater than 0 for  $P > 0$ . It is left to show that

$$\ln(1+T/P) > T/(T+P) \quad (17)$$

Let  $v = T/(T+P)$ , then Eq. (17) is equivalent to

$$1/(1-v) > e^v \quad (18)$$

for some  $0 < v < 1$ . The inequality (18) holds since  $1/(1-v) = 1 + \sum_{i=1}^{\infty} v^i$  and  $e^v = 1 + \sum_{i=1}^{\infty} v^i/i!$  for  $0 < v < 1$ .

**Netanel Raviv** (S'15–M'17) received a B.Sc. in mathematics and computer science in 2010, an M.Sc. and Ph.D. in computer science in 2013 and 2017, respectively, all from the Technion, Israel. He is now an Assistant Professor at the Department of Computer Science and Engineering, Washington University in St. Louis at St. Louis, MO. He is an awardee of the IBM Ph.D. fellowship for the academic year of 2015-2016, the first prize in the Feder family competition for best student work in communication technology, and the Lester-Deutsche Postdoctoral Fellowship. His research interests include applications of coding theory to privacy, distributed computations, and machine learning.

**Jehoshua Bruck** (S'86–M'89–SM'93–F'01) is the Gordon and Betty Moore Professor of computation and neural systems and electrical engineering at the California Institute of Technology (Caltech). His current research interests include information theory and systems and the theory of computation in nature.

Dr. Bruck received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion-Israel Institute of Technology, in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from Stanford University, in 1989. His industrial and entrepreneurial experiences include working with IBM Research where he participated in the design and implementation of the first IBM parallel computer; cofounding and serving as Chairman of Rainfinity (acquired in 2005 by EMC), a spin-off company from Caltech that created the first virtualization solution for Network Attached Storage; as well as cofounding and serving as Chairman of XtremIO (acquired in 2012 by EMC), a start-up company that created the first scalable all-flash enterprise storage system.

Dr. Bruck is a recipient of the Feynman Prize for Excellence in Teaching, the Sloan Research Fellowship, the National Science Foundation Young Investigator Award, the IBM Outstanding Innovation Award and the IBM Outstanding Technical Achievement Award.

**Jin Sima** is a PhD candidate in the Department of Electrical Engineering at the California Institute of Technology. He received a B.Eng. and a M.Sc. in Electronic Engineering from Tsinghua University, China, in 2013 and 2016 respectively. His research interests include information and coding theory, with its applications in data storage systems. He is a recipient of the 2019 IEEE Jack Keil Wolf ISIT Student Paper Award.