Single-Error Detection and Correction for Duplication and Substitution Channels

Yuanyuan Tang, Yonatan Yehezkeally, *Student Member, IEEE*,
Moshe Schwartz, *Senior Member, IEEE,* and Farzad Farnoud, *Member, IEEE*

***Abstract*—Motivated by mutation processes occurring in in-vivo DNA-storage applications, a channel that mutates stored strings by duplicating substrings as well as substituting symbols is studied. Two models of such a channel are considered: one in which the substitutions occur only within the duplicated substrings, and one in which the location of substitutions is unrestricted. Both error-detecting and error-correcting codes are constructed, which can handle correctly any number of tandem duplications of a fixed length k, and at most a single substitution occurring at any time during the mutation process.**

***Index Terms*—DNA storage, string-duplication systems, error correction, error detection**

I. Introduction

RECENT advances in DNA sequencing and synthesis technologies have increased the potential of DNA as a data-storage medium. In addition to its high data density, data storage in DNA provides a long-lasting alternative to current storage media. Furthermore, given the need for accessing biological data stored in DNA of living organisms, technologies for retrieving data from DNA will not become obsolete, unlike flash memory, magnetic disks, and optical disks.

Data can be stored in DNA *in vitro* or *in vivo*. While the former will likely provide a higher density, the latter can provide a more reliable and cost-effective replication method, as well as a protective shell [10]. In-vivo storage also has applications such as watermarking genetically modified organisms. This technology was recently demonstrated experimentally using CRISPR/Cas gene editing [9], [10]. One of the challenges of this technology is that a diverse set of errors are possible, including substitutions, duplications, insertions, and deletions. Duplication errors, in particular, have been previously studied by a number of recent works, including [3]–[6], [8], among others. This paper focuses on error-control codes for duplication and substitution errors.

In a (tandem) duplication event, a substring of the DNA sequence, the *template*, is duplicated and the resulting *copy* is inserted into the sequence next to the template [12]. Evidence of this process is found in the genomes of many organisms as patterns that are repeated multiple times [2]. In a substitution event, a symbol in the sequence is changed to another symbol of the alphabet. It has been observed that point mutations such as substitutions are more common in tandem repeat regions of the genomes [7]. We consider two models for combined duplication and substitution errors. In the first model, called the *noisy-duplication model*, the copy is a noisy version of the template. Noisy duplications in this model can be viewed as exact duplications followed by substitutions that are restricted to the newly added copy. Hence, this model is also referred to as the *restricted-substitution model*. We also consider an *unrestricted-substitution model*, which relaxes the noisy duplication model by allowing substitutions at any position in the sequence.

In this paper we construct both error-detecting and error-correcting codes, which are capable of correctly handling any number of tandem duplications of a fixed length k, and at most a single substitution error, which occurs at any stage during the sequence of duplication events. The main approach in both cases is to reverse the duplication process while accounting for the single substitution (which may spuriously create the appearance of a duplication that never happened, or eliminate one that did). Different challenges are also presented by the possible locations for substitutions. We bring these differences to light by providing a construction for an error-detecting code for the restricted substitution model, and an error-correcting code for the unrestricted substitution model.

Our main contributions are the following:

- We present an upper bound on the minimum required redundancy cost for detecting a single restricted substitution, over the necessary rate loss required to correct an unlimited number of duplication events, in Theorem 11. That cost is upper bounded by $O(\log(n-k))$.
- Through Construction B and Construction C, we also show that the redundancy cost (over the rate loss due to duplication noise) is upper bounded by $O(\log(k))$ and $O(k)$, respectively. While the former guarantees larger codes, it is nonconstructive, as opposed to the latter. In the likely regime where k is fixed, both require only $O(1)$ extra redundancy.

This paper was presented in part at ISIT 2019.

Yuanyuan Tang is with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, 22903, USA, (email: yt5tz@virginia.edu).

Yonatan Yehezkeally is with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer Sheva 8410501, Israel (e-mail: yonatany@bgu.ac.il).

Moshe Schwartz is with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer Sheva 8410501, Israel (e-mail: schwartz@ee.bgu.ac.il).

Farzad Farnoud (Hassanzadeh) is with the Department of Electrical and Computer Engineering and the Department of Computer Science, University of Virginia, Charlottesville, VA, 22903, USA, (email: farzad@virginia.edu).

This work was supported in part by National Science Foundation (NSF) grants under grant nos. 1816409 and 1755773, and a U.S-Israel Binational Science Foundation (BSF) grant under grant no. 2017652.

- Through Construction D, we show that the redundancy cost of detecting a single unrestricted substitution (again, over the rate loss due to duplication noise) is upper bounded by $O\big(\log(k^2 n)\big)$.
- Finally, in Theorem 26 and Theorem 28, we correct a single unrestricted substitution in addition to any number of duplications, but we incur further rate loss.

This paper is organized as follows. In Section II, we provide the notation as well as relevant background and known results. In Section III, we construct error-detecting codes for the restricted substitution model. In Section IV, we introduce error-detecting codes for unrestricted substitution channels. Finally, in Section V, we give a construction for an error-correcting code for the unrestricted substitution model. We conclude with a discussion of the results, and point out some open problems, in Section VI.

II. Notation and Preliminaries

Throughout the paper, we assume that the alphabet Σ is a unital ring of size $q \geqslant 2$ (e.g., $\mathbb{Z}_q$ or, when q is a prime power, $\mathbb{F}_q$). Thus, addition (or subtraction) and multiplications of letters from the alphabet are well-defined. The set of finite strings and strings of length at least k over Σ is denoted Σ^* and $\Sigma^{\geqslant k}$, respectively. The concatenation of two strings, $u, v \in \Sigma^*$ is denoted by uv, and u^k denotes concatenating k copies of u. To avoid confusion, the multiplication in the ring is denoted as $a \cdot b$. We say $y \in \Sigma^*$ is a substring of $w \in \Sigma^*$ if there exist $x, z \in \Sigma^*$ such that $w = xyz$.

The length (number of letters) of u is denoted by $|u|$, and for $a \in \Sigma$, we use $|u|_a$ to denote the number of occurrences of a in u. The Hamming weight of u is denoted by $\mathrm{wt}(u)$, and if $|u| = |v|$ we use $d(u, v)$ to denote the Hamming distance between u and v. If the need arises to refer to specific positions in words, positions are numbered $1, 2, \ldots$.

A (tandem) *duplication* of length k duplicates a substring of length k and inserts it in tandem into the string, namely, the copy immediately follows the template. For example, from uvw, where $|v| = k$, we may obtain $uvvw$. As an example for $k = 3$ and alphabet $\Sigma = \mathbb{Z}_3$, consider

$$x = 1012121 \to x' = 1012\underline{012}121, \tag{1}$$

where the underlined part is the copy. Since throughout the paper all duplications considered will be in tandem and of length k, we shall just use the term "duplication" to avoid cumbersome terminology.

The analysis of duplication errors will be facilitated by the *k-discrete-derivative* transform, defined in [1] in the following way. For $x \in \Sigma^{\geqslant k}$, we define $\phi(x) \triangleq \hat{\phi}(x)\bar{\phi}(x)$, where

$$\hat{\phi}(x) \triangleq x_1 \cdots x_k, \quad \bar{\phi}(x) \triangleq x_{k+1} \cdots x_n - x_1 \cdots x_{n-k},$$

in which subtraction is performed entry-wise over Σ. We note that $\phi(\cdot)$ is a bijection. The duplication length k is implicit in the definition of ϕ. For a set of strings S, we define $\phi(S) \triangleq \{\phi(s) \mid s \in S\}$.

Let x' be obtained through a tandem duplication of length k from x. It is not difficult to see that $\hat{\phi}(x) = \hat{\phi}(x')$ and that $\bar{\phi}(x')$ can be obtained from $\bar{\phi}(x)$ by inserting 0^k in an appropriate position [3]. For the example given in (1),

$$x = 1012121 \to x' = 1012\underline{012}121$$
$$\phi(x) = 101, 1112 \to \phi(x') = 101, 1\underline{000}112$$

Here, a comma separates the two parts of ϕ for clarity.

Sometimes duplications are *noisy* and the duplicated symbols are different from the original symbols. (Unless otherwise stated duplications are assumed to be exact.) We only consider the case where a single symbol is different. We view a noisy duplication as a duplication followed by a substitution in the duplicated substring. Continuing the example, the duplication resulting in x' may be followed by a substitution,

$$x' = 1012\underline{0}12121 \to x'' = 1012\underline{1}12121,$$
$$\phi(x') = 101, 1\underline{0}00\underline{1}12 \to \phi(x'') = 101, 1\underline{1}00\underline{0}12.$$

We also consider unrestricted substitutions, which can occur at any position in the string, rather than only in a substring that is duplicated by the previous duplication. A substitution may be considered as the mapping $x \to x + ae_i$, where $e_i \in \Sigma^n$ is a standard unit vector at index i, and $a \in \Sigma$, $a \neq 0$. Since ϕ is linear over Σ (i.e., $\phi(x + ae_i) = \phi(x) + a\phi(e_i)$), we denote the transform of e_i as $\epsilon_i \triangleq \phi(e_i)$, and observe that $\epsilon_i = e_i - e_{i+k}$ for $i \leqslant n - k$ and $\epsilon_i = e_i$ for $n - k < i \leqslant n$. We note that substitutions might affect two positions in the ϕ-transform domain.

Let $D^{t(p)}(x)$ (for $t \geqslant p$) denote the set of strings that can be obtained from x through t tandem duplications, p of which are noisy (in any order), with each noisy duplication containing a single substitution. $D^{t(p)}$ is called a *descendant cone* of x. Continuing our earlier examples, we have $x' \in D^{1(0)}(x)$ and $x'' \in D^{1(1)}(x)$. We further define

$$D^{*(p)}(x) \triangleq \bigcup_{t=p}^{\infty} D^{t(p)}(x), \quad D^{*(P)}(x) \triangleq \bigcup_{p \in P} D^{*(p)}(x), \tag{2}$$

where P is a subset of non-negative integers. We denote $P = \{0, 1\}$ as $\leqslant 1$.

We define $D^{t,p}(x)$ to be the set of strings obtained from x through t tandem duplications and p substitutions, where substitutions can occur in any position (and so we do not require $t \geqslant p$), and at any stage during the duplication sequence. We extend this definition similarly to (2). Obviously, for all $x \in \Sigma^*$,

$$D^{t(0)}(x) = D^{t,0}(x).$$

For a string $z \in \Sigma^*$, $\mu(z)$ is obtained by removing all copies of 0^k from z. Specifically, for

$$z = 0^{m_0} w_1 0^{m_1} w_2 \cdots w_d 0^{m_d},$$

where m_i are non-negative integers and $w_i \in \Sigma \setminus \{0\}$ are nonzero symbols, we define

$$\mu(z) \triangleq 0^{m_0 \bmod k} w_1 0^{m_1 \bmod k} w_2 \cdots w_d 0^{m_d \bmod k},$$

where k is implicit in the notation $\mu(z)$. For example, if $z = 1000112 = \bar{\phi}(x')$ from our earlier example, with $k = 3$, then $\mu(z) = 1112$; note, then, that in that example, $\mu(z) = \bar{\phi}(x)$.

Define the *duplication root* $\mathrm{drt}(x)$ of x as the unique string obtained from x by removing all tandem repeats of length k,

where the dependence on k is implicit in the notation. For proof of the uniqueness of $\mathrm{drt}(x)$ see, e.g., [3]. Note that

$$\phi(\mathrm{drt}(x)) = \hat{\phi}(x)\mu(\bar{\phi}(x))$$

(see [3]); indeed, in our running example, $x = \mathrm{drt}(x')$. For a set of strings S, we define

$$\mathrm{drt}(S) \triangleq \{\mathrm{drt}(s) \mid s \in S\}.$$

A string x is *irreducible* if $x = \mathrm{drt}(x)$. The set of irreducible strings of length n is denoted $\mathrm{Irr}(n)$, where the duplication length k is again implicit. We denote by $\mathrm{RLL}(m)$ the set of strings in Σ^m that do not contain 0^k as a substring, i.e., the $(0, k-1)$-run-length limited (RLL) constrained strings of length m. A string x of length n is irreducible if and only if $\bar{\phi}(x) \in \mathrm{RLL}(n-k)$.

A code $C \subseteq \Sigma^n$ that can correct any number of k-duplication errors is called a *k-duplication code*. We note that a code is a k-duplication code if and only if no two distinct codewords $c_1, c_2 \in C$ have a common descendant, namely,

$$D^{*,0}(c_1) \cap D^{*,0}(c_2) = \varnothing. \tag{3}$$

It was proved in [3] that this condition is equivalent to all codewords having distinct roots:

Theorem 1 ([3]) For all strings, $x_1, x_2 \in \Sigma^*$,

$$D^{*,0}(x_1) \cap D^{*,0}(x_2) \neq \varnothing$$

if and only if $\mathrm{drt}(x_1) = \mathrm{drt}(x_2)$.

Using Theorem 1, it was suggested in [3] that error-correcting codes that protect against any number of duplications may be obtained simply by taking irreducible words as codewords. Up to a minor tweaking, this strategy was shown in [3] to produce optimal codes.

Finally, we define the *redundancy* of a code $C \subseteq \Sigma^n$ as

$$r(C) \triangleq n - \log_q |C| = n - \log_{|\Sigma|} |C|,$$

and the code's *rate* as

$$R(C) \triangleq 1 - \frac{r(C)}{n}.$$

III. Restricted Error-Detecting Codes

A. The error model and the descendant cone

In this section, we consider the case of noisy-duplication errors. Our goal is to correct errors consisting of any number of exact duplications, or detect the presence of a single noisy duplication, which contains only one substitution. We refer to codes with this capability as *1-noisy duplication (1ND)* detecting. Let us first be more precise in our definition:

Definition 2 A code $C \subseteq \Sigma^*$ is a 1ND-detecting code if there exists a decoding function $\mathcal{D} : \Sigma^* \to C \cup \{\mathrm{error}\}$ such that if $c \in C$ was transmitted and $y \in \Sigma^*$ was received then $\mathcal{D}(y) = c$ if only duplication errors occurred, and $\mathcal{D}(y) \in \{c, \mathrm{error}\}$ if exactly one of the duplication errors that occurred was noisy, where the noisy duplication could have occurred at any point in the sequence of the duplication errors.

The following lemma, which relates the intersection of descendant cones to the intersection of the sets of roots of these cones, is of use in the discussion of 1ND-detecting codes.

Lemma 3 For any strings $x_1, x_2 \in \Sigma^*$ and sets $P_1, P_2 \subseteq \mathbb{Z}_{\geqslant 0}$,

$$D^{*(P_1)}(x_1) \cap D^{*(P_2)}(x_2) \neq \varnothing$$

if and only if

$$\mathrm{drt}(D^{*(P_1)}(x_1)) \cap \mathrm{drt}(D^{*(P_2)}(x_2)) \neq \varnothing.$$

Proof: The 'only if' direction follows from definition. For the other direction, assume there exist $x_1' \in D^{*(P_1)}(x_1)$ and $x_2' \in D^{*(P_2)}(x_2)$ such that $\mathrm{drt}(x_1') = \mathrm{drt}(x_2')$. But then, by Theorem 1, there exists $x \in D^{*(0)}(x_1') \cap D^{*(0)}(x_2')$. It follows that $x \in D^{*(P_1)}(x_1) \cap D^{*(P_2)}(x_2)$. This is illustrated in Figure 1, where $y = \mathrm{drt}(x_1') = \mathrm{drt}(x_2')$. ∎

We can now characterize 1ND-detecting codes in terms of duplication roots and descendant cones.

Lemma 4 A code $C \subseteq \Sigma^n$ is a 1ND-detecting k-duplication code if and only if for any two distinct codewords $c_1, c_2 \in C$,

$$D^{*(\leqslant 1)}(c_1) \cap D^{*(0)}(c_2) = \varnothing, \tag{4}$$

or equivalently,

$$\mathrm{drt}(c_2) \neq \mathrm{drt}(c_1), \tag{5}$$

$$\mathrm{drt}(c_2) \notin \mathrm{drt}(D^{*(1)}(c_1)). \tag{6}$$

Proof: Consider the following decoder: If there is a codeword with the same (exact-)duplication root as the received word, output that codeword. If not, declare that a noisy duplication error has occurred. Now, suppose (4) holds and that c_1 is transmitted. If only exact duplications occur, the decoder outputs c_1 since exact duplications do not alter the root and there is no other codeword c_2 with the same root as c_1. If, in addition, a noisy duplication occurs, then the received word either has the same root as c_1 or it does not. Note again that the duplication root of the received word only changes as a result of the noisy duplication, regardless of when it occurs in the sequence of duplication events. In the former case, the decoder correctly outputs c_1. In the latter case, (4) implies that no codeword has the same root as the received word, and thus the decoder correctly declares that a noisy duplication has occurred.

On the other hand, if (4) does not hold, no decoding method can both 'correct any number of exact duplications' and 'detect the presense of one noisy duplication'. That is because there exist distinct c_1 and c_2 and some $x \in D^{*(\leqslant 1)}(c_1) \cap D^{*(0)}(c_2)$. If x is received then there is no way to determine whether c_1 or c_2 was transmitted.

The equivalence between (4) and (5, 6) follows from Lemma 3. ∎

Based on Lemma 4, we consider codes whose distinct codewords satisfy (5) and (6). Further, the decoder outputs the codeword with the same root as the retrieved word if it exists, and otherwise declares a noisy duplication.

As a result of the substitution in the noisy duplication error, the length of the duplication root may change. One

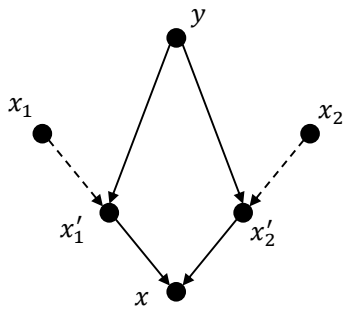

Figure 1. Illustration for the proof of Lemma 3. Solid lines denote any number of exact duplications and dashed lines represent a mixture of exact and noisy duplications (the number of noisy duplications is determined by P_1 and P_2).

way to simplify the code design is to restrict ourselves to codes whose codewords all have duplication roots with the same length. Then, error patterns that modify this length can be easily detected and we can focus on patterns that keep the duplication-root length the same. Specifically, for a given length n, we consider codes whose codewords are irreducible strings of length n. The effect of this restriction on the size of the code is discussed following Theorem 11.

Definition 5 A substitution error (as a component of a noisy-duplication error) that changes the root but not the length of the root is called an *ambiguous substitution*.

It is easy to verify that when $k = 1$ a noisy duplication is never ambiguous. Thus, challenges arise only when $k \geqslant 2$. The following sequence of lemmas characterize the conditions under which a substitution is ambiguous.

Lemma 6 Let $x \in \Sigma^*$ be some string resulting from a k-duplication, $k \geqslant 2$. If a substitution occurs (as part of a noisy duplication) in the last k positions of x then it is not ambiguous.

Proof: Since a substitution that occurs as part of a noisy duplication changes the copied part, we must have $z \triangleq \bar{\phi}(x) = u0^k w$, with $|w| \leqslant k-1$. After the substitution we get x', with $z' \triangleq \bar{\phi}(x') = u0^{k-i-1}b0^i w$, for some $b \in \Sigma \setminus \{0\}$ and $i + |w| \leqslant k-1$. It is, however, obvious that $|\mu(z)| < |\mu(z')|$, and thus $|\mathrm{drt}(x)| < |\mathrm{drt}(x')|$. ∎

Lemma 7 Let $x \in \Sigma^*$ be some string resulting from a k-duplication, $k \geqslant 2$. If x' is obtained from x as a result of a substitution that occurs (as part of a noisy duplication) in position $\ell \leqslant |x| - k$, and in $\phi(x)$ positions $\ell+1, \ldots, \ell+k-1$ contain only zeros, then the substitution is not ambiguous.

Proof: Denote $z = \bar{\phi}(x)$. Assume $z' \triangleq z + b \cdot \epsilon_{\ell-k}$ (where the subscript is indeed $\ell - k$ since by considering $\bar{\phi}(x)$ we are omitting the prefix $\hat{\phi}(x)$ of length k). Then we may write

$$\begin{array}{rcccccc} z & = & u & 0 & 0^{k-1} & b' & w \\ b \cdot \epsilon_{\ell-k} & = & 0^{|u|} & b & 0^{k-1} & (-b) & 0^{|w|} \\ z' & = & u & b & 0^{k-1} & (b'-b) & w \end{array}$$

where $u \in \Sigma^{\ell-k-1}$, $w \in \Sigma^*$, $b \in \Sigma \setminus \{0\}$, and $b' \in \Sigma$. We now have two cases. If $b' \neq b$, then obviously $|\mu(z)| < |\mu(z')|$, namely $|\mathrm{drt}(x)| < |\mathrm{drt}(x')|$. If $b' = b$, then $\mathrm{drt}(x) = \mathrm{drt}(x')$, which is again not ambiguous. ∎

The remaining cases are all handled in the following lemma.

TABLE I
EXAMPLES OF AMBIGUOUS SUBSTITUTION ERRORS FOUND IN LEMMA 8. IN ALL CASES $y = \hat{\phi}(x)$, $z = \bar{\phi}(x)$, $z' = \bar{\phi}(x')$

1a	2c
$x = 12122022002200$ $(y, z) = (121, 10200010201)$ $\mathrm{drt}(x) = 12122002200$	$x = 12122122002200$ $(y, z) = (121, 10000210201)$ $\mathrm{drt}(x) = 12122002200$
$x' = 12122022202200$ $(y, z') = (121, 10200210001)$ $\mathrm{drt}(x') = 12122022200$	$x' = 12122120002200$ $(y, z') = (121, 10001212201)$ $\mathrm{drt}(x') = 12120002200$

Lemma 8 Let $x \in \Sigma^*$ be some string resulting from a k-duplication, $k \geqslant 2$, and let x' be obtained from x as a result of a substitution that occurs as part of a noisy duplication. Denote $z \triangleq \bar{\phi}(x)$ and $z' \triangleq \bar{\phi}(x') = z + \epsilon_{\ell-k}$. Assume

$$\begin{array}{rcccccccc} z & = & u & 0^{pk+m+i-1} & 0 & 0^{k-i} & v & b' & w \\ b \cdot \epsilon_{\ell-k} & = & 0^{|u|} & 0^{pk+m+i-1} & b & 0^{k-i} & 0^{|v|} & (-b) & 0^{|w|} \\ z' & = & u & 0^{pk+m+i-1} & b & 0^{k-i} & v & (b'-b) & w \end{array}$$

where $u, w \in \Sigma^*$, $v \in \Sigma^{i-1}$, v is not empty and begins with a non-zero letter, $b \in \Sigma \setminus \{0\}$, $b' \in \Sigma$, the run of zeros 0^{pk+m+k} in z between u and v is maximal, $p \in \mathbb{Z}_{\geqslant 0}$, $0 \leqslant m < k$, $1 < i \leqslant k$. Furthermore, denote the length of the run of zeros to the left of b' in z by m_1, and to its right by m_2. Then the substitution is ambiguous exactly when either:

C.1 $1 < i \leqslant k - m$, $b' = b$, and $\lfloor \frac{m_2}{k} \rfloor < \lfloor \frac{m_1+m_2+1}{k} \rfloor$.
C.2 $k - m < i \leqslant k$ and ($b' \notin \{0, b\}$ or $\lfloor \frac{m_2}{k} \rfloor = \lfloor \frac{m_1+m_2+1}{k} \rfloor$).

Proof: The following cases are possible:

1) If $1 < i \leqslant (k-m)$ then:
   a) if $b' = b$ and $\lfloor \frac{m_2}{k} \rfloor < \lfloor \frac{m_1+m_2+1}{k} \rfloor$, then a run of 0s of length at least k will be created in z', leading to $|\mu(z')| = |\mu(z)|$ but $\mu(z') \neq \mu(z)$. Thus the substitution is ambiguous.
   b) if $b' = 0$ and $\lfloor \frac{m_2}{k} \rfloor < \lfloor \frac{m_1+m_2+1}{k} \rfloor$, then length of the root over all increases by $2k$.
   c) in all other cases, the root's length increases by k.
2) If $(k-m) < i \leqslant k$, then a run of 0s of length $m+i-1 \geqslant k$ will exist before b, implying that the length of the root before v will not change. Then:
   a) if $b' = b$ and $\lfloor \frac{m_2}{k} \rfloor < \lfloor \frac{m_1+m_2+1}{k} \rfloor$, then the length of the root decreases by k.
   b) if $b' = 0$ and $\lfloor \frac{m_2}{k} \rfloor < \lfloor \frac{m_1+m_2+1}{k} \rfloor$, then the length of the root increases by k.
   c) in all other cases, the length of the root remains the same, resulting in an ambiguous substitution.

∎

Examples for the two cases in which ambiguous substitutions occur, as described in Lemma 8, are given in Table I.

B. Bounds on the size of the code

We use the analysis of the previous section to find lower bounds on the size of 1ND-detecting codes. For $x \in \Sigma^n$, a quantity that will be useful in bounding the size of codes is the following:

$$V(x) \triangleq \left|\mathrm{drt}(D^{*(\leqslant 1)}(x)) \cap \Sigma^n\right|.$$

This counts the number of strings x' that can be obtained from x through any number of duplications, at most one of them noisy, and such that $|\mathrm{drt}(x)| = |\mathrm{drt}(x')|$.

Lemma 9 For $x \in \mathrm{Irr}(n)$, where $n \geqslant 2k \geqslant 4$,

$$V(x) \leqslant (n-k)(q-1) - \mathrm{wt}\big(\bar{\phi}(x)\big)(q-2).$$

Proof: We first assume, without loss of generality, that the noisy duplication occurs last, since subsequent duplications (which are not noisy) do not change the duplication root. Assume the notation is as defined in Lemma 8.

We first bound the contribution of the case 1a of the proof of Lemma 8 to $V(x)$. Since $n \geqslant 2k$ and x is irreducible, we have that $\mathrm{wt}(z) \geqslant 1$. There are $\mathrm{wt}(z)$ non-zero elements in z that can serve as the first letter of v, which we shall call the *anchor*. In this case, $b' \neq 0$, and it is found at most $k-m-1$ positions after the anchor. We contend that there is at most one such choice for b'. Indeed, if we are in case 1a, then there is a run of m_1 zeros immediately to the left of b', and m_2 to the right. But

$$\left\lfloor \frac{m_1+m_2+1}{k} \right\rfloor > \left\lfloor \frac{m_2}{k} \right\rfloor \geqslant 0,$$

implying

$$m_1 + m_2 + 1 \geqslant k.$$

Thus, if case 1a holds then there is a single non-zero element in the k positions following the anchor. Additionally, since $b' = b$, we have a single choice for the value of b. Finally, we note that case 1a cannot occur when the anchor is the last non-zero element in z. Hence, in total, the contribution of case 1a does not exceed $\mathrm{wt}(z) - 1$.

We now turn to the case of 2c. Assuming an anchor was chosen, the value of i can take at most m values, which is the length of the run of zeros before the anchor, taken modulo k. Ranging over all the run's zeros, the effect of modulo k simply leaves us with a choice of a position containing a 0 in z, since x is irreducible. There are $n-k-\mathrm{wt}(z)$ such positions. Finally, there are at most $q-1$ possibilities for b. Thus, this case contributes at most $(n-k-\mathrm{wt}(z))(q-1)$ to $V(x)$. Noting that x itself also contributes to $V(x)$ completes the proof. ∎

To find a lower bound on the size of the code, we apply the Gilbert-Varshamov (GV) bound with the average size of the sphere (see, e.g., [11]).

Lemma 10 Let x be a randomly and uniformly chosen string from $\mathrm{Irr}(n)$. If $n \geqslant 2k \geqslant 4$, then

$$\mathbb{E}[V(x)] \leqslant 2(n-k)(q-1)/q.$$

Proof: Let $z = \bar{\phi}(x)$. From Lemma 9, to find the expected value of $V(x)$, it suffices to find the expected value of $\mathrm{wt}(z)$.

Fix i and let U be the set of strings obtained by removing position i from the strings in $\mathrm{RLL}(n-k)$ (if multiple copies of a string exist we keep only one). Let S be the set of strings s in U that contain a run of 0s of length at least $k-1$ that includes s_{i-1} or s_i. Furthermore, let $S^c = U \setminus S$. Now, the number of strings in $\mathrm{RLL}(n-k)$ that contain a 0 in position i equals $|S^c|$, while the total number of strings in $\mathrm{RLL}(n-k)$ equals $|S^c|q + |S|(q-1)$. Hence, for a randomly chosen $z \in \mathrm{RLL}(n-k)$,

$$\Pr(z_i = 0) = \frac{|S^c|}{|S^c|q + |S|(q-1)} \leqslant \frac{1}{q}$$

Thus, $\mathbb{E}[\mathrm{wt}(z)] \geqslant (n-k)(q-1)/q$. The result then follows from Lemma 9. ∎

The above lemma leads to the lower bound in the following theorem.

Theorem 11 For positive integers $n \geqslant 2k \geqslant 4$, the maximum size $A_{1\mathrm{ND}}(n,q,k)$ of a 1ND-detecting codes of length n over $\mathbb{Z}_q$ satisfies

$$\frac{1}{4(n-k)} \cdot M \leqslant A_{1\mathrm{ND}}(n,q,k) \leqslant M,$$

where

$$M \triangleq \sum_{i=0}^{\lfloor n/k \rfloor - 1} |\mathrm{Irr}(n-ik)| = \sum_{i=1}^{\lfloor n/k \rfloor} q^k |\mathrm{RLL}(n-ik)| \quad (7)$$

is the number of irreducible words whose descendant cones intersect Σ^n.

Proof: First we show that

$$\frac{q^{k+1}|\mathrm{RLL}(n-k)|}{2(n-k)(q-1)} \leqslant A_{1\mathrm{ND}}(n,q,k) \leqslant M.$$

The lower bound follows by applying the generalized GV bound [11] with Lemma 10. The upper bound follows from the fact that the code must be able to correct any number of duplication errors and from [3] where such codes are discussed.

To get the lower bound to the more appealing form we claim, we note that to any string of length $m-k$ that has no 0^k substring, we can append a string of length k whose first element is nonzero, and thus obtain a string of length m that has no 0^k substring. Hence,

$$|\mathrm{RLL}(m)| \geqslant |\mathrm{RLL}(m-k)|(q-1)q^{k-1}.$$

Thus

$$|\mathrm{RLL}(n-ik)| \leqslant \frac{|\mathrm{RLL}(n-k)|}{(q-1)^{i-1}q^{(i-1)(k-1)}}.$$

We then have

$$\begin{aligned} M &= \sum_{i=1}^{\lfloor n/k \rfloor} q^k |\mathrm{RLL}(n-ik)| \\ &\leqslant q^k |\mathrm{RLL}(n-k)| \sum_{i=1}^{\lfloor n/k \rfloor} \frac{1}{(q-1)^{i-1} q^{(i-1)(k-1)}} \\ &\leqslant q^k |\mathrm{RLL}(n-k)| \sum_{i=1}^{\infty} \frac{1}{(q-1)^{i-1} q^{(i-1)(k-1)}} \\ &\leqslant q^k |\mathrm{RLL}(n-k)| \frac{(q-1)q^{k-1}}{(q-1)q^{k-1}-1}. \end{aligned}$$

Since $q + k \geqslant 4$,

$$|\mathrm{Irr}(n)| = q^k |\mathrm{RLL}(n-k)| \geqslant M/2, \tag{8}$$

and we have the desired claim. ∎

C. Code construction

The goal of this section is to construct 1ND-detecting codes. We shall first consider an auxiliary code construction which will be useful not only here, but also in the following section. The error we would like to detect by this auxiliary code is as follows:

Definition 12 For $n, k > 0$, let $z, z' \in \Sigma^n$ be some strings. If we can write

$$\begin{aligned} z &= u \;\; v \;\; w \;\; 0^{|v|} \;\; x \\ z' &= u \;\; 0^{|v|} \;\; w \;\; v \;\; x \end{aligned}$$

where $u, v, w, x \in \Sigma^*$, $1 \leqslant |v| \leqslant k-1$, v is a non-zero string, and $|v| + |w| = k$, then we say z and z' differ by a single k-switch error.

Intuitively, a single k-switch error takes a non-zero non-empty substring of length at most $k-1$, and switches it with an all-zero substring of the same length found k positions before or after it.

Any non-empty string $z \in \Sigma^n$ may be partitioned into non-overlapping blocks of length k:

$$z = B_1(z) B_2(z) \ldots B_{\lceil n/k \rceil}(z),$$

where $B_i(z) \in \Sigma^k$ for all i, except if k does not divide n, in which case, $B_{\lceil n/k \rceil} \in \Sigma^{n \bmod k}$. We note that k is implicit in the definition of $B_i(z)$.

We now give a construction for a family of codes which we then show are all capable of detecting a single k-switch error.

Construction A Let $k \geqslant 2$ and let p be the smallest odd integer larger than $k-1$, namely

$$p \triangleq 2\left\lceil \frac{k-1}{2} \right\rceil + 1.$$

Fix a code length $n \in \mathbb{N}$ and let $S \subseteq \Sigma^n$ be an arbitrary set of strings. For any string $x \in S$, and $\ell = 0, 1, 2, 3$, we define

$$Z_\ell(x) \triangleq \sum_{i \in I_\ell} |B_i(x)|_0,$$

where $I_\ell = \{1 \leqslant t \leqslant \lceil n/k \rceil \mid t \equiv \ell \pmod 4\}$. For all $0 \leqslant i, j < p$, we construct

$$\begin{aligned} C^{\mathrm{aux}}_{i,j}(S) \triangleq \big\{ x \in S \;\big|\; & Z_0(x) + 2Z_2(x) \equiv i \pmod p, \\ & Z_1(x) + 2Z_3(x) \equiv j \pmod p \big\}. \end{aligned}$$

Theorem 13 Each code $C^{\mathrm{aux}}_{i,j}(S)$ of Construction A can detect a single k-switch error or a single zero replaced by a non-zero letter.

Proof: Since $k \geqslant 2$ we have $p \geqslant 3$ which immediately enables the detection of a single zero replaced by a non-zero letter. Let us therefore focus on the problem of detecting a single k-switch error.

We assume $n \geqslant k+1$, otherwise the claim is trivial. Assume $x \in C^{\mathrm{aux}}_{i,j}(S)$ sustains a single k-switch error, resulting in the string $x' \in \Sigma^n$. For $0 \leqslant \ell \leqslant 3$, let

$$\Delta_\ell \triangleq Z_\ell(x') - Z_\ell(x).$$

Furthermore, for $0 \leqslant \ell \leqslant 1$, let

$$F_\ell \triangleq \Delta_\ell + 2\Delta_{\ell+2}.$$

To prove the error detection capabilities of the code it now suffices to show that

$$F_0 \not\equiv 0 \pmod p \qquad \text{or} \qquad F_1 \not\equiv 0 \pmod p. \tag{9}$$

Based on the definition of a k-switch error, the number of zeros changes in some blocks. We consider the following possible cases.

First, if the number of zeros changes in 2 consecutive blocks, then one of the pairs (Δ_0, Δ_1), (Δ_1, Δ_2), (Δ_2, Δ_3), (Δ_3, Δ_0) equals $(\delta, -\delta)$ for $0 < |\delta| < k$, and the two other Δ's are equal to 0. Then, $|F_0| = |\delta|$ or $|F_0| = 2|\delta|$. In the former case $F_0 \not\equiv 0 \pmod p$ since $0 < |\delta| < k \leqslant p$. In the latter case, $F_0 \not\equiv 0 \pmod p$ since $0 < 2|\delta| < 2p$ and $2\delta \neq p$ (recall that p is odd).

Second, if the number of zeros changes in two non-consecutive blocks, then only one of the pairs (Δ_0, Δ_2) and (Δ_1, Δ_3) equals $(\delta, -\delta)$ for $0 < |\delta| < k$, and the other equals $(0, 0)$. Then, either $|F_0| = |\delta|$ or $|F_1| = |\delta|$, and in both cases (9) is satisfied.

Third, if the change of number of zeros occurs in three consecutive blocks, then there exists ℓ such that $\Delta_\ell = \delta' \neq 0$ and $\Delta_{2+\ell} = 0$ (indices taken modulo 4), where $0 < |\delta'| < k$ and $2|\delta'| \neq p$. Then either F_0 or F_1 takes on the value of δ' or $2\delta'$. But $\delta' \not\equiv 0 \pmod p$ and $2\delta' \not\equiv 0 \pmod p$, implying that (9) is satisfied. ∎

We now turn to construct 1ND-detecting codes. As before, we consider codes that consist of irreducible strings of length n. We thus need to devise a method to detect ambiguous substitutions.

As mentioned before, when $k = 1$ ambiguous substitutions cannot occur. Hence $\mathrm{Irr}(n)$ is a 1ND-detecting code. For $k \geqslant 2$, our analysis rests on the following lemma.

Lemma 14 Let $k \geqslant 2$. If $x \in \Sigma^*$ and x' is obtained from x via any number of duplications among which one contains an

ambiguous substitution, then $\bar{\phi}(\mathrm{drt}(x))$ and $\bar{\phi}(\mathrm{drt}(x'))$ differ by a single k-switch error, or

$$\big||\bar{\phi}(\mathrm{drt}(x))|_0 - |\bar{\phi}(\mathrm{drt}(x'))|_0\big| = 1.$$

Proof: Denote $z \triangleq \bar{\phi}(x)$ and $z' \triangleq \bar{\phi}(x')$. With the notation of Lemma 8, one can verify that in Case 1a we have

$$\begin{array}{l} \mu(z) \;=\; u' \;\; v \;\; 0^{i-1-|v|}b0^{k-i} \;\; 0^{|v|} \;\; w' \\ \mu(z') \;=\; u' \;\; 0^{|v|} \;\; 0^{i-1-|v|}b0^{k-i} \;\; v \;\; w' \end{array} \tag{10}$$

and in Case 2c,

$$\begin{array}{l} \mu(z) \;=\; u' \;\; 0 \;\; 0^{k-|v|-1}v \;\;\; b' \;\;\; w' \\ \mu(z') \;=\; u' \;\; b \;\; 0^{k-|v|-1}v \;\; (b'-b) \;\; w' \end{array} \tag{11}$$

for some $u', w' \in \Sigma^*$. In (10) we see a single k-switch error. In (11), if $b' = b$ we have a single k-switch error, and if $b \neq b'$ then the number of zeros differ by one. ∎

Construction B Let n, k be positive integers, $n \geqslant k$, and let $S \triangleq \mathrm{RLL}(n-k)$. For all $0 \leqslant i, j < p$, we construct

$$C_{i,j} \triangleq \big\{\phi^{-1}(yz) \;\big|\; y \in \Sigma^k, z \in C^{\mathrm{aux}}_{i,j}(S)\big\},$$

where p and $C^{\mathrm{aux}}_{i,j}(S)$ are defined in Construction A.

Theorem 15 With the setting as in Construction B, the code $C_{i,j}$ is a 1ND-detecting code.

Proof: By our choice of S, we necessarily have that $C_{i,j} \subseteq \mathrm{Irr}(n)$. If $k = 1$, then $C_{0,0} = \mathrm{Irr}(n)$ is the only code and the theorem is immediate.

Assume $k \geqslant 2$. Let $c_1, c_2 \in C_{i,j}$ be distinct codewords. Since $C_{i,j} \subseteq \mathrm{Irr}(n)$, $\mathrm{drt}(c_1) = c_1$ and $\mathrm{drt}(c_2) = c_2$, which are distinct. Based on (6) it suffices to show that for any $c_1' \in D^{*(1)}(c_1)$, we have $c_2 \neq \mathrm{drt}(c_1')$.

If $\mathrm{drt}(c_1') = \mathrm{drt}(c_1) = c_1$, then clearly $c_2 \neq \mathrm{drt}(c_1')$. So we assume $\mathrm{drt}(c_1') \neq c_1$. It is then sufficient to show that $\mathrm{drt}(c_1') \notin C_{i,j}$. This is obvious if $|\mathrm{drt}(c_1')| \neq n$ and the substitution is not ambiguous. If the substitution is ambiguous, we obtain the claimed result by combining Lemma 14 and Theorem 13. ∎

Corollary 16 If $n \geqslant k \geqslant 2$ then

$$A_{1\mathrm{ND}}(n, q, k) \geqslant \frac{1}{2(k+1)^2} \cdot M,$$

where M is given by (7).

Proof: Let p and $C_{i,j}$ be defined as in Construction B. The set $\{C_{i,j} \;\mid\; 0 \leqslant i, j < p\}$ forms a partition of $\mathrm{Irr}(n)$. Thus, a simple averaging argument shows that there exist i and j such that

$$|C_{i,j}| \geqslant \frac{|\mathrm{Irr}(n)|}{p^2}.$$

Since $p \leqslant k+1$, and by (8), we obtain the claim. ∎

Note that the lower bound on $A_{1\mathrm{ND}}(n, q, k)$ in this corollary may be better than the one given in Theorem 11.

The problem with the bound of Corollary 16 is that it is not constructive. In particular, we do not know exactly what choice of i and j gives the largest code $C_{i,j}$ in Construction B. Construction C below provides a sub-code of $C_{0,0}$ from Construction B whose size can be lower bounded, albeit, somewhat smaller than the guarantee of Corollary 16.

Construction C Let $k \geqslant 2$ and let p be the smallest odd integer larger than $k-1$, namely

$$p \triangleq 2\left\lceil \frac{k-1}{2} \right\rceil + 1.$$

Fix a code length $n \in \mathbb{N}$, $n \geqslant 5k$. We construct a code $C \subseteq \Sigma^n$ in the following way: For each $y \in \mathrm{RLL}(n-5k)$, construct four strings of length k, denoted $B_0, B_1, B_2, B_3 \in \Sigma^k$,

$$B_i = 0^{\beta_i}1^{k-\beta_i}, \qquad \forall 0 \leqslant i \leqslant 3$$

where

$$\begin{aligned} \beta_i &= (-(\zeta_i + 2\zeta_{i+2}) \bmod p) - 2\beta_{i+2}, && i = 0, 1 \\ \beta_{i+2} &= \left\lfloor \frac{(-(\zeta_i + 2\zeta_{i+2}) \bmod p)}{2} \right\rfloor, && i = 0, 1 \\ \zeta_i &= Z_i(\phi^{-1}(0^k y)), && i = 0, 1, 2, 3 \end{aligned}$$

and add the codewords $\phi^{-1}(BB_0B_1B_2B_3y)$ where B runs over all strings in Σ^k.

Theorem 17 Let q be the alphabet size, k the duplication length, $q + k \geqslant 4$, and $n \in \mathbb{N}$, $n \geqslant 5k$. Then the code C from Construction C is a 1ND-detecting code of size

$$|C| = \mathrm{Irr}(n-4k) \geqslant \frac{1}{2 \cdot q^{4k}} \cdot M,$$

where M is given in (7).

Proof: One can easily verify that $0 \leqslant \beta_1 < k$, hence all the blocks B_i end with a non-zero symbol and therefore all the codewords are irreducible. Additionally, by inspection we can verify that $C \subseteq C_{0,0}$, where $C_{0,0}$ is obtained from Construction B. Thus, C is 1ND-detecting. Finally, all the codewords constructed are distinct, hence

$$|C| = q^k|\mathrm{RLL}(n-5k)| = |\mathrm{Irr}(n-4k)| \geqslant \frac{1}{2 \cdot q^{4k}} \cdot M,$$

where the last inequality follows from the fact that $|\mathrm{Irr}(n-4k)| \geqslant |\mathrm{Irr}(n)|/q^{4k}$ and then from (8). ∎

IV. Unrestricted Error-detecting codes

Substitution mutations might occur not only in duplication copies, but also independently in other positions. In what follows, we consider a single substitution error occurring in addition to however many duplications, at any stage during the sequence of duplication events, but not necessarily in a duplicated substring. We refer to codes correcting many duplication errors and detecting a single independent substitution error as *1S-detecting* codes.

Definition 18 A code $C \subseteq \Sigma^*$ is a 1S-detecting code if there exists a decoding function $\mathcal{D} : \Sigma^* \to C \cup \{\mathrm{error}\}$ such that if $c \in C$ was transmitted and $y \in \Sigma^*$ was received then $\mathcal{D}(y) = c$ if only duplication errors occurred, and $\mathcal{D}(y) \in \{c, \mathrm{error}\}$ if in addition to the duplications, exactly one unrestricted substitution occurred.

Lemma 19 A code $C \in \Sigma^n$ is a 1S-detecting code if and only if for any two distinct codewords $c_1, c_2 \in C$, we have

$$\mathrm{drt}(c_1) \neq \mathrm{drt}(c_2) \quad \text{and} \quad \mathrm{drt}(c_2) \notin \mathrm{drt}(D^{*,1}(c_1)). \quad (12)$$

Proof: In the one direction, we define for any $y \in \Sigma^*$, $\mathcal{D}(y) = c$ if $\mathrm{drt}(c) = \mathrm{drt}(y)$, and $\mathcal{D}(y) = \text{error}$ otherwise. Clearly if (12) holds then $\mathcal{D}$ is a decoding function proving that C is a 1S-detecting code.

In the other direction, if (4) does not hold we have two (not mutually exclusive) cases. If there exist $c_1, c_2 \in C$ such that $\mathrm{drt}(c_1) = \mathrm{drt}(c_2)$ then by Theorem 1 there exists $y \in D^{*,0}(c_1) \cap D^{*,0}(c_2)$ and no decoding function can always correctly decode y. Similarly, if $\mathrm{drt}(c_2) \in \mathrm{drt}(D^{*,1}(c_1))$ then there exists $y \in D^{*,1}(c_1)$ such that $\mathrm{drt}(y) = \mathrm{drt}(c_2)$ and no decoding function $\mathcal{D}$ can always decode y correctly. ∎

We shall adopt the same general strategy as the previous section. Namely, we will construct a code based on irreducible words of length n. Descendants whose duplication root is not of length n will be easily detected. Our challenge is therefore to detect errors that do not change the length of the root caused by, what we refer to as, ambiguous substitutions.

Definition 20 An unrestricted substitution error that changes the root but not the length of the root is called an *ambiguous unrestricted substitution*.

As in the previous section, when the duplication length is $k = 1$ there are no ambiguous unrestricted substitutions. In that case $\mathrm{Irr}(n)$ can easily serve as a 1S-detecting code. Thus, we shall focus on the case of $k \geqslant 2$.

Lemma 21 Let $n \geqslant 2k \geqslant 4$. For any string $x \in \Sigma^n$, let $x' \in \mathrm{drt}(D^{*,\leqslant 1}(x)) \cap \Sigma^n$ be a string obtained from x via a single ambiguous unrestricted substitution. If

$$d(\phi(\mathrm{drt}(x)), \phi(\mathrm{drt}(x'))) \geqslant 3,$$

then $\bar{\phi}(\mathrm{drt}(x))$ and $\bar{\phi}(\mathrm{drt}(x'))$ differ by a single k-switch error.

To improve the flow of reading, the proof of this technical lemma is given in the appendix.

Our strategy, based on Lemma 21, is to build a code as an intersection of two other component codes. If one component code can detect the swapping of two substrings and the other component code has a minimum Hamming distance of 3 or more, then their intersection is a 1S-detecting code.

Construction D Let q be a prime power, and $\Sigma \triangleq \mathbb{F}_q$ be the finite field of q elements. Let $n \geqslant k \geqslant 2$ and let r be the unique positive integer such that $\frac{q^{r-1}-1}{q-1} < n \leqslant \frac{q^r-1}{q-1}$, namely,

$$r \triangleq \left\lceil \log_q(n(q-1)+1) \right\rceil.$$

Denote by C^H the $[n, n-r, 3]$ shortened Hamming code over $\mathbb{F}_q$, and by $C_0^H, C_1^H, \ldots, C_{q^r-1}^H$ its q^r cosets. Finally, let p and $C_{i,j}$ be defined as in Construction B. For all $0 \leqslant i, j < p$ and $0 \leqslant \ell < q^r$, we construct

$$C_{i,j,\ell} \triangleq \left\{ c \in C_{i,j} \mid \phi(c) \in C_\ell^H \right\}.$$

Theorem 22 With the setting as in Construction D, the code $C_{i,j,\ell}$ is a 1S-detecting code. In particular, there exist i, j, ℓ such that

$$|C_{i,j,\ell}| \geqslant \frac{|\mathrm{Irr}(n)|}{q^r p^2} \geqslant \frac{|\mathrm{Irr}(n)|}{q(n(q-1)+1)(k+1)^2}.$$

Proof: By Construction B we have that $C_{i,j} \subseteq \mathrm{Irr}(n)$, hence also $C_{i,j,\ell} \subseteq \mathrm{Irr}(n)$, which implies it can correct any number of duplications. Thus, following Lemma 19, it only remains to consider two distinct codewords $c_1, c_2 \in C_{i,j,\ell}$ and show that $\mathrm{drt}(c_2) \notin \mathrm{drt}(D^{*,1}(c_1))$, namely consider the case in which a single ambiguous unrestricted substitution occurred as part of the duplications.

Assume to the contrary this is not the case. By Lemma 21, if $d(\phi(c_1), \phi(c_2)) \geqslant 3$, then $\bar{\phi}(c_1)$ and $\bar{\phi}(c_2)$ differ by a single k-switch error, and this is contradicts the fact that $C_{i,j}$ detects a single k-switch error in the $\bar{\phi}$ part of the root, a fact that has already been used in Theorem 15. If $d(\phi(c_1), \phi(c_2)) \leqslant 2$, then this contradicts the minimum distance implied by using the shortened Hamming code.

Finally, the existence of the code with the lower bounded size is guaranteed using a simple averaging argument since $\{C_{i,j,\ell} \mid 0 \leqslant i, j < p, 0 \leqslant \ell < q^r\}$ forms a partition of $\mathrm{Irr}(n)$. ∎

V. Unrestricted Error-Correcting Codes

In this section, we again observe the case of many tandem-duplications and a single substitution, occurring at any point during the duplication sequence, and not necessarily in a duplicated substring. However, unlike previous sections, there is no mix of correction and detection – rather we aim to *correct* all duplications and a single substitution (occurring at any stage during the sequence of duplication events), which makes the definition of the code more straightforward. We refer to codes able to correct such errors as a *single-substitution correcting* (1S-correcting) code. Obviously, a code C is 1S-correcting if and only if for any two distinct codewords $c_1, c_2 \in C$, we have

$$D^{*,\leqslant 1}(c_1) \cap D^{*,\leqslant 1}(c_2) = \varnothing.$$

In this context, we will find it easier to consider strings in the ϕ-transform domain. We also define the *substitution distance* $\sigma(u, v)$ to measure the number of substitutions required to transform one string into the other, when u, v are assumed to be in the transform domain. More precisely, if $u, v \in \Sigma^n$ and $v - u = \sum_{i=1}^n a_i \cdot \epsilon_i$, then

$$\sigma(u, v) \triangleq |\{1 \leqslant i \leqslant n \mid a_i \neq 0\}|.$$

A. Error-correcting codes

In contrast to Lemma 21 and Construction D, we shall see in the following example that an intersection of a single substitution correcting code with a duplication correcting code is not, in general, a 1S-correcting code.

Example 23 Set $\Sigma = \mathbb{Z}_2$ and $k = 3$, and observe the following two sequences of duplication and substitution, as seen in the ϕ-transform domain:

$$u \triangleq 111010111 \to 111010111\underline{000} \to 111\underline{0}00101\underline{0}1000$$
$$v \triangleq 111101010 \to 111\underline{000}101010 \to 111000101\underline{0}\underline{0}0$$

It is clear that if $C \subseteq \Sigma^{\geqslant k}$ is a code correcting even a single duplication and a single substitution, even given the order in which they occur, then $\phi^{-1}(u) = 111101010$ and $\phi^{-1}(v) = 111010000$ cannot both belong to C. Observing that $u, v \in \mathrm{RLL}(9)$ and $\sigma(u, v) = 4$, however, we find that $C \triangleq \{\phi^{-1}(u), \phi^{-1}(v)\}$ can correct any number of duplications, or correct a single substitution. Simple intersections, hence, do not suffice for a code correcting a combination of such errors. □

In what follows, we propose a constrained-coding approach which resolves the issue demonstrated in the last example. It relies on the following observation: substitution noise might create a 0^k substring in the transform domain–that is not due to a duplication–as well as break a run of zeros. However, a constrained system exists which allows us to de-couple the effects of duplication and substitution noise.

More precisely, we denote

$$\mathcal{W} \triangleq \big\{u \in \Sigma^{\geqslant k} \;\big|\; \forall \text{ substring } v \text{ of } u, |v| = k : \mathrm{wt}(v) > 1\big\}.$$

We shall show that intersecting a single-substitution-error-correcting code with the reverse image of $\mathcal{W} \cap \Sigma^{n-k}$, instead of $\mathrm{RLL}(n-k)$, is a 1S-correcting code. More precisely, we aim to show that restricting codewords to be taken from $\mathcal{W}$ (in the transform domain), the following holds.

Lemma 24 Take an irreducible $x \in \Sigma^{\geqslant k}$, and $y \in D^{*,\leqslant 1}(x)$. If $v \triangleq \bar{\phi}(y)$ contains a 0^k substring, and $\bar{v}$ is derived from v by removing that substring, and if $\bar{\phi}(x) \in \mathcal{W}$, then $\bar{v} \in \bar{\phi}\big(D^{*,\leqslant 1}(x)\big)$.

Proof: We denote

$$v = \alpha c 0^k \beta$$

for $0 \neq c \in \Sigma$ and $\alpha, \beta \in \Sigma^*$, and by abuse of notation assume $|\alpha c| \geqslant 0$ is the shortest with the properties stated above (allowing $v = 0^k\beta$ as a private case).

We also take $y' \in D^{*,0}(x)$ to be the descendant of x derived by the same sequence of duplications as y, where a substitution never occurs, and

$$v' = \bar{\phi}(y') = \alpha' c' 0^j a 0^{k-j-1} \beta',$$

for $0 \leqslant j < k$, $c', a \in \Sigma$, $\alpha', \beta' \in \Sigma^*$, where $|\alpha' c'| = |\alpha c|$. (We know v' can be represented in this fashion since y suffered a single substitution.)

If $a = 0$ then the claim is trivial. Assume, therefore, $a \neq 0$. Note that $\bar{\phi}(x) \in \mathcal{W}$ and $\mathrm{wt}_H(0^j a 0^{k-j-1}) = 1$, implying that $0^{k-j-1}\beta'$ begins with a k-tuple of zeros. I.e., $\beta' = 0^{j+1}\beta''$, for some $\beta'' \in \Sigma^*$. Thus, a descendant of x is also z', where $\bar{\phi}(z') = \alpha' c' 0^j a \beta''$.

We now reexamine v', v:

$$\begin{aligned} v' &= \alpha' \;\; c' \;\; 0^j \;\; a \;\; 0^{k-j-1} \;\; \beta' \\ v &= \alpha \;\; c \;\; 0^j \;\; 0 \;\; 0^{k-j-1} \;\; \beta \end{aligned}$$

and since y is derived from x by the same sequence of tandem-duplications as y', with a single substitution, we may deduce that α, β and α', β' differ, respectively, in precisely one of the following manners:

- There exist $b \in \Sigma$ and $\alpha_1, \alpha_2 \in \Sigma^*$, with $|\alpha_2 c| = k-j-1$, such that

  $$\begin{aligned} v' &= \alpha_1 \;\; (b-a) \;\; \alpha_2 \;\; c' \;\; 0^j \;\; a \;\; 0^{k-j-1}\beta \\ v &= \alpha_1 \;\;\;\; b \;\;\;\; \alpha_2 \;\; c \;\; 0^j \;\; 0 \;\; 0^{k-j-1}\beta \end{aligned}$$

  and, again, by abuse of notation, including the case of $|\alpha_2 c| = 0$, meaning $b = c$ and $b - a = c'$; in all other cases $c' = c$.
  In this case

  $$\begin{aligned} \bar{v} &= \alpha c \beta = \alpha_1 b \alpha_2 c 0^{j+1} \beta'' \\ &= \alpha_1 (b-a) \alpha_2 c' 0^j a \beta'' + a \cdot \epsilon_{|xc|+j-k} \\ &= \bar{\phi}(z) + a \cdot \epsilon_{|xc|+j-k}. \end{aligned}$$

- $\beta = 0^j a \beta''$, implying $\alpha' c' = \alpha c$ and

  $$\bar{v} = \alpha c \beta = \alpha c 0^j a \beta'' = \bar{\phi}(z).$$

- There exist $s \geqslant 0$, $b \in \Sigma$, $\gamma \in \Sigma^{k-1}$ and $\beta''' \in \Sigma^*$ such that $\beta'' = 0^{sk}\gamma b \beta'''$, and

  $$\begin{aligned} v' &= \alpha c \;\; 0^j \;\; a \;\; 0^{k-j-1} \;\; 0^{j+1+sk}\gamma \;\;\;\; b \;\;\;\; \beta''' \\ v &= \alpha c \;\; 0^j \;\; 0 \;\; 0^{k-j-1} \;\; 0^{j+1+sk}\gamma \;\; (b+a) \;\; \beta''' \end{aligned}$$

  Let z'' be the ancestor of z' (thus descendant of x) satisfying

  $$\bar{\phi}(z'') = \alpha c 0^j a \gamma b \beta'''$$

  and note that

  $$\begin{aligned} \bar{v} &= \alpha c 0^{j+sk} 0 \gamma (b+a) \beta''' \\ &= \alpha c 0^{j+sk} a \gamma b \beta''' + (-a) \cdot \epsilon_{|\alpha c|+sk+j} \\ &\in \bar{\phi}\big(D^{*,0}\big(z'' + (-a) \cdot e_{|\alpha c|+j}\big)\big) \end{aligned}$$

■

Recall from [3] that a decoder for correcting an unbounded number of duplications simply has to remove incidents of 0^k from the $\bar{\phi}$-part of the noisy string. This lemma shows that the same approach can be taken with the addition of a single substitution–without increasing the substitution distance–provided that coding is done in $\mathcal{W}$.

Next, we consider the case where a substitution breaks a run of zeros (in the transform domain). The following lemma allows us to remove appearances of $0^j a 0^{k-1-j}$ from the $\bar{\phi}$-part of a noisy string (by applying an appropriate substitution) without increasing the substitution distance.

Lemma 25 Suppose $u \in \Sigma^{\geqslant k}$ contains a substring 0^k starting at index i, and suppose $v = u + a \cdot \epsilon_\ell$ for some $i \leqslant j < i + k$, $0 \neq a \in \Sigma$, and $\ell \in \{j, j-k\}$ (so that $v_j \neq 0$). Note that $v' \triangleq v - v_j \cdot \epsilon_j$ has a 0^k substring at index i (like u); we remove that substring from both u, v' to produce $\bar{u}, \bar{v}$, respectively. Then, irrespective of what value ℓ takes, $\sigma(\bar{u}, \bar{v}) \leqslant 1$.

Proof: The lemma is straightforward to prove by case for ℓ. If $\ell = j$ then $v' = u$, and consequently $\bar{v} = \bar{u}$.

Otherwise, $\ell = j - k$ and $v_j = -a$, hence

$$v' = u + a \cdot (\epsilon_{j-k} + \epsilon_j)$$

and $\bar{v} = \bar{u} + a \cdot \epsilon_{j-k}$, which concludes the proof. ∎

It is therefore seen that a restriction to $\mathcal{W}$ allows the correction of the substitution error without encountering the issue demonstrated in Example 23. This fact is more precisely stated in the following theorem:

Theorem 26 If $C \subseteq \Sigma^n$, $n \geqslant k$, is an error-correcting code for a single substitution, and $\bar{\phi}(C) \subseteq \mathcal{W}$, then C is a 1S-correcting code.

Proof: Take $x \in C$, $y \in D^{*,\leqslant 1}(x)$, and define $u \triangleq \hat{\phi}(x)$, $v \triangleq \bar{\phi}(y)$. We first remove 0^k substrings from v, stopping if we reach length $n - k$. By Lemma 24, every removal of 0^k does not increase the substitution distance of the received sequence from a duplication descendant of x; if indeed it is possible to arrive at $\hat{v}$ of length $n-k$, then the error-correcting capabilities of C now suffice to deduce x from $\phi^{-1}(u\hat{v})$.

The only other possible case is that we ultimately arrive at $\hat{v}$ of length n which contains a substring of length k of weight 1. We remove that substring to obtain $\hat{v}'$, and reverse the ϕ-transform, namely, $y' \triangleq \phi^{-1}(u\hat{v}')$. By Lemma 25, this produces y' of the same length as x and differing from it by at most a single substitution, which we may once more correct in the standard fashion. ∎

B. Code Construction and Size

In this section we construct a family of codes satisfying Theorem 26. We also study the redundancy and rate of the proposed construction. We start by bounding the rate loss of using constrained coding by restricting codes to $\mathcal{W}$:

Lemma 27 For every integers $q \geqslant 2$ and $n \geqslant k \geqslant 1$,

$$\frac{r(\mathcal{W} \cap \Sigma^n)}{n} \leqslant \frac{2}{k} \log_q \frac{q}{q-1}.$$

Proof: We note that $C_n \subseteq \mathcal{W} \cap \Sigma^n$, where C_n is the set of length-n strings in which, divided into blocks of length k, every block ends with two non-zero elements. Hence,

$$\begin{aligned}\frac{r(\mathcal{W} \cap \mathbb{Z}_q^n)}{n} &\leqslant \frac{r(C_n)}{n} = \frac{1}{n}\left(\left\lfloor \frac{n}{k} \right\rfloor + \left\lfloor \frac{n+1}{k} \right\rfloor\right) \\ &\leqslant \frac{2}{k} \log_q \frac{q}{q-1}.\end{aligned}$$

∎

Theorem 28 If q is a prime power, $r \geqslant 2$, and $n = \frac{q^r-1}{q-1} + \left\lceil \frac{2r}{k} \right\rceil$, then a 1S-correcting k-duplication code $C \subseteq \mathcal{W} \cap \mathbb{F}_q^n$ exists, with

$$R(C) \geqslant 1 - \frac{2}{k} \log_q \frac{q}{q-1} - o(1).$$

Proof: We begin by encoding data into $\mathcal{W} \cap \mathbb{F}_q^{\frac{q^r-1}{q-1}-r}$, incurring by Lemma 27 redundancy

$$r\left(\mathcal{W} \cap \mathbb{F}_q^{\frac{q^r-1}{q-1}-r}\right) \leqslant \left(\frac{q^r-1}{q-1} - r\right)\frac{2}{k} \log_q \frac{q}{q-1}.$$

Next, a systematic encoder for the $\left[\frac{q^r-1}{q-1}, r, 3\right]$ Hamming code (under the change of basis to $\{\epsilon_i\}$) can encode $\mathcal{W} \cap \mathbb{F}_q^{\frac{q^r-1}{q-1}-r} \to \mathbb{F}_q^{\frac{q^r-1}{q-1}}$, incurring r additional symbols of redundancy, and resulting in a code which can correct a single substitution.

Note, due to the systematic encoding, that the projection of this code onto the first $\frac{q^r-1}{q-1} - r$ coordinates is contained in $\mathcal{W}$. We may simply cushion the last r symbols with $\left\lceil \frac{2r}{k} \right\rceil$ interleaved 1's (two per k data symbols) to achieve a code $C \subseteq \mathcal{W} \cap \mathbb{F}_q^n$ which may still correct a single substitution. ∎

Taking $n \to \infty$, we can compare the rate obtained by the code in Theorem 28 to a simple upper bound of the best codes correcting only tandem duplications of length k (see [3]),

$$R(C) \leqslant 1 - \frac{(q-1)\log_q e}{q^{k+2}} + o(1).$$

Clearly, then, a gap in rate exists, as $\frac{2}{k}\log_q\left(\frac{q}{q-1}\right) > \frac{(q-1)}{q^{k+2}}\log_q(e) + o(1)$ for all $k \geqslant 2$. Note, however, that this upper bound is not necessarily tight, as it does not account for the combined error mode.

VI. Conclusion

We have studied the combination of a single substitution error with an unlimited number of tandem-duplication errors, with a fixed duplication-window length. We focused on two noise models, where the substitution error is either restricted to occur in an inserted copy during one of the duplication events, or may occur at any position in the string. We have presented bounds and a construction of error-detecting codes in the former error-model, as well as constructions of error-detecting and error-correcting codes in the latter.

In all cases, a rate loss is observed due to the need to recover from an unlimited number of duplications. Thus, we are interested in the *extra redundancy cost* due to single-error detection or correction. In the first case, of detecting a single restricted substitution, we show that the additional required cost in redundancy is bounded from above by $\log_q(4(n-k))$ using a GV argument in Theorem 11, where Construction B also shows that it is bounded from above by $\log_q(2(k+1)^2)$; depending on the asymptotic regime of k, either may be tighter than the other. In Construction C we find a constructive procedure for generating codes for that purpose, which incur a higher redundancy cost of $4k\log_q(2)$; if k is fixed, which is a likely scenario, then that cost is nonetheless constant as well, and improves upon Theorem 11.

Further, in the second case of unrestricted substitution noise, Construction D provides error-detecting codes for a single substitution incurring an extra redundancy cost of $O(\log(k^2 n))$. Finally, in the same error model, Theorem 26 and Theorem 28 provide error-correcting codes which have lower rates than codes designed solely to correct duplication errors. Although

we did not develop lower bounds on the required redundancy, it is our conjecture that both solutions offered here are sub-optimal. In particular, these latter codes rely on a constrained-coding approach which we do not believe is necessary in this context. We also note that while both the upper bound and lower bound on the rates of these codes approach 1 as $k \to \infty$, the lower bound does so as $\Theta(k^{-1})$ whereas the upper bound is much faster as $\Theta(q^{-k})$, implying a gap yet to be resolved.

For future research, we would like to suggest a few generalizations of the noise model considered herein. First, we suggest studying codes capable of handling a higher number of substitution errors. We also believe codes designed for handling only a bounded number of duplication events are of interest. Finally, we suggest to observe combinations of different noise mechanisms, including bounded tandem-duplication, end- or interspersed-duplication noise [1], or duplication and deletion noise.

REFERENCES

[1] F. Farnoud, M. Schwartz, and J. Bruck, "The capacity of string-duplication systems," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 811–824, 2016.

[2] ——, "Estimation of duplication history under a stochastic model for tandem repeats," *BMC Bioinformatics*, vol. 20, no. 1, 2019.

[3] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Duplication-correcting codes for data storage in the DNA of living organisms," *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 4996–5010, Aug. 2017.

[4] M. Kovačević and V. Y. F. Tan, "Asymptotically optimal codes correcting fixed-length duplication errors in DNA storage systems," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2194–2197, Nov. 2018.

[5] A. Lenz, A. Wachter-Zeh, and E. Yaakobi, "Duplication-correcting codes," *Designs, Codes and Cryptography*, vol. 87, no. 2, pp. 277–298, Mar. 2019.

[6] H. Mahdavifar and A. Vardy, "Asymptotically optimal sticky-insertion-correcting codes with efficient encoding and decoding," in *2017 IEEE International Symposium on Information Theory (ISIT)*, Jun. 2017, pp. 2683–2687.

[7] D. Pumpernik, B. Oblak, and B. Borštnik, "Replication slippage versus point mutation rates in short tandem repeats of the human genome," *Molecular Genetics and Genomics*, vol. 279, no. 1, pp. 53–61, 2008.

[8] F. Sala, R. Gabrys, C. Schoeny, and L. Dolecek, "Exact reconstruction from insertions in synchronization codes," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 2428–2445, 2017.

[9] S. L. Shipman, J. Nivala, J. D. Macklis, and G. M. Church, "Molecular recordings by directed CRISPR spacer acquisition," *Science*, Jun. 2016.

[10] ——, "CRISPR–Cas encoding of a digital movie into the genomes of a population of living bacteria," *Nature*, vol. 547, no. 7663, pp. 345–349, Jul. 2017.

[11] L. M. G. M. Tolhuizen, "The generalized Gilbert-Varshamov bound is implied by Turán's theorem," *IEEE Transactions on Information Theory*, vol. 43, no. 5, pp. 1605–1606, Sep. 1997.

[12] K. Zhou, A. Aertsen, and C. W. Michiels, "The role of variable DNA tandem repeats in bacterial adaptation," *FEMS Microbiology Reviews*, vol. 38, no. 1, pp. 119–141, Jan. 2014.

APPENDIX

Proof of Lemma 21: Let $x' \in D^{*,\leqslant 1}(x)$, where

$$|\mathrm{drt}(x')| = |\mathrm{drt}(x)|, \qquad \text{but} \qquad \mathrm{drt}(x') \neq \mathrm{drt}(x),$$

namely, an ambiguous unrestricted substitution occurred. Let us denote

$$y \triangleq \hat{\phi}(x), \qquad z \triangleq \bar{\phi}(x),$$
$$y' \triangleq \hat{\phi}(x'), \qquad z' \triangleq \bar{\phi}(x').$$

Since duplications do not change the root, we assume without loss of generality that no duplications occur and only a single substitution occurs. Thus, we can write

$$x' = x + a \cdot e_i, \qquad yz = y'z' + a \cdot \epsilon_i,$$

where i denotes the location of the substitution, and $a \in \Sigma \setminus \{0\}$. Depending on i, a single substitution may result in one or two changed positions in the transform doamin of ϕ. The proof of the claim comprises of many cases, and we start with some simple ones.

In the first simple case, the substitution occurs in the first k positions, namely, $1 \leqslant i \leqslant k$. Since $\phi(\mathrm{drt}(x')) = y'\mu(z')$, and $y \neq y'$, if we have $|\mathrm{drt}(x')| = |\mathrm{drt}(x)|$ then

$$d(\phi(\mathrm{drt}(x)), \phi(\mathrm{drt}(x'))) \leqslant 2,$$

by virtue of positions i and $i+k$.

In a similar fashion, if the substitution occurs in the last k positions, namely, $|x|-k+1 \leqslant i \leqslant |x|$, only a single position is changed in the transform ϕ. Since $\phi(\mathrm{drt}(x')) = y'\mu(z')$, and $z \neq z'$, if we have $|\mathrm{drt}(x')| = |\mathrm{drt}(x)|$ then

$$d(\phi(\mathrm{drt}(x)), \phi(\mathrm{drt}(x'))) \leqslant 1,$$

by virtue of positions i.

We are now left with the last interesting case, in which the substitution changes two positions, i and $i+k$, both in the z part of the ϕ-transform. We therefore disregard the part $y = y'$. We may now write

$$\begin{aligned} z &= u \quad a_1 \quad v \quad a_2 \quad w \\ z' &= u \ (a_1 + a) \ v \ (a_2 - a) \ w \end{aligned}$$

where $u, w \in \Sigma^*$, $v \in \Sigma^{k-1}$, $a, a_1, a_2 \in \Sigma$, and $a \neq 0$. We distinguish between two major cases, depending on whether $v = 0^{k-1}$.

Case I: In the first major case we have $v = 0^{k-1}$. Let us write

$$u = u'0^{m_1}, \qquad w = 0^{m_4}w',$$

where all the indicated runs of zeros are maximal. Thus,

$$\begin{aligned} z &= u' \ 0^{m_1} \quad a_1 \quad 0^{k-1} \quad a_2 \quad 0^{m_4} \ w' \\ z' &= u' \ 0^{m_1} \ (a_1 + a) \ 0^{k-1} \ (a_2 - a) \ 0^{m_4} \ w'. \end{aligned}$$

The length of the substring between u' and w' is $m_1 + m_4 + k + 1$ and we note that

$$\left\lfloor \frac{m_1 + m_4 + k + 1}{k} \right\rfloor = \left\lfloor \frac{m_1}{k} \right\rfloor + \left\lfloor \frac{m_4}{k} \right\rfloor + s,$$

where $s \in \{1, 2\}$. We distinguish between the following cases:

1) If $a_1 \neq 0$ and $a_2 \neq 0$:
   a) If $a_1 + a \neq 0$ and $a_2 - a \neq 0$
   $$d(\phi(\mathrm{drt}(x)), \phi(\mathrm{drt}(x'))) \leqslant 2.$$
   b) If exactly one of $a_1 + a$ and $a_2 - a$ is zero, the length of $\mu(z')$ decreases by k.
   c) If $a_1 + a = a_2 - a = 0$, the length of $\mu(z')$ decreases by sk.
2) If $a_1 \neq 0$ and $a_2 = 0$:

a) If $a_1 + a \neq 0$, since $a_2 - a \neq 0$ the length of $\mu(z')$ increases by k.
b) If $a_1 + a = 0$, since $a_2 - a \neq 0$

$$d(\phi(\mathrm{drt}(x)), \phi(\mathrm{drt}(x'))) \leqslant 1.$$

3) If $a_1 = 0$ and $a_2 \neq 0$:
a) If $a_2 - a \neq 0$, since $a_1 + a \neq 0$ the length of $\mu(z')$ increases by k.
b) If $a_2 - a = 0$, since $a_1 + a \neq 0$

$$d(\phi(\mathrm{drt}(x)), \phi(\mathrm{drt}(x'))) \leqslant 1.$$

4) If $a_1 = a_2 = 0$, the length of $\mu(z')$ increases by sk.

Case II: In the second major case, assume $v \neq 0^{k-1}$. Let us write

$$u = u'0^{m_1}, \qquad v = 0^{m_2}v'0^{m_3}, \qquad w = 0^{m_4}w',$$

where all the indicated runs of zeros are maximal. Let $c \in \Sigma \setminus \{0\}$ be some nonzero letter in v', important to us only for the purpose of being able to refer to the part of the string left of c and the part of the string to the right of c.

1) Examining the part of the string to the left of c:
a) If $a_1 \neq 0$:
i) If $a_1 = -a$:
A) If $\left\lfloor \frac{m_1+m_2+1}{k} \right\rfloor > \left\lfloor \frac{m_1}{k} \right\rfloor$, the length before c decreases by k and the substring $0^{j-1}(-a)0^{k-j}$ is deleted.
B) If $\left\lfloor \frac{m_1+m_2+1}{k} \right\rfloor = \left\lfloor \frac{m_1}{k} \right\rfloor$, the length before c stays the same and the substitution $a_1 \to 0$ occurs.
ii) If $a_1 \neq -a$, the length before c stays the same and the substitution $a_1 \to (a_1 + a)$ occurs.
b) If $a_1 = 0$, then $a_1 \neq -a$, and:
i) If $\left\lfloor \frac{m_1+m_2+1}{k} \right\rfloor > \left\lfloor \frac{m_1}{k} \right\rfloor$, the length before c increases by k and $0^{j-1}a0^{k-j}$ is inserted.
ii) If $\left\lfloor \frac{m_1+m_2+1}{k} \right\rfloor = \left\lfloor \frac{m_1}{k} \right\rfloor$, the length before c stays the same and the substitution $0 \to a$ occurs.

2) Examining the part of the string to the right of c:
a) If $a_2 \neq 0$:
i) If $a_2 = a$:
A) If $\left\lfloor \frac{m_3+m_4+1}{k} \right\rfloor > \left\lfloor \frac{m_4}{k} \right\rfloor$, the length after c decreases by k and $0^{t-1}a0^{k-t}$ is deleted.
B) If $\left\lfloor \frac{m_3+m_4+1}{k} \right\rfloor = \left\lfloor \frac{m_4}{k} \right\rfloor$, the length after c stays the same and the substitution $a_2 \to 0$ occurs.
ii) If $a_2 \neq a$, the length after c stays the same and the substitution $a_2 \to (a_2 - a)$ occurs.
b) If $a_2 = 0$, then $a_2 \neq a$, and:
i) If $\left\lfloor \frac{m_3+m_4+1}{k} \right\rfloor > \left\lfloor \frac{m_4}{k} \right\rfloor$, the length after c increases by k and $0^{t-1}(-a)0^{k-t}$ is inserted.
ii) If $\left\lfloor \frac{m_3+m_4+1}{k} \right\rfloor = \left\lfloor \frac{m_4}{k} \right\rfloor$, the length after c stays the same and the substitution $0 \to (-a)$ occurs.

Based on the changes of a_1 and a_2, there are two types of ambiguous unrestricted substitutions:

- Define the sets of cases $A \triangleq \{1(a)iB, 1(a)ii, 1(b)ii\}$ and $B \triangleq \{2(a)iB, 2(a)ii, 2(b)ii\}$. Any substitution scenario from $A \times B$ results in only two changed symbols, hence

$$d(\phi(\mathrm{drt}(x)), \phi(\mathrm{drt}(x'))) \leqslant 2.$$

- The scenarios $(1(a)iA,2(b)i)$ and $(1(b)i,2(a)iA)$ are more complex because they involve both an inserted a substring and a deleted substring of length k. Since the two cases are similar, we only show the analysis of the first case $(1(a)iA,2(b)i)$. We therefore have

$$\begin{aligned} z &= u'\ 0^{m_1}\ a\ 0^{m_2}\ v'\ 0^{m_3}\ 0\ 0^{m_4}\ w' \\ z' &= u'\ 0^{m_1}\ 0\ 0^{m_2}\ v'\ 0^{m_3}\ a\ 0^{m_4}\ w' \end{aligned}$$

where we recall that $a \neq 0$, $|v'| \leqslant k-1$, and v' starts and ends with a non-zero letter. Looking at $\mu(z')$ compared with $\mu(z)$, the part to the left of v' becomes shorter by k letters, whereas the part to the right of it becomes longer by k letters. In particular, we can write

$$\begin{aligned} \mu(z) &= u''\ 0^{|v'|}\ 0^{m_3}a0^{m_2}\ \ v'\ \ w'' \\ \mu(z') &= u''\ \ v'\ \ 0^{m_3}a0^{m_2}\ 0^{|v'|}\ w'' \end{aligned} \tag{13}$$

where $m_2 + m_3 + |v'| + 1 = k$.

Having considered all cases, this last case is the only one in which we have an ambiguous unrestricted substitution in which potentially $d(\phi(\mathrm{drt}(x)), \phi(\mathrm{drt}(x'))) \geqslant 3$. The swapping described in (13) completes the proof of the claim. ■

Yuanyuan Tang is a Ph.D. candidate in the Department of Electrical and Computer Engineering at the University of Virginia. His research interests consist of information theory, coding theory, and wireless communications.

He received the Bachelor's degree in Engineering from the Department of Communication Engineering at Chongqing University in 2015 and the Master's degree in Engineering from the Department of Electronic Engineering at Tsinghua University in 2018.

Yonatan Yehezkeally (S'12) is a graduate student at the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel. His research interests include coding for DNA storage, combinatorial structures, algebraic coding and finite group theory.

Yonatan received the B.Sc. (*cum laude*) degree in Mathematics in 2013, and the M.Sc. (*summa cum laude*) degree in Electrical and Computer Engineering in 2017, all from Ben-Gurion University of the Negev.

Moshe Schwartz (M'03–SM'10) is a professor at the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel. His research interests include algebraic coding, combinatorial structures, and digital sequences.

Prof. Schwartz received the B.A. (*summa cum laude*), M.Sc., and Ph.D. degrees from the Technion – Israel Institute of Technology, Haifa, Israel, in 1997, 1998, and 2004 respectively, all from the Computer Science Department. He was a Fulbright post-doctoral researcher in the Department of Electrical and Computer Engineering, University of California San Diego, and a post-doctoral researcher in the Department of Electrical Engineering, California Institute of Technology. While on sabbatical 2012–2014, he was a visiting scientist at the Massachusetts Institute of Technology (MIT).

Prof. Schwartz received the 2009 IEEE Communications Society Best Paper Award in Signal Processing and Coding for Data Storage, and the 2020 NVMW Persistant Impact Prize. He has also been serving as an Associate Editor for Coding Techniques for the IEEE Transactions on Information Theory since 2014.

Farzad Farnoud (Hassanzadeh) (M'13) is an Assistant Professor in the Department of Electrical and Computer Engineering and the Department of Computer Science at the University of Virginia. Previously, he was a postdoctoral scholar at the California Institute of Technology.

He received his MS degree in Electrical and Computer Engineering from the University of Toronto in 2008. From the University of Illinois at Urbana-Champaign, he received his MS degree in mathematics and his Ph.D. in Electrical and Computer Engineering in 2012 and 2013, respectively. His current research interests include coding for data storage, data deduplication, and probabilistic modeling of genomic data for computational biology and data compression. He is the recipient of the 2013 Robert T. Chien Memorial Award from the University of Illinois for demonstrating excellence in research in electrical engineering and the recipient of the 2014 IEEE Data Storage Best Student Paper Award.