# Duplication-Correcting Codes for Data Storage in the DNA of Living Organisms

Siddharth Jain, *Student Member, IEEE*, Farzad Farnoud (Hassanzadeh), *Member, IEEE*, Moshe Schwartz, *Senior Member, IEEE*, and Jehoshua Bruck, *Fellow, IEEE*

***Abstract*— The ability to store data in the DNA of a living organism has applications in a variety of areas including synthetic biology and watermarking of patented genetically modified organisms. Data stored in this medium are subject to errors arising from various mutations, such as point mutations, indels, and tandem duplication, which need to be corrected to maintain data integrity. In this paper, we provide error-correcting codes for errors caused by tandem duplications, which create a copy of a block of the sequence and insert it in a tandem manner, i.e., next to the original. In particular, we present two families of codes for correcting errors due to tandem duplications of a fixed length: the first family can correct any number of errors, while the second corrects a bounded number of errors. We also study codes for correcting tandem duplications of length up to a given constant $k$, where we are primarily focused on the cases of $k = 2, 3$. Finally, we provide a full classification of the sets of lengths allowed in tandem duplication that result in a unique root for all sequences.**

***Index Terms*— Error-correcting codes, DNA, string-duplication systems, tandem-duplication errors.**

## I. Introduction

DATA storage in the DNA of living organisms (henceforth *live DNA*) has a multitude of applications. It can enable in-vivo synthetic-biology methods and algorithms that need "memory," e.g., to store information about their state or record changes in the environment. Embedding data in live DNA also allows watermarking genetically-modified organisms (GMOs) to verify authenticity and to track unauthorized use [1], [9], [19], as well as labeling organisms in biological studies [24]. DNA watermarking can also be used to tag infectious agents used in research laboratories to identify sources of potential malicious use or accidental release [13]. Furthermore, live DNA can serve as a protected medium for storing large amounts of data in a compact format for long periods of time [2], [24]. An additional advantage of using DNA as a medium is that data can be disguised as part of the organisms' original DNA, thus providing a layer of secrecy [3].

Manuscript received June 1, 2016; revised December 9, 2016; accepted March 13, 2017. Date of publication March 28, 2017; date of current version July 12, 2017. This work was supported by the NSF Expeditions in Computing Program (The Molecular Programming Project). This paper was presented at the 2016 IEEE International Symposium on Information Theory. *(Corresponding Author: Siddharth Jain.)*

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

F. Farnoud was with the Electrical Engineering Department, California Institute of Technology, Pasadena, CA 91125 USA. He is now with the Department of Electrical and Computer Engineering and Department of Computer Science, University of Virginia, Charlottesville, VA, 22903 USA (e-mail: farzad@virginia.edu).

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

Communicated by C.-C. Wang, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2017.2688361

While the host organism provides a level of protection to the data-carrying DNA molecules as well as a method for replication, the integrity of the stored information suffers from mutations such as tandem duplications, point mutations, insertions, and deletions. Furthermore, since each DNA replication may introduce new mutations, the number of such deleterious events increases with the number of generations. As a result, to ensure decodability of the stored information, the coding/decoding scheme must be capable of a level of error correction. Motivated by this problem, we study designing codes that can correct errors arising from tandem duplications. In addition to improving the reliability of data storage in live DNA, studying such codes may help to acquire a better understanding of how DNA stores and protects biological information in nature.

Tandem duplication is the process of inserting a copy of a segment of the DNA adjacent to its original position, resulting in a *tandem repeat*. A process that may lead to a tandem duplication is *slipped-strand mispairings* [20] during DNA replication, where one strand in a DNA duplex is displaced and misaligned with the other. Tandem repeats constitute about 3% of the human genome [15] and may cause important phenomena such as chromosome fragility, expansion diseases, silencing genes [22], and rapid morphological variation [7].

Different approaches to the problem of error-control for data stored in live DNA have been proposed in the literature. In the work of Arita and Ohashi [1], each group of five bits of information is followed by one parity bit for error detection. Heider and Barnekow [9] use the extended [8, 4, 4] binary Hamming code or repetition coding to protect the data. Yachie *et al.* [25] propose to enhance reliability by inserting multiple copies of the data into multiple regions of the genome of the host organism. Finally, Haughton and Balado [8] present an encoding method satisfying certain biological constraints, which is studied in a substitution-mutation model. None of the aforementioned encodings, with the possible exception of repetition coding, are designed to combat tandem duplications, which is the focus of this paper. While repetition coding can correct duplication errors, it is not an efficient method because of its high redundancy.

It should also be noted that error control for storage in live DNA is inherently different from that in DNA that is stored outside of a living organism (see [26] for an overview), since the latter is not concerned with errors arising during organic DNA replication.

In this work, we ignore the potential biological effects of embedding data into the DNA. Furthermore, constructing codes that, in addition to tandem-duplication errors, can combat other types of errors, such as substitutions, are postponed to a future work.

We also note that tandem duplication, as well as other duplication mechanisms, were studied in the context of information theory [5], [6], [12]. However, these works used duplications as a generative process, and attempted to measure its capacity and diversity. In contrast, we consider duplications as a noise source, and design error-correcting codes to combat it.

We will first consider the tandem-duplication channel with duplications of a fixed length $k$. For example with $k = 3$, after a tandem duplication, the sequence $ACAGT$ may become $ACAG\underline{CAG}T$, which may then become $ACA\underline{ACA}GCAGT$ where the copy is underlined. In our analysis, we provide a mapping in which tandem duplications of length $k$ are equivalent to insertion of $k$ zeros. Using this mapping, we demonstrate the strong connection between codes that correct duplications of a fixed length and Run-Length Limited (RLL) systems. We present constructions for codes that can correct an unbounded number of tandem duplications of a fixed length and show that our construction is optimal, i.e., of the largest size. A similar idea was used in [4], where codes were constructed for duplication-error correction with the number of tandem duplications restricted to a given size $r$ and a duplication length of 1 only. In this paper, we generalize their result by constructing optimal (i.e., maximum size) error-correcting codes for arbitrary duplication length $k$ and with no restriction on the number of tandem duplications.

We then turn our attention to codes that correct $t$ tandem duplications (as opposed to an unbounded number of duplications), and show that these codes are closely related to constant-weight codes in the $\ell_1$ metric.

We also consider codes for correcting duplications of bounded length. Here, our focus will be on duplication errors of length at most 2 or 3, for which we will present a construction that corrects any number of such errors. In the case of duplication length at most 2 the codes we present are optimal.

Finally, when a sequence has been corrupted by a tandem-duplication channel, the challenge arises in finding the *root* sequences from which the corrupted sequence could be generated. A root sequence does not contain any tandem-duplicated subsequences. For example, for the sequence $ACGT\underline{GT}$, with $GTGT$ as a tandem-duplication error, a root sequence would be $ACGT$ since $ACGTGT$ can be generated from $ACGT$ via a tandem duplication of length 2 on $GT$. But there can be sequences that have more than one root. For example, the sequence $ACGCACGCG$ can be generated from $ACG$ through a tandem duplication of $CG$ first, followed by a tandem duplication of $ACGC$. Alternatively, it can also be generated from $ACGCACG$ by doing a tandem duplication of the suffix $CG$. Hence, $ACGCACGCG$ has two roots. However, if we restrict the length of duplication to 2 in the previous example, then $ACGCACGCG$ has only one root i.e., $ACGCACG$. This means that the number of roots that a sequence can have depends on the set of duplication lengths that are allowed, and the size of the alphabet. We provide in Section V a complete classification of the parameters required for the unique-root property. This unique-root property of the fixed length, 2-bounded and 3-bounded tandem-duplication channels allows us to construct error-correcting codes for them.

The paper is organized as follows. The preliminaries and notation are described in Section II. In Sections III and IV we present the results concerning duplications of a fixed length $k$ and duplications of length at most $k$, respectively. In Section V, we fully characterize tandem-duplication channels which have a unique root. We conclude with some open questions in Section VI.

## II. Preliminaries

We let $\Sigma$ denote some finite alphabet, and $\Sigma^*$ denote the set of all finite strings (words) over $\Sigma$. The unique empty word is denoted by $\epsilon$. The set of finite non-empty words is denoted by $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. Given two words $x, y \in \Sigma^*$, their concatenation is denoted by $xy$, and $x^t$ denotes the concatenation of $t$ copies of $x$, where $t$ is some positive integer. By convention, $x^0 = \epsilon$. The length of a string $x \in \Sigma^*$ is denoted by $|x|$. We normally index the letters of a word starting with 1, i.e., $x = x_1 x_2 \dots x_n$, with $x_i \in \Sigma$. With this notation, the $t$-prefix and $t$-suffix of $x$ are defined by

$$\begin{aligned} \mathrm{Pref}_t(x) &= x_1 x_2 \dots x_t, \\ \mathrm{Suff}_t(x) &= x_{n-t+1} x_{n-t+2} \dots x_n. \end{aligned}$$

Given a string $x \in \Sigma^*$, a *tandem duplication of length* $k$ is a process by which a contiguous substring of $x$ of length $k$ is copied next to itself. More precisely, we define the tandem-duplication rules, $T_{i,k} : \Sigma^* \to \Sigma^*$, as

$$T_{i,k}(x) = \begin{cases} uvvw & \text{if } x = uvw,\ |u| = i,\ |v| = k \\ x & \text{otherwise.} \end{cases}$$

We note that the "otherwise" case describes a degenerate case when $|x| < k + i$, and therefore $x$ cannot be decomposed into a prefix $u$ of length $i$, an inner part $v$ of length $k$, and some suffix $w$. Two specific sets of duplication rules are of interest to us throughout the paper.

$$\begin{aligned} \mathcal{T}_k &= \left\{ T_{i,k} \;\middle|\; i \geq 0 \right\}, \\ \mathcal{T}_{\leq k} &= \left\{ T_{i,k'} \;\middle|\; i \geq 0, 1 \leq k' \leq k \right\}. \end{aligned}$$

Given $x, y \in \Sigma^*$, if there exist $i$ and $k$ such that

$$y = T_{i,k}(x),$$

non-degenerately,[1]then we say $y$ is a direct descendant of $x$, and denote it by

$$x \underset{k}{\Longrightarrow} y.$$

If a sequence of $t$ non-degenerate tandem duplications of length $k$ is employed to reach $y$ from $x$ we say $y$ is

[1]Here, and throughout the paper, non-degenerately refers to tandem duplications that avoid the "otherwise" case in the definition of $T_{i,k}$.

a $t$-descendant of $x$ and denote it by

$$x \overset{t}{\underset{k}{\Longrightarrow}} y.$$

More precisely, we require the existence of $t$ non-negative integers $i_1, i_2, \ldots, i_t$, with $0 \le i_j \le |x| + k(j-2)$, such that

$$y = T_{i_t,k}(T_{i_{t-1},k}(\ldots T_{i_1,k}(x)\ldots)).$$

Finally, if there exists a finite sequence of tandem duplications of length $k$ transforming $x$ into $y$, we say $y$ is a descendant of $x$ and denote it by

$$x \overset{*}{\underset{k}{\Longrightarrow}} y.$$

We note that $x$ is its own descendant via an empty sequence of tandem duplications.

*Example 1: Let* $\Sigma = \{0,1,2,3\}$ *and* $x = 02123$. *Since,* $T_{1,2}(x) = 0212123$ *and* $T_{0,2}(0212123) = 020212123$*, the following hold*

$$02123 \underset{2}{\Longrightarrow} 0212123, 02123 \overset{2}{\underset{2}{\Longrightarrow}} 020212123,$$

*where in both expressions, the relation could be replaced with* $\overset{*}{\underset{2}{\Longrightarrow}}$.

□

We define the *descendant cone* of $x$ as

$$D_k^*(x) = \left\{ y \in \Sigma^* \;\middle|\; x \overset{*}{\underset{k}{\Longrightarrow}} y \right\}.$$

In a similar fashion we define the *t-descendant cone* $D_k^t(x)$ by replacing $\overset{*}{\underset{k}{\Longrightarrow}}$ with $\overset{t}{\underset{k}{\Longrightarrow}}$ in the definition of $D_k^*(x)$.

The set of definitions given thus far was focused on tandem-duplication rules of substrings of length exactly $k$, i.e., for rules from $\mathcal{T}_k$. These definitions as well as others in this section are extended in the natural way for tandem-duplication rules of length up to $k$, i.e., $\mathcal{T}_{\le k}$. We denote these extensions by replacing the $k$ subscript with the $\le k$ subscript. Thus, we also have $D_{\le k}^*(x)$ and $D_{\le k}^t(x)$.

*Example 2: Consider* $\Sigma = \{0,1\}$ *and* $x = 01$. *It is not difficult to see that*

$$\begin{aligned} D_1^2(x) &= \{0001, 0011, 0111\}, \\ D_1^*(x) &= \left\{ 0^i 1^j \;\middle|\; i, j \in \mathbb{N} \right\}, \\ D_2^*(x) &= \left\{ (01)^i \;\middle|\; i \in \mathbb{N} \right\}, \\ D_{\le 2}^*(x) &= \left\{ 0s1 \mid s \in \Sigma^* \right\}. \end{aligned}$$

□

Using the notation $D_k^*$, we restate the definition of the *tandem string-duplication system* given in [6]. Given a finite alphabet $\Sigma$, a seed string $s \in \Sigma^*$, the tandem string-duplication system is given by

$$S_k = S(\Sigma, s, \mathcal{T}_k) = D_k^*(s),$$

i.e., it is the set of all the descendants of $s$ under tandem duplication of length $k$.

The process of tandem duplication can be naturally reversed. Given a string $y \in \Sigma^*$, for any positive integer, $t > 0$, we define the *t-ancestor cone* as

$$D_k^{-t}(y) = \left\{ x \in \Sigma^* \;\middle|\; x \overset{t}{\underset{k}{\Longrightarrow}} y \right\},$$

or in other words, the set of all words for which $y$ is a $t$-descendant.

Yet another way of viewing the $t$-ancestor cone is by defining the *tandem-deduplication rules*, $T_{i,k}^{-1} : \Sigma^* \to \Sigma^*$, as

$$T_{i,k}^{-1}(y) = \begin{cases} uvw & \text{if } y = uvvw,\ |u| = i,\ |v| = k \\ \epsilon & \text{otherwise,} \end{cases}$$

where we recall $\epsilon$ denotes the empty word. This operation takes an adjacently-repeated substring of length $k$, and removes one of its copies. Thus, a string $x$ is in the $t$-ancestor cone of $y$ (where we assume $x, y \ne \epsilon$ to avoid trivialities) iff there is a sequence of $t$ non-degenerate deduplication operations transforming $y$ into $x$, i.e., there exist $t$ non-negative integers $i_1, i_2, \ldots, i_t$, such that, non-degenerately,

$$x = T_{i_t,k}^{-1}(T_{i_{t-1},k}^{-1}(\ldots T_{i_1,k}^{-1}(y)\ldots)).$$

In a similar fashion we define the *ancestor cone* of $y$ as

$$D_k^{-*}(y) = \left\{ x \in \Sigma^* \;\middle|\; x \overset{*}{\underset{k}{\Longrightarrow}} y \right\}.$$

By flipping the direction of the derivation arrow, we let $\Longleftarrow$ denote deduplication. Thus, if $y$ may be deduplicated to obtain $x$ in a single step, we write

$$y \underset{k}{\Longleftarrow} x.$$

For multiple steps we add $*$ in superscript.

*Example 3: We have*

$$0212123 \underset{2}{\Longleftarrow} 02123, 020212123 \overset{2}{\underset{2}{\Longleftarrow}} 02123,$$

*and*

$$D_2^{-*}(020212123) = \{020212123, 0212123, 0202123, 02123\}.$$

□

A word $y \in \Sigma^*$ is said to be *irreducible* (with respect to duplications of length $k$) if there is nothing to deduplicate in it, i.e., $y$ is its only ancestor, meaning

$$D_k^{-*}(y) = \{y\}.$$

The set of irreducible words is denoted by $\mathrm{Irr}_k$. We will find it useful to denote the set of irreducible words of length $n$ by

$$\mathrm{Irr}_k(n) = \mathrm{Irr}_k \cap \Sigma^n.$$

The ancestors of $y \in \Sigma^*$ that cannot be further deduplicated, are called the *roots* of $y$, and are denoted by

$$R_k(y) = D_k^{-*}(y) \cap \mathrm{Irr}_k.$$

Note that since the aforementioned definitions extend to tandem-duplication rules of length up to $k$, we also have $S_{\le k}$, $D_{\le k}^{-t}(y)$, $D_{\le k}^{-*}(y)$, $\mathrm{Irr}_{\le k}$, $\mathrm{Irr}_{\le k}(n)$, and $R_{\le k}(y)$. In some previous works (e.g., [17]), $S_k$ is called the *uniform-bounded-duplication system*, whereas $S_{\leq k}$ is called the *bounded-duplication system.*

*Example 4: For the binary alphabet* $\Sigma = \{0, 1\}$,

$$\mathrm{Irr}_{\leq 2} = \{0, 1, 01, 10, 010, 101\},$$

*and for any alphabet that contains* $\{0, 1, 2, 3\}$,

$$R_2(020212123) = \{02123\},$$
$$R_{\leq 4}(012101212) = \{012, 0121012\}.$$

□

Inspired by the DNA-storage scenario, we now define error-correcting codes for tandem string-duplication systems.

*Definition 5: An* $(n, M; t)_k$ *code* $C$ *for the* $k$*-tandem-duplication channel is a subset* $C \subseteq \Sigma^n$ *of size* $|C| = M$*, such that for each* $x, y \in C$, $x \neq y$,

$$D_k^t(x) \cap D_k^t(y) = \emptyset.$$

*Here* $t$ *stands for either a non-negative integer, or* $*$*. In the former case we say the code can correct* $t$ *errors, whereas in the latter case we say the code can correct all errors. In a similar fashion, we define an* $(n, M; t)_{\leq k}$ *by replacing all "*$k$*" subscripts by "*$\leq k$*".*

Assume the size of the finite alphabet is $|\Sigma| = q$. We then denote the size of the largest $(n, M; t)_k$ code over $\Sigma$ by $A_q(n; t)_k$. The capacity of the channel is then defined as

$$\mathsf{cap}_q(t)_k = \limsup_{n\to\infty} \frac{1}{n} \log_q A_q(n; t)_k.$$

Analogous definitions are obtained by replacing $k$ with $\leq k$ or by replacing $t$ with $*$.

In certain places we shall point out connections between string-duplication systems and formal languages. These connections have not gone unnoticed, and appear in papers such as [17], which we describe in the appropriate context. A *language*, $L$, is nothing but a set of words, $L \subseteq \Sigma^*$, where $\Sigma$ is some finite alphabet. A type of language we shall encounter frequently is a *regular language*, which is exactly a set of words that may be recognized by a finite automaton. Intuitively, such an automaton is defined by a finite set of states, and a finite set of transitions between states, each transition labeled by a symbol from $\Sigma$. Additionally, a single state is assigned the role of a starting state, and a subset of the states is assigned the role of accepting states. A word is recognized, if it is the result of concatenating the symbols of transitions describing a path from the starting state to some accepting state. Regular languages may also be described by regular expressions. The interested reader is referred to [10].

## III. $k$-Tandem-Duplication Codes

In this section we consider tandem string-duplication systems where the substring being duplicated is of a constant length $k$. Such systems were studied in the context of formal languages [17] (also called *uniform-bounded-duplication systems*), and also in the context of coding and information theory [6].

In [17] it was shown that for any finite alphabet $\Sigma$ and any word $x \in \Sigma^*$, under $k$-tandem duplication, $x$ has a unique root, i.e.,

$$|R_k(x)| = 1.$$

Additionally, finding the unique root may be done efficiently, even by a greedy algorithm which searches for occurrences of $ww$ as substrings of $x$, with $|w| = k$, removing one copy of $w$, and repeating the process. This was later extended in [16], where it was shown that the roots of a regular language also form a regular language. In what follows we give an alternative elementary proof to the uniqueness of the root. This proof will enable us to easily construct codes for $k$-tandem-duplication systems, as well as to state bounds on their parameters. The proof technique may be seen as an extension of the string-derivative technique used in [4], which was applied only for $k = 1$ over a binary alphabet.

The system $S_k$ was also studied in [6] from a coding and information-theoretic perspective. In particular, it was proved in [6] that the capacity of $S_k$ is 0. This fact will turn out to be extremely beneficial when devising error-correcting codes for $k$-tandem-duplication systems.

Throughout this section, without loss of generality, we assume $\Sigma = \mathbb{Z}_q$. We also use $\mathbb{Z}_q^*$ to denote the set of all finite strings of $\mathbb{Z}_q$ (not to be confused with the non-zero elements of $\mathbb{Z}_q$), and $\mathbb{Z}_q^{\geq k}$ to denote the set of all finite strings over $\mathbb{Z}_q$ of length $k$ or more.

We shall require the following mapping, $\phi_k : \mathbb{Z}_q^{\geq k} \to \mathbb{Z}_q^k \times \mathbb{Z}_q^*$. The mapping is defined by,

$$\phi_k(x) = (\mathrm{Pref}_k(x), \mathrm{Suff}_{|x|-k}(x) - \mathrm{Pref}_{|x|-k}(x)),$$

where subtraction is performed entry-wise over $\mathbb{Z}_q$. We easily observe that $\phi_k$ is a bijection between $\mathbb{Z}_q^n$ and $\mathbb{Z}_q^k \times \mathbb{Z}_q^{n-k}$ by noting that we can recover $x$ from $\phi_k(x)$ in the following manner: first set $x_i = \phi_k(x)_i$, for all $1 \leq i \leq k$, and for $i = k+1, k+2, \ldots$, set $x_i = x_{i-k} + \phi_k(x)_i$, where $\phi_k(x)_i$ denotes the $i$th symbol of $\phi_k(x)$. Thus, $\phi_k^{-1}$ is well defined.

Another mapping we define is one that injects $k$ consecutive zeros into a string. More precisely, we define $\zeta_{i,k} : \mathbb{Z}_q^k \times \mathbb{Z}_q^* \to \mathbb{Z}_q^k \times \mathbb{Z}_q^*$, where

$$\zeta_{i,k}(x, y) = \begin{cases} (x, u0^k w) & \text{if } y = uw,\ |u| = i \\ (x, y) & \text{otherwise.} \end{cases}$$

The following lemma will form the basis for the proofs to follow.

*Lemma 6: The following diagram commutes:*

$$\begin{tikzcd} \mathbb{Z}_q^{\geq k} \arrow[r, "T_{i,k}"] \arrow[d, "\phi_k"] & \mathbb{Z}_q^{\geq k} \arrow[d, "\phi_k"] \\ \mathbb{Z}_q^k \times \mathbb{Z}_q^* \arrow[r, "\zeta_{i,k}"] & \mathbb{Z}_q^k \times \mathbb{Z}_q^* \end{tikzcd}$$

*i.e., for every string* $x \in \mathbb{Z}_q^{\geq k}$,

$$\phi_k(T_{i,k}(x)) = \zeta_{i,k}(\phi_k(x)).$$

Before presenting the proof, we provide an example for the diagram of the lemma.

*Example 7: Assume $\Sigma = \mathbb{Z}_4$. Starting with* 02123 *and letting $i = 1$ and $k = 2$ leads to*

$$\begin{tikzcd} 02123 \arrow[r, "T_{1,2}"] \arrow[d, "\phi_2"] & 0212\underline{21}23 \arrow[d, "\phi_2"] \\ (02, 102) \arrow[r, "\zeta_{1,2}"] & (02, 1\underline{00}02) \end{tikzcd}$$

*where the inserted elements are underlined.* □

*Proof:* Let $x \in \mathbb{Z}_q^{\geq k}$ be some string, $x = x_1\ x_2 \dots x_n$. Additionally, let $\phi_k(x) = (y, z)$ with $y = y_1 \dots y_k$, and $z = z_1 \dots z_{n-k}$. We first consider the degenerate case, where $i \geq n-k+1$. In that case, $T_{i,k}(x) = x$, and then by definition $\zeta_{i,k}(y, z) = (y, z)$ since $z$ does not have a prefix of length at least $n-k+1$. Thus, for $i \geq n-k+1$ we indeed have

$$\phi_k(T_{i,k}(x)) = \phi_k(x) = (y, z) = \zeta_{i,k}(y, z) = \zeta_{i,k}(\phi_k(x)).$$

We are left with the case of $0 \leq i \leq n-k$. We now write

$$T_{i,k}(x) = x_1\ x_2 \dots x_{i+k} x_{i+1} x_{i+2} \dots x_n.$$

Thus, if we denote $\phi_k(T_{i,k}(x)) = (y, z)$, then

$$\begin{aligned} y &= x_1 \dots x_k = \mathrm{Pref}_k(x), \\ z &= x_{k+1} - x_1, \dots, x_{k+i} - x_i, 0^k, \\ &\qquad x_{k+i+1} - x_{i+1}, \dots, x_n - x_{n-k}. \end{aligned}$$

This is exactly an insertion of $0^k$ after $i$ symbols in the second part of $\phi_k(x)$. It therefore follows that

$$\phi_k(T_{i,k}(x)) = (y, z) = \zeta_{i,k}(\phi_k(x)),$$

as claimed. ■

Recalling that $\phi_k$ is a bijection between $\mathbb{Z}_q^n$ and $\mathbb{Z}_q^k \times \mathbb{Z}_q^{n-k}$, together with Lemma 6 gives us the following corollary.

*Corollary 8: For any $x \in \mathbb{Z}_q^{\geq k}$, and for any sequence of non-negative integers $i_1, \dots, i_t$,*

$$T_{i_t,k}(\dots T_{i_1,k}(x)\dots) = \phi_k^{-1}(\zeta_{i_t,k}(\dots \zeta_{i_1,k}(\phi_k(x))\dots)).$$

*Example 9: Continuing Example* 7 *and using the notation of Corollary* 8*, let $x = 02123$, $k = t = 2$, $i_1 = 1$, and $i_2 = 0$. Then*

$$\begin{aligned} T_{0,2}(T_{1,2}(02123)) &= T_{0,2}(0212123) \\ &= 020212123 \\ &= \phi_k^{-1}((02, 0010002)) \\ &= \phi_k^{-1}(\zeta_{0,2}((02, 10002))) \\ &= \phi_k^{-1}(\zeta_{0,2}(\zeta_{1,2}((02, 102)))) \\ &= \phi_k^{-1}(\zeta_{0,2}(\zeta_{1,2}(\phi_k(02123)))). \end{aligned}$$

□

Corollary 8 paves the way to working in the $\phi_k$-transform domain. In this domain, a tandem-duplication operation of length $k$ translates into an insertion of a block of $k$ consecutive zeros. Conversely, a tandem-deduplication operation of length $k$ becomes a removal of a block of $k$ consecutive zeros.

The uniqueness of the root, proved in [17], now comes for free. In the $\phi_k$-transform domain, given $(x, y) \in \mathbb{Z}_q^k \times \mathbb{Z}_q^*$, as long as $y$ contains a substring of $k$ consecutive zeros, we may perform another deduplication. The process stops at the unique outcome in which the length of every run of zeros in $y$ is reduced modulo $k$.

This last observation motivates us to define the following operation on a string in $\mathbb{Z}_q^*$. We define $\mu_k : \mathbb{Z}_q^* \to \mathbb{Z}_q^*$ which reduces the lengths of runs of zeros modulo $k$ in the following way. Consider a string $x \in \mathbb{Z}_q^*$, where

$$x = 0^{m_0} w_1 0^{m_1} w_2 \dots w_t 0^{m_t},$$

where $m_i$ are non-negative integers, and $w_1, \dots, w_t \in \mathbb{Z}_q \setminus \{0\}$, i.e., $w_1, \dots, w_t$ are single non-zero symbols. We then define

$$\mu_k(x) = 0^{m_0 \bmod k} w_1 0^{m_1 \bmod k} w_2 \dots w_t 0^{m_t \bmod k}.$$

For example, for $z = 0010002$,

$$\mu_2(z) = 102.$$

Additionally, we define

$$\sigma_k(x) = \left( \left\lfloor \frac{m_0}{k} \right\rfloor, \left\lfloor \frac{m_1}{k} \right\rfloor, \dots, \left\lfloor \frac{m_t}{k} \right\rfloor \right) \in \ (\mathbb{N} \cup \{0\})^*$$

and call $\sigma(x)$ the *zero signature* of $x$. For $z$ given above,

$$\sigma_2(z) = (1, 1, 0).$$

We note that $\mu_k(x)$ and $\sigma(x)$ together uniquely determine $x$.

We also observe some simple properties. First, the Hamming weight of a vector, denoted $\mathrm{wt}_H$, counts the number of non-zero elements in a vector. By definition we have for every $x \in \mathbb{Z}_q^n$,

$$\mathrm{wt}_H(x) = \mathrm{wt}_H(\mu_k(x)).$$

Additionally, the length of the vector $\sigma_k(x)$, denoted $|\sigma_k(x)|$, is given by

$$|\sigma_k(x)| = \mathrm{wt}_H(x) + 1 = \mathrm{wt}_H(\mu_k(x)) + 1. \qquad (1)$$

Note that for $z = 0010002$ as above, we have

$$|\sigma_2(z)| = 3 = \mathrm{wt}_H(z) + 1 = \mathrm{wt}_H(102) + 1.$$

Thus, our previous discussion implies the following corollary.

*Corollary 10: For any string $x \in \mathbb{Z}_q^{\geq k}$,*

$$R_k(x) = \left\{ \phi_k^{-1}(y, \mu_k(z)) \ \middle|\ \phi_k(x) = (y, z) \right\}.$$

We recall the definition of the $(0, k-1)$-RLL system over $\mathbb{Z}_q$ (for example, see [11], [18]). It is defined as the set of all finite strings over $\mathbb{Z}_q$ that do not contain $k$ consecutive zeros. We denote this set as $C_{\mathrm{RLL}_q(0,k-1)}$. In our notation,

$$C_{\mathrm{RLL}_q(0,k-1)} = \left\{ x \in \mathbb{Z}_q^* \ \middle|\ \sigma_k(x) \in \ 0^* \right\}.$$

By convention, $C_{\mathrm{RLL}_q(0,k-1)} \cap \mathbb{Z}_q^0 = \{\epsilon\}$. The following is another immediate corollary.

*Corollary 11: For all $n \geq k$,*

$$\mathrm{Irr}_k(n) = \left\{ \phi_k^{-1}(y, z) \ \middle|\ y \in \mathbb{Z}_q^k, z \in C_{\mathrm{RLL}_q(0,k-1)} \cap \mathbb{Z}_q^{n-k} \right\}.$$

*Proof:* The proof is immediate since $x$ is irreducible iff no deduplication action may be applied to it. This happens iff for $\phi_k(x) = (y, z)$, $z$ does not contain $k$ consecutive zeros, i.e., $z \in C_{\mathrm{RLL}_q(0,k-1)} \cap \mathbb{Z}_q^{n-k}$. ■

Given two strings, $x, x' \in \mathbb{Z}_q^{\geq k}$, we say $x$ and $x'$ are $k$-congruent, denoted $x \sim_k x'$, if $R_k(x) = R_k(x')$. It is easily seen that $\sim_k$ is an equivalence relation.

*Corollary 12: Let $x, x' \in \mathbb{Z}_q^*$ be two strings, and denote $\phi_k(x) = (y, z)$ and $\phi_k(x') = (y', z')$. Then $x \sim_k x'$ iff $y = y'$ and $\mu_k(z) = \mu_k(z')$.*

*Proof:* This is immediate when using Corollary 10 to express the roots of $x$ and $x'$. ■

*Example 13: For instance,* 02123, 0212323, 0212123*, and* 020212123 *are all 2-congruent, since they have the unique root* 02123. *In the $\phi_2$-transform domain, for each sequence $x$ in the preceding list, if we let $\phi_2(x) = (y, z)$, then $y = 02$ and $\mu_2(z) = 102$.* □

The following lemma appeared in [17, Proposition 2]. We restate it and give an alternative proof.

*Lemma 14: For all $x, x' \in \mathbb{Z}_q^{\geq k}$, we have*

$$D_k^*(x) \cap D_k^*(x') \neq \emptyset$$

*if and only if $x \sim_k x'$.*

*Proof:* In the first direction, assume $x \not\sim_k x'$. By the uniqueness of the root, let us denote $R_k(x) = \{u\}$ and $R_k(x') = \{u'\}$, with $u \neq u'$. If there exists $w \in D_k^*(x) \cap D_k^*(x')$, then $w$ is a descendant of both $u$ and $u'$, therefore $u, u' \in R_k(w)$, which is a contradiction. Hence, no such $w$ exists, i.e., $D_k^*(x) \cap D_k^*(x') = \emptyset$.

In the other direction, assume $x \sim_k x'$. We construct a word $w \in D_k^*(x) \cap D_k^*(x')$. Denote $\phi_k(x) = (y, z)$ and $\phi_k(x') = (y', z')$. By Corollary 12 we have

$$y = y',$$
$$\mu_k(z) = \mu_k(z').$$

Let us then denote

$$z = 0^{m_0} v_1 0^{m_1} v_2 \dots v_t 0^{m_t},$$
$$z' = 0^{m'_0} v_1 0^{m'_1} v_2 \dots v_t 0^{m'_t},$$

with $v_i$ a non-zero symbol, and

$$m_i \equiv m'_i \pmod{k},$$

for all $i$. We now define

$$z'' = 0^{\max(m_0, m'_0)} v_1 0^{\max(m_1, m'_1)} v_2 \dots v_t 0^{\max(m_t, m'_t)}.$$

Since $z''$ differs from $z$ and $z'$ by insertion of blocks of $k$ consecutive zeros, it follows that

$$w = \phi_k^{-1}(y, z'') \in D_k^*(x) \cap D_k^*(x'),$$

which completes the proof. ■

We now turn to constructing error-correcting codes. The first construction is for a code capable of correcting any number of tandem duplications of length $k$.

*Construction A: Fix $\Sigma = \mathbb{Z}_q$ and $k \geq 1$. For any $n \geq k$ we construct*

$$C = \bigcup_{i=0}^{\lfloor n/k \rfloor - 1} \left\{ \phi_k^{-1}(y, z0^{ki}) \;\middle|\; \phi_k^{-1}(y, z) \in \mathrm{Irr}_k(n - ik) \right\}.$$

*Theorem 15: The code $C$ from Construction A is an $(n, M; *)_k$ code, with*

$$M = \sum_{i=0}^{\lfloor n/k \rfloor - 1} q^k M_{\mathrm{RLL}_q(0,k-1)}(n - (i+1)k).$$

*Here $M_{\mathrm{RLL}_q(0,k-1)}(m)$ denotes the number of strings of length $m$ which are $(0, k-1)$-RLL over $\mathbb{Z}_q$, i.e.,*

$$M_{\mathrm{RLL}_q(0,k-1)}(m) = \left| C_{\mathrm{RLL}_q(0,k-1)} \cap \mathbb{Z}_q^m \right|.$$

*Proof:* The size of the code is immediate, by Corollary 11. Additionally, the roots of distinct codewords are distinct as well, since we constructed the code from irreducible words with blocks of $k$ consecutive zeros appended to their end. Thus, by Lemma 14, the descendant cones of distinct codewords are disjoint. ■

We can say more about the size of the code we constructed.

*Theorem 16: The code $C$ from Construction A is optimal, i.e., it has the largest cardinality of any $(n; *)_k$ code.*

*Proof:* By Lemma 14, any two distinct codewords of an $(n; *)_k$ code must belong to different equivalence classes of $\sim_k$. The code $C$ of Construction A contains exactly one codeword from each equivalence class of $\sim_k$, and thus, it is optimal. ■

The code $C$ from Construction A also allows a simple decoding procedure, whose correctness follows from Corollary 10. Assume a word $x' \in \mathbb{Z}_q^{\geq k}$ is received, and let $\phi_k(x') = (y', z')$. The decoded word is simply

$$\tilde{x} = \phi_k^{-1}(y', \mu_k(z') 0^{n-k-|\mu_k(z')|}), \tag{2}$$

where $n$ is the length of the code $C$. In other words, the decoding procedure recovers the unique root of the received $x'$, and in the $\phi_k$-transform domain, pads it with enough zeros.

*Example 17: Let $n = 4$, $q = 2$, and $k = 1$. By inspection, the code $C$ of Construction A can be shown to equal*

$$C = \left\{ \underline{0}000, \underline{01}11, \underline{010}0, \underline{0101}, \underline{1}111, \underline{10}00, \underline{101}1, \underline{1010} \right\},$$

*where in each codeword the $k$-irreducible part is underlined. As an example of decoding, both* 01100 *and* 01000 *decode to* 0100. *Specifically for the former case, $x' = 01100$, we have $\phi_k(x') = (y', z') = (0, 1010)$. So $\mu_k(z') = 11$ and*

$$\tilde{x} = \phi_k^{-1}(0, 110) = 0100.$$

□

Encoding may be done using any of the many various ways for encoding RLL-constrained systems. The reader is referred to [11] and [18] for further reading. After encoding the RLL-constrained string $z$, a string $y \in \mathbb{Z}_q^k$ is added, and $\phi_k^{-1}$ employed, to obtain a codeword.

Finally, the asymptotic rate of the code family may also be obtained, thus, giving the capacity of the channel.

*Corollary 18: For all $q \geq 2$ and $k \geq 1$,*

$$\mathsf{cap}_q(*)_k = \mathsf{cap}(\mathrm{RLL}_q(0, k-1)),$$

*where $\mathsf{cap}(\mathrm{RLL}_q(0, k-1))$ is the capacity of the $q$-ary $(0, k-1)$-RLL constrained system.*

*Proof:* We use $C_n$ to denote the code from Construction A, where the subscript $n$ is used to denote the length of the code. It is easy to see that for $n \geq k$,

$$q^k M_{\mathrm{RLL}_q(0,k-1)}(n-k) \leq |C_n| \leq nq^k M_{\mathrm{RLL}_q(0,k-1)}(n-k).$$

Then by standard techniques [18] for constrained coding,

$$\lim_{n\to\infty} \frac{1}{n} \log_2 |C_n| = \mathsf{cap}(\mathrm{RLL}_q(0, k-1)).$$

■

It is well known (see e.g. [18]) that

$$\mathsf{cap}(\mathrm{RLL}_q(0, k-1)) = \log_2 \lambda(A_q(k-1)),$$

where $\lambda(A_q(k-1))$ is the largest eigenvalue of the $k \times k$ matrix $A_q(k-1)$ defined as

$$A_q(k-1) = \begin{pmatrix} q-1 & 1 & & & \\ q-1 & & 1 & & \\ \vdots & & & \ddots & \\ q-1 & & & & 1 \\ q-1 & & & & \end{pmatrix}. \qquad (3)$$

As a side note, we comment that an asymptotic (in $k$) expression for the capacity may be given by

$$\mathsf{cap}(\mathrm{RLL}_q(0,k)) = \log_2 q - \frac{(q-1)\log_2 e}{q^{k+2}}(1+o(1)). \qquad (4)$$

This expression agrees with the expression for the binary case $q = 2$ mentioned in [14] without proof or reference. For completeness, we bring a short proof of this claim in the appendix.

Having considered $(n, M; *)_k$ codes, we now turn to study $(n, M; t)_k$ codes for $t \in \mathbb{N} \cup \{0\}$. We note that $\mathbb{Z}_q^n$ is an optimal $(n, q^n; 0)_k$ code. Additionally, any $(n, M; *)_k$ code is trivially also an $(n, M; t)_k$ code, though not necessarily optimal.

We know by Lemma 14 that the descendant cones of two words overlap if and only if they are $k$-congruent. Thus, the strategy for constructing $(n, M; *)_k$ codes was to pick single representatives of the equivalence classes of $\sim_k$ as codewords. However, the overlap that is guaranteed by Lemma 14 may require a large number of duplication operations. If we are interested in a small enough value of $t$, then an $(n, M; t)_k$ code may contain several codewords from the same equivalence class. This observation will be formalized in the following, by introducing a metric on $k$-congruent words, and applying this metric to pick $k$-congruent codewords.

Fix a length $n \geq 1$, and let $x, x' \in \mathbb{Z}_q^n$, $x \sim_k x'$, be two $k$-congruent words of length $n$. We define the distance between $x$ and $x'$ as

$$d_k(x, x') = \min\left\{t \geq 0 \;\middle|\; D_k^t(x) \cap D_k^t(x') \neq \emptyset\right\}.$$

Since $x$ and $x'$ are $k$-congruent, Lemma 14 ensures that $d_k$ is well defined.

*Lemma 19: Let $x, x' \in \mathbb{Z}_q^n$, $x \sim_k x'$, be two $k$-congruent strings. Denote $\phi_k(x) = (y, z)$ and $\phi_k(x') = (y, z')$. Additionally, let*

$$\sigma_k(z) = (s_0, s_1, \dots, s_r),$$
$$\sigma_k(z') = (s'_0, s'_1, \dots, s'_r).$$

*Then*

$$d_k(x, x') = \frac{1}{2}\sum_{i=0}^{r} \left|s_i - s'_i\right| = \frac{1}{2} d_{\ell_1}(\sigma_k(z), \sigma_k(z')),$$

*where $d_{\ell_1}$ stands for the $\ell_1$-distance function.*

*Proof:* Let $x$ and $x'$ be two strings as required. By Corollary 12 we indeed have $y = y'$, and $\mu_k(z) = \mu_k(z')$. In particular, the length of the vectors of the zero signatures of $z$ and $z'$ are the same,

$$|\sigma_k(z)| = \left|\sigma_k(z')\right| = r + 1.$$

We now observe that the action of a $k$-tandem duplication on $x$ corresponds to the addition of a standard unit vector $e_i$ (an all-zero vector except for the $i$th coordinate which equals 1) to $\sigma_k(z)$.

Let $\tilde{x}$ denote a vector that is a descendant both of $x$ and $x'$, and that requires the least number of $k$-tandem duplications to reach from $x$ and $x'$. If we denote $\phi_k(\tilde{x}) = (\tilde{y}, \tilde{z})$, then we have

$$\begin{aligned} \tilde{y} &= y = y', \\ \mu_k(\tilde{z}) &= \mu_k(z) = \mu_k(z'), \\ \sigma_k(\tilde{z}) &= (\max(s_0, s'_0), \dots, \max(s_r, s'_r)). \end{aligned}$$

Thus,

$$\begin{aligned} d_k(x, x') &= \sum_{i=0}^{r} (\max(s_i, s'_i) - s_i) \\ &= \sum_{i=0}^{r} (\max(s_i, s'_i) - s'_i) \\ &= \frac{1}{2}\sum_{i=0}^{r} \left|s_i - s'_i\right| = \frac{1}{2} d_{\ell_1}(\sigma_k(z), \sigma_k(z')). \end{aligned}$$

■

From Lemma 19 we also deduce that $d_k$ is a metric over any set of $k$-congruent words of length $n$.

The following theorem shows that a code is $(n; t)_k$ if and only if the zero signatures of the $z$-part of $k$-congruent codewords in the $\phi_k$-transform domain, form a constant-weight code in the $\ell_1$-metric with distance at least $2(t+1)$. We recall that the $\ell_1$-metric weight of a vector $s = s_1\ s_2 \dots s_n \in \mathbb{Z}^n$ is defined as the $\ell_1$-distance to the zero vector, i.e.,

$$\mathrm{wt}_{\ell_1}(s) = \sum_{i=1}^{n} |s_i|.$$

*Theorem 20: Let $C \subseteq \mathbb{Z}_q^n$, $n \geq k$, be a subset of size $M$. Then $C$ is an $(n, M; t)_k$ code if and only if for each $y \in \mathbb{Z}_q^k$, $z \in \mathbb{Z}_q^{n-k}$, the following sets*

$$C(y, z) = \left\{\sigma_k(z') \;\middle|\; z' \in \mathbb{Z}_q^{n-k}, \mu_k(z) = \mu_k(z'), \phi_k^{-1}(y, z') \in C\right\}$$

*are constant-weight $(n(y, z), M(y, z), 2(t+1))$ codes in the $\ell_1$-metric, with constant weight*

$$\mathrm{wt}_{\ell_1}(\sigma(z)) = \frac{n - k - |\mu_k(z)|}{k},$$

*and length*

$$n(y,z) = \mathrm{wt}_H(z) + 1 = \mathrm{wt}_H(\mu_k(z)) + 1,$$

*where* $\mathrm{wt}_H$ *denotes the Hamming weight.*

*Proof:* In the first direction, let $C$ be an $(n,M;t)_k$ code. Fix $y$ and $z$, and consider the set $C(y,z)$. Assume to the contrary that there exist distinct $\sigma_k(z'), \sigma_k(z'') \in C(y,z)$, $z', z'' \in \mathbb{Z}_q^{n-k}$, such that $d_{\ell_1}(\sigma_k(z'), \sigma_k(z'')) \le 2t$ (note that $d_{\ell_1}$ between two vectors of the same weight is even).

The length of the code, $n(y,z)$, is obvious given (1). We note that $\sigma_k(z') \neq \sigma_k(z'')$ implies $z' \neq z''$. By definition, we have

$$\mu_k(z) = \mu_k(z') = \mu_k(z'').$$

Thus,

$$\begin{aligned}\mathrm{wt}_{\ell_1}(\sigma(z)) &= \mathrm{wt}_{\ell_1}(\sigma(z')) = \mathrm{wt}_{\ell_1}(\sigma(z'')) \\ &= \frac{n-k-|\mu_k(z)|}{k},\end{aligned}$$

where $|\mu_k(z)|$ denotes the length of the vector $\mu_k(z)$. Additionally, the two codewords

$$c' = \phi_k^{-1}(y,z') \in C \quad \text{and} \quad c'' = \phi_k^{-1}(y,z'') \in C$$

are $k$-congruent and distinct. By Lemma 19,

$$d_k(c',c'') = \frac{1}{2} d_{\ell_1}(\sigma_k(z'), \sigma_k(z'')) \le t. \tag{5}$$

However, that contradicts the code parameters since we have (5) imply $D_k^t(c') \cap D_k^t(c'') \neq \emptyset$, whereas in an $(n,M;t)_k$ code, the $t$-descendant cones of distinct codewords have an empty intersection.

In the other direction, assume that for every choice of $y$ and $z$, the corresponding $C(y,z)$ is a constant-weight code with minimum $\ell_1$-distance of $2(t+1)$. Assume to the contrary $C$ is not an $(n,M;t)_k$ code. Therefore, there exist two distinct codewords, $c', c'' \in C$ such that $d_k(c',c'') \le t$.

By Lemma 14 we conclude that $c'$ and $c''$ are $k$-congruent. Thus, there exist $y \in \mathbb{Z}_q^k$ and $z \in \mathbb{Z}_q^{n-k}$ ($z$ is not necessarily unique) such that,

$$\begin{aligned}\phi_k(c') &= (y,z') \\ \phi_k(c'') &= (y,z'') \\ \mu_k(z) &= \mu_k(z') = \mu_k(z'').\end{aligned}$$

We can now use Lemma 19 and obtain

$$d_{\ell_1}(\sigma_k(z'), \sigma_k(z'')) = 2\, d_k(c',c'') \le 2t,$$

which contradicts the minimal distance of $C(y,z)$. ■

With the insight given by Theorem 20 we now give a construction for $(n,M;t)_k$ codes.

*Construction B: Fix* $\Sigma = \mathbb{Z}_q$, $k \ge 1$, $n \ge k$, *and* $t \ge 0$. *Furthermore, for all*

$$\begin{aligned}1 &\le m \le n-k+1, \\ 0 &\le w \le \left\lfloor \frac{n-k}{k} \right\rfloor,\end{aligned}$$

*fix* $\ell_1$*-metric codes over* $\mathbb{Z}_q$*, denoted* $C_1(m,w)$*, which are of length* $m$*, constant* $\ell_1$*-weight* $w$*, and minimum* $\ell_1$*-distance* $2(t+1)$*. We construct*

$$C = \left\{ \phi_k^{-1}(y,z) \;\middle|\; \begin{aligned} & y \in \mathbb{Z}_q^k, z \in \mathbb{Z}_q^{n-k}, \\ & \sigma_k(z) \in C_1\left(\mathrm{wt}_H(\mu_k(z)) + 1, \frac{n-k-|\mu_k(z)|}{k}\right) \end{aligned} \right\}.$$

*Corollary 21: The code* $C$ *from Construction B is an* $(n,M;t)_k$ *code.*

*Proof:* Let $c, c' \in C$ be two $k$-congruent codewords, i.e., $\phi_k(c) = (y,z)$, $\phi_k(c') = (y,z')$, and $\mu_k(z) = \mu_k(z')$. It follows, by construction, that $\sigma_k(z)$ and $\sigma_k(z')$ belong to the same $\ell_1$-metric code with minimum $\ell_1$-distance at least $2(t+1)$. By Theorem 20, $C$ is an $(n,M;t)_k$ code. ■

Due to Theorem 20, a choice of optimal $\ell_1$-metric codes in Construction B will result in optimal $(n,M;t)_k$ codes. We are unfortunately unaware of explicit construction for such codes. However, we may deduce such a construction from codes for the similar Lee metric (e.g., [21]), while applying a standard averaging argument for inferring the existence of a constant-weight code. We leave the construction of such codes for a future work.

## IV. $\le k$-Tandem-Duplication Codes

In this section, we consider error-correcting codes that correct duplications of length at most $k$, which correspond to $S_{\le k}$. In particular, we present constructions for codes that can correct any number of duplications of length $\le 3$ as well as a lower bound on the capacity of the corresponding channel. In the case of duplications of length $\le 2$ we give optimal codes, and obtain the exact capacity of the channel.

It is worth noting that the systems $S_{\le k}$ were studied in the context of formal languages [17] and also in the context of coding and information theory [12]. In [17], it was shown that $S_{\le k}$, with $k \ge 4$, is not a regular language for alphabet size $|\Sigma| \ge 3$. However, it was proved in [12] that $S_{\le 3}$ is indeed a regular language irrespective of the seed string and the alphabet size.

In this paper, we will show that strings that can be generated by bounded tandem string-duplication systems with maximum duplication length 3 have a unique duplication root, a fact that will be useful for our code construction. Theorem 24 formalizes this statement. To simplify our description, we use the term *square* to denote a sequence of the form $\alpha^2 = \alpha\alpha$, where $\alpha \in \Sigma^*$. We begin with the following definition.

*Definition 22: Let two squares* $y_1 = \alpha\alpha \in \Sigma^+$ *and* $y_2 = \beta\beta \in \Sigma^+$ *appear as substrings of some string* $u \in \Sigma^*$*, i.e.,*

$$u = x_1\ y_1\ z_1 = x_2\ y_2\ z_2,$$

*with* $|x_1| = i$, $|x_2| = j$*. We say* $y_1$ *and* $y_2$ *are* overlapping squares *in* $u$ *if the following conditions both hold:*

1) $i \le j \le i + 2|\alpha| - 1$ *or* $j \le i \le j + 2|\beta| - 1$.
2) *If* $i = j$*, then* $\alpha \neq \beta$.

TABLE I

A List of All Overlapping Squares of Length at Most 3 (Up to a Permutation of the Alphabet Symbols)

| $\lvert u_v\rvert$ | $\lvert u_w\rvert$ | Overlapping squares $u_v$ and $u_w$ |
|---|---|---|
| 1 | 1 | *aaa* |
| 1 | 2 | *aaaa*, *aaaaa*, *aabab*, *ababb* |
| 1 | 3 | *aaaaaa*, *aaaaaaa*, *aaabaab*, *aabaab*, *aabaaba*, *aabaabb*, *aabbabb*, *aabcabc*, *abaaba*, *abaabaa*, *abbabb*, *abbabbb*, *abcabcc* |
| 2 | 2 | *aaaaa*, *aaaaaa*, *aaaaaaa*, *aaaabab*, *ababa*, *ababab*, *abababa*, *ababbbb*, *ababcbc* |
| 2 | 3 | *aaaaaa*, *aaaaaaa*, *aaaaaaaa*, *aaaaaaaaa*, *aaaaaabab*, *aaaaabaab*, *aaaabaab*, *aaaabaaba*, *aaaabbabb*, *aaaabcabc* *aabaabab*, *aabaababa*, *aabaabbbb*, *aabaabcbc*, *abaabaaaa*, *abaabab*, *abaababa*, *abaababab*, *abaabacac*, *abababaaba* *ababaabaa*, *abababbab*, *ababacbac*, *ababbab*, *ababbabb*, *ababbabba*, *ababbbbbb*, *ababbcbbc*, *ababcabc*, *ababcabca* *ababcbbcb*, *ababccbcc*, *ababcdbcd*, *abbabbaba*, *abbabbbb*, *abbabbbbb*, *abbabbcbc*, *abcabcaca*, *abcabcbc*, *abcabcbcb* *abcabcccc*, *abcabcdcd* |
| 3 | 3 | *aaaaaaa*, *aaaaaaaa*, *aaaaaaaaa*, *aaaaaaaaaa*, *aaaaaaaaaaa*, *aaaaaaabaab*, *aaaaaabaab*, *aaaaaabaaba*, *aaaaaabbabb*, *aaaaaabcabc* *aabaaba*, *aabaabaa*, *aabaabaab*, *aabaabaaba*, *aabaabaabaa*, *aabaababbab*, *aabaabacbac*, *aabaabbabb*, *aabaabbabba*, *aabaabbbbbb* *aabaabbcbbc*, *aabaabcabc*, *aabaabcabca*, *aabaabcbbcb*, *aabaabccbcc*, *aabaabcdbcd*, *abaabaa*, *abaabaaaaaa*, *abaabaab*, *abaabaaba* *abaabaabaa*, *abaabaabaab*, *abaabaacaac*, *abaababaaba*, *abaababbab*, *abaababbabb*, *abaababcabc*, *abaabacaaca*, *abaabacbac*, *abaabacbacb* *abaabaccacc*, *abaabacdacd*, *abbabba*, *abbabbaabaa*, *abbabbab*, *abbabbabb*, *abbabbabba*, *abbabbabbab*, *abbabbacbac*, *abbabbbabba* *abbabbbbbb*, *abbabbbbbbb*, *abbabbbcbbc*, *abbabbcabca*, *abbabbcbbc*, *abbabbcbbcb*, *abbabbccbcc*, *abbabbcdbcd*, *abcabca*, *abcabcaacaa* *abcabcab*, *abcabcabc*, *abcabcabca*, *abcabcabcab*, *abcabcaccac*, *abcabcadcad*, *abcabcbacba*, *abcabcbbcb*, *abcabcbbcbb*, *abcabcbccbc* *abcabcbdcbd*, *abcabccacca*, *abcabccbcc*, *abcabccbccb*, *abcabcccccc*, *abcabccdccd*, *abcabcdacda*, *abcabcdbcd*, *abcabcdbcdb*, *abcabcdccdc* *abcabcddcdd*, *abcabcdecde* |

*Example 23: Consider the sequence u,*

$$u = 0\,1\,\overbrace{\underbrace{2\,3\,2\,3}_{\beta_1\beta_1}\underbrace{4\,5\,2\,4\,5\,2}_{\beta_2\beta_2}3\,2\,3\,4\,5\rlap{$\underbrace{2\,4\,5\,6\,2\,4\,5\,6}_{\beta_3\beta_3}$}2\,4\,5}^{\alpha\alpha}6\,2\,4\,5\,6\,7,$$

*where $\alpha\alpha$ and $\beta_i\beta_i$ for each $i \in \{1, 2, 3\}$ are overlapping squares.* □

The following theorem shows that every word has a unique root under tandem deduplication of length up to 3.

*Theorem 24: For any $z \in \Sigma^*$ we have $\left|R_{\leq 3}(z)\right| = 1$.*

*Proof:* Fix some $z \in \Sigma^*$, and assume $z$ has exactly $m$ distinct roots, $R_{\leq 3}(z) = \{y_1, y_2, \ldots, y_m\}$. Let us assume to the contrary that $m \geq 2$.

Let us follow a deduplication sequence starting at $x_0 = z$. At each step, we deduplicate $x_i \underset{\leq 3}{\Longleftarrow} x_{i+1}$, and we must have $\left|R_{\leq 3}(x_i)\right| \geq \left|R_{\leq 3}(x_{i+1})\right|$. At each step, out of the possible immediate ancestors of $x_i$, we choose $x_{i+1}$ to be one with $\left|R_{\leq 3}(x_{i+1})\right| \geq 2$ if possible. Since the end-point of a deduplication process is an irreducible sequence, we must reach a sequence $x$ in the deduplication process with the following properties:

1) $z \overset{*}{\underset{\leq 3}{\Longleftarrow}} x$
2) $\left|R_{\leq 3}(x)\right| \geq 2$
3) For each $x' \in \Sigma^*$ such that $x \underset{\leq 3}{\Longleftarrow} x'$, $\left|R_{\leq 3}(x')\right| = 1$.
4) There exist $v, w \in \Sigma^*$ such that $x \underset{\leq 3}{\Longleftarrow} v$ and $x \underset{\leq 3}{\Longleftarrow} w$ with $\left|R_{\leq 3}(v)\right| = \left|R_{\leq 3}(w)\right| = 1$.
5) $R_{\leq 3}(v) = \{y_i\} \neq \{y_j\} = R_{\leq 3}(w)$.

Intuitively, in the deduplication process starting from $z$, we reach a sequence $x$ with more than one root, but any following single deduplication moves us into a single descendant cone of one of the roots of $z$. We note that all ancestors of $v$ must have a single root $y_i$, and all ancestors of $w$ must have a single root $y_j$.

Thus, $x$ must contain a square $u_v u_v$ whose deduplication results in $v$, and a square $u_w u_w$ whose deduplication results in $w$. We contend that the squares $u_v u_v$ and $u_w u_w$ overlap. Otherwise, if $u_v u_v$ and $u_w u_w$ do not overlap in $x$, we may deduplicate them in any order to obtain the same result. Hence, there exists $t \in \Sigma^*$ such that $v \underset{\leq 3}{\Longleftarrow} t$ and $w \underset{\leq 3}{\Longleftarrow} t$. But then, since $t$ is an ancestor both of $v$ and $w$,

$$\{y_i\} = R_{\leq 3}(v) = R_{\leq 3}(t) = R_{\leq 3}(w) = \left\{y_j\right\},$$

a contradiction.

We now know that $u_v u_v$ and $u_w u_w$ must overlap. We also note $\left|u_v\right|, \left|u_w\right| \leq 3$. Let $a, b, c \in \Sigma$ be three distinct symbols. If the alphabet is smaller, then some of the cases below may be ignored, and the proof remains the same. We use brute force to enumerate all the overlapping squares, and the results are given in Table I. In the table, each string describes the shortest subsequence that contains the overlapping squares. The enumeration is complete, up to a permutation of the alphabet symbols. For example, if $u_v = abc$ and $u_w = cbc$, the corresponding string appears in the table as *abcabcbccbc* since we have,

$$\overbrace{a\ b\ c}^{u_v}\ \overbrace{a\ b\ \rlap{$\underbrace{c\ b\ c}_{u_w}$}c}^{u_v}\ b\ c\ \underbrace{c\ b\ c}_{u_w}$$

It is tedious, yet easy, to check that each of the cases in Table I has a unique root if deduplication of maximum length 3 is allowed.[2] In the above example, indeed, the only possible root is *abc*,

$$\underline{abcabc}bccbc \underset{\leq 3}{\Longleftarrow} abcbccbc \overset{*}{\underset{\leq 3}{\Longleftarrow}} abc,$$
$$abcab\underline{cbccbc} \underset{\leq 3}{\Longleftarrow} abcabcbc \overset{*}{\underset{\leq 3}{\Longleftarrow}} abc.$$

Let $x = \alpha\beta\gamma \in \Sigma^*$, where $\beta$ covers exactly the overlapping squares, and is one of the above listed cases. Then, by deduplication of $u_v u_v$ from $\beta$ in $x$, we get $v$, and by deduplication of $u_w u_w$ from $\beta$ in $x$, we get $w$. However, since $\beta$ has a unique root, we may deduplicate $v$ and $w$ to the same word $t = \alpha\beta'\gamma \in \Sigma^*$, where $R(\beta) = \left\{\beta'\right\}$, i.e., $\beta'$ is the unique

[2]We used a computer to verify the list of overlapping squares and the fact that they each have a unique root.

root of $\beta$. Thus, $t$ is an ancestor of both $v$ and $w$. Again,

$$\{y_i\} = R_{\le 3}(v) = R_{\le 3}(t) = R_{\le 3}(w) = \{y_j\},$$

which is a contradiction. ■

*Corollary 25: For any $z \in \Sigma^*$ we have $|R_{\le k}(z)| = 1$ for $k = 1, 2$.*

In a similar fashion to the previous section, we define the following relation. We say $x, x' \in \Sigma^*$ are $\le 3$-congruent, denoted $x \sim_{\le 3} x'$, if $R_{\le 3}(x) = R_{\le 3}(x')$. Clearly $\sim_{\le 3}$ is an equivalence relation. Having shown any sequence has a unique root when duplicating up to length 3, we obtain the following corollary.

*Corollary 26: For any two words $x, x' \in \Sigma^*$, if*

$$D^*_{\le 3}(x) \cap D^*_{\le 3}(x') \ne \emptyset$$

*then $x \sim_{\le 3} x'$.*

We note that unlike Lemma 14, we do not have $x \sim_{\le 3} x'$ necessarily imply that their descendant cones intersect. Here is a simple example illustrating this case. Fix $q = 3$, and let $x = 012012$ and $x' = 001122$. We note that $x \sim_{\le 3} x'$, since

$$R_{\le 3}(x) = R_{\le 3}(x') = \{012\}.$$

However, $D^*_{\le 3}(x) \cap D^*_{\le 3}(x') = \emptyset$ since all the descendants of $x$ have a 0 to the right of a 2, whereas none of the descendants of $x'$ do.

We are missing a simple operator which is required to define an error-correcting code. For any sequence $x \in \Sigma^+$, we define its $k$-suffix-extension to be

$$\xi_k(x) = x\ (\mathrm{Suff}_1(x))^k,$$

i.e., the sequence $x$ with its last symbol repeated an extra $k$ times.

*Construction C: Let $\Sigma$ be some finite alphabet. We construct the code*

$$C = \bigcup_{i=1}^{n} \left\{\xi_{n-i}(x) \mid x \in \mathrm{Irr}_{\le 3}(i)\right\}.$$

*Theorem 27: The code $C$ from Construction C is an $(n, M; *)_{\le 3}$ code, where*

$$M = \sum_{i=1}^{n} \left|\mathrm{Irr}_{\le 3}(i)\right|.$$

*Proof:* The parameters of the code are obvious. Since the last letter duplication induced by the suffix extension may be deduplicated, we clearly have exactly one codeword from each equivalence class of $\sim_{\le 3}$. By Corollary 26, the descendant cones of the codewords do not intersect and the code can indeed correct all errors. ■

*Example 28: Let $n = 4$ and $\Sigma = \{0, 1, 2\}$. The code $C_{\le 3}$ obtained by Construction C is given by*

$$C_{\le 3} = \bigcup_{\{a,b,c\} = \Sigma} \left\{\underline{a}aaa, \underline{ab}bb, \underline{aba}a, \underline{abc}c, \underline{abac}, \underline{abca}, \underline{abcb}\right\},$$

*where in each codeword, the irreducible part is underlined, and the union is taken over all possible assignments of a, b, and c, to distinct symbols of $\Sigma$. Thus $|C_{\le 3}| = 7 \cdot 3! = 42$.* □

For the remainder of the section we denote by $\mathrm{Irr}_{q;\le 3}$ the set of irreducible words with respect to $\underset{\le 3}{\Longleftarrow}$ over $\mathbb{Z}_q$, in order to make explicit the dependence on the size of the alphabet. We also assume $q \ge 3$, since $q = 2$ is a trivial case with

$$\mathrm{Irr}_{2;\le 3} = \{0, 1, 01, 10, 010, 101\}. \qquad (6)$$

We observe that $\mathrm{Irr}_{q;\le 3}$ is a regular language. Indeed, it is defined by a finite set of subsequences we would like to avoid. This set is exactly

$$\mathcal{F}_q = \left\{ uu \in \mathbb{Z}_q^* \;\middle|\; 1 \le |u| \le 3 \right\}.$$

We can easily construct a finite directed graph with labeled edges such that paths in the graph generate exactly $\mathrm{Irr}_{q;\le 3}$. This graph is obtained by taking the De Bruijn graph $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q)$ of order 5 over $\mathbb{Z}_q$, i.e., $\mathcal{V}_q = \mathbb{Z}_q^5$, and edges of the form $(a_1, a_2, a_3, a_4, a_5) \to (a_2, a_3, a_4, a_5, a_6)$, for all $a_i \in \mathbb{Z}_q$ (for more on De Bruijn graphs the reader is referred to [23, Ch. 8]). Thus, each edge is labeled with a word $w = (a_1, a_2, a_3, a_4, a_5, a_6) \in \mathbb{Z}_q^6$. We then remove all edges labeled by words $\alpha\beta\gamma \in \mathbb{Z}_q^6$ such that $\beta \in \mathcal{F}_q$. We call the resulting graph $\mathcal{G}'_q$. It is easy to verify that each path in $\mathcal{G}'_q$ generates a sequence of sliding windows of length 6. Reducing each window to its first letter we get exactly $\mathrm{Irr}_{q;\le 3}$. An example showing $\mathcal{G}'_3$ is given in Figure 1. Finally, using known techniques [18], we can calculate $\mathsf{cap}(\mathrm{Irr}_{q;\le 3})$.

*Corollary 29: For all $q \ge 3$,*

$$\mathsf{cap}_q(*)_{\le 3} \ge \mathsf{cap}(\mathrm{Irr}_{q;\le 3}).$$

*Proof:* Let $M_n$ denote the size of the length $n$ code over $\mathbb{Z}_q$ from Construction C. By definition, $A_q(n; *)_{\le 3} \ge M_n$. We note that trivially

$$M_n = \sum_{i=1}^{n} \left|\mathrm{Irr}_{q;\le 3}(i)\right| \ge \left|\mathrm{Irr}_{q;\le 3}(n)\right|.$$

Plugging this into the definition of the capacity gives us the desired claim. ■

*Example 30: Using the constrained system presented in Figure 1 that generates $\mathrm{Irr}_{3;\le 3}$, we can calculate*

$$\mathsf{cap}_3(*)_{\le 3} \ge 0.347934.$$

□

Stronger statements may be given when the duplication length is upper bounded by 2 instead of 3.

*Lemma 31: For all $x, x' \in \Sigma^*$, we have*

$$D^*_{\le 2}(x) \cap D^*_{\le 2}(x') \ne \emptyset$$

*if and only if $x \sim_{\le 2} x'$.*

*Proof:* In the first direction, assume $x \nsim_{\le 2} x'$. By the uniqueness of the root from Corollary 25, let us denote $R_{\le 2}(x) = \{u\}$ and $R_{\le 2}(x') = \{u'\}$, with $u \ne u'$. If there exists $w \in D^*_{\le 2}(x) \cap D^*_{\le 2}(x')$, then $w$ is a descendant of both $u$ and $u'$, therefore $u$ and $u' \in R_{\le 2}(w)$, which is a contradiction. Hence, no such $w$ exists, i.e., $D^*_{\le 2}(x) \cap D^*_{\le 2}(x') = \emptyset$.

In the other direction, assume $x \sim_{\le 2} x'$. We construct a word $w \in D^*_{\le 2}(x) \cap D^*_{\le 2}(x')$. Let $R_{\le 2}(x) = R_{\le 2}(x') = \{v\}$, and denote $v = a_1 a_2 \dots a_m$, where $a_i \in \Sigma$. Consider the tandem-duplication string system $S_{\le 2} =$

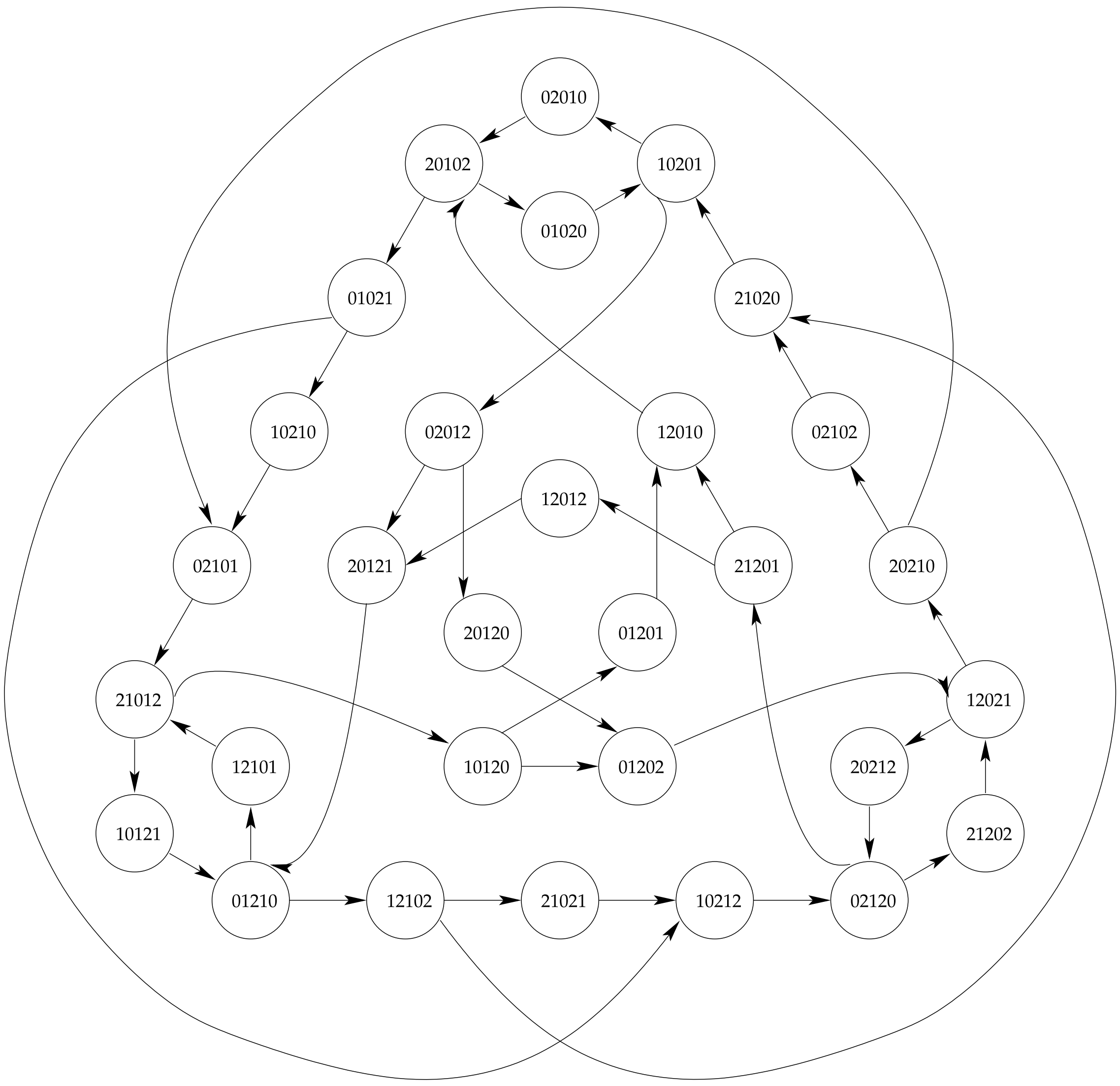

Fig. 1. The graph $\mathcal{G}'_3$ producing the set of ternary irreducible words $\mathrm{Irr}_{3;\leq 3}$. Vertices without edges were removed as well.

$(\Sigma, \upsilon, \mathcal{T}_{\leq 2})$. Using [12], the regular expression for the language generated by $S_{\leq 2}$ is given by

$$a_1^+ a_2^+ (a_1^+ a_2^+)^* a_3^+ (a_2^+ a_3^+)^* \dots a_m^+ (a_{m-1}^+ a_m^+)^*.$$

Since $x, x' \in S$, we have

$$x = \prod_{i=1}^{\alpha_1} (a_1^{p_{1i}} a_2^{q_{1i}})\ a_3^{q_{21}} \prod_{i=2}^{\alpha_2} (a_2^{p_{2i}} a_3^{q_{2i}}) \dots a_m^{q_{(m-1)1}} \prod_{i=2}^{\alpha_{m-1}} (a_{m-1}^{p_{(m-1)i}} a_m^{q_{(m-1)i}}),$$

and

$$x' = \prod_{i=1}^{\beta_1} (a_1^{e_{1i}} a_2^{f_{1i}})\ a_3^{f_{21}} \prod_{i=2}^{\beta_2} (a_2^{e_{2i}} a_3^{f_{2i}}) \dots a_m^{f_{(m-1)1}} \prod_{i=2}^{\beta_{m-1}} (a_{m-1}^{e_{(m-1)i}} a_m^{f_{(m-1)i}}),$$

where $\prod$ represents concatenation and $p_{ji}$, $q_{ji}$, $e_{ji}$, $f_{ji}$, $\alpha_j$, $\beta_j \geq 1$. Now, it is easy to observe that we can obtain

$$w = \prod_{i=1}^{\gamma_1} (a_1^{g_1} a_2^{h_1})\ a_3^{h_2} \prod_{i=2}^{\gamma_2} (a_2^{g_2} a_3^{h_2}) \dots a_m^{h_{m-1}} \prod_{i=2}^{\gamma_{m-1}} (a_{m-1}^{g_{m-1}} a_m^{h_{m-1}})$$

by doing tandem duplication of length up to 2 on $x$ and $x'$, and choosing $\gamma_j = \max\{\alpha_j, \beta_j\}$, $g_j = \max_i \{p_{ji}, e_{ji}\}$, and $h_j = \max_i \{q_{ji}, f_{ji}\}$. Note, $p_{ji}$ and $q_{ji}$ are assumed to be 0 for $i > \alpha_j$ and $e_{ji}$ and $f_{ji}$ are assumed to be 0 for $i > \beta_j$. Thus, $w \in D_{\leq 2}(x) \cap D_{\leq 2}(x')$. ■

*Construction 1: Let $\Sigma$ be some finite alphabet. The constructed code is*

$$C = \bigcup_{i=1}^{n} \left\{ \xi_{n-i}(x) \mid x \in \mathrm{Irr}_{\leq 2}(i) \right\}.$$

*Theorem 32: The code $C$ from Construction 1 is an optimal $(n, M; *)_{\leq 2}$ code, where*

$$M = \sum_{i=1}^{n} \left| \mathrm{Irr}_{\leq 2}(i) \right|.$$

*Proof:* The correctness of the parameters follows the same reasoning as the proof of Theorem 27. By Lemma 31, any two distinct codewords of an $(n; *)_{\leq 2}$ code must belong to different equivalence classes of $\sim_{\leq 2}$. The code $C$ of Construction 1 contains exactly one codeword from each equivalence class of $\sim_{\leq 2}$, and thus, it is optimal. ■

*Corollary 33: For all $q \geq 3$,*

$$\mathsf{cap}_q(*)_{\leq 2} = \mathsf{cap}(\mathrm{Irr}_{q;\leq 2}).$$

*Proof:* Let $M_n$ denote the size of the length $n$ code over $\mathbb{Z}_q$ from Construction 1. By definition, $A_q(n; *)_{\leq 2} \geq M_n$. We note that trivially

$$M_n = \sum_{i=1}^{n} \left|\mathrm{Irr}_{q;\leq 2}(i)\right| \geq \left|\mathrm{Irr}_{q;\leq 2}(n)\right|.$$

Additionally, $\left|\mathrm{Irr}_{q;\leq 2}\right|(n)$ is monotone increasing in $n$ since any irreducible length-$n$ word $x$ may be extended to an irreducible word of length $n+1$ by adding a letter that is not one of the last two letters appearing in $x$. Thus,

$$M_n = \sum_{i=1}^{n} \left|\mathrm{Irr}_{q;\leq 2}(i)\right| \leq n\left|\mathrm{Irr}_{q;\leq 2}(n)\right|.$$

Plugging this into the definition of the capacity gives us the desired claim. ■

## V. Duplication Roots

In Section III, we stated that if the duplication length is uniform (i.e., a constant $k$ ), then every sequence has a unique root. Further in Section IV, we proved in Theorem 24 that if the duplication length is bounded by 3 (i.e. $\leq 3$), then again every sequence will have a unique root. The full characterization of all cases that have a unique root is stated in Theorem 40. Before moving to Theorem 40, we present an example and some lemmas required to prove the theorem.

*Example 34: Let $U = \{2, 3, 4\}$ be a set of duplication lengths and $\Sigma = \{1, 2, 3\}$. Consider*

$$z = \overbrace{1\,2\,3\,2\,1\,2\,3\,2}^{\alpha\alpha}\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\underbrace{\;\;\;\;\;\;\;\;\;\;\;\;\;\;\;\;\;}_{\beta\beta}\,3.$$

*The sequence $z$ has two tandem repeats $\alpha\alpha$ and $\beta\beta$ with $|\alpha| = 4$ and $|\beta| = 2$. If we deduplicate $\alpha\alpha$ first from $z$, we get*

$$123212323 \underset{4}{\Longleftarrow} 12323 \underset{2}{\Longleftarrow} 123.$$

*However, if we deduplicate $\beta\beta$ first from $z$ we get*

$$123212323 \underset{2}{\Longleftarrow} 1232123.$$

*Both 123 and 1232123 are irreducible and thus roots of $z$.* □

Theorem 40 generalizes the statement presented in the example above to any set of duplication lengths. We naturally extend all previous notation to allow duplication and deduplication of several lengths by replacing the usual $k$ subscript with a set $U$, where $U \subseteq \mathbb{N}$. For example, $R_U(z)$ denotes the set of roots obtained via a sequence of deduplications with lengths from $U$, starting with the string $z$. The property we would like to study is formally defined next.

*Definition 35: Let $\Sigma \neq \emptyset$ be an alphabet, and $U \subseteq \mathbb{N}$, $U \neq \emptyset$, a set of tandem-duplication lengths. We say $(\Sigma, U)$ is a* unique-root pair, *iff for all $z \in \Sigma^*$ we have $|R_U(z)| = 1$. Otherwise, we call $(\Sigma, U)$ a* non-unique-root pair.

We observe that the actual identity of the letters in the alphabet is immaterial, and only the size of $\Sigma$ matters. Additionally, simple monotonicity is evident: If $(\Sigma, U)$ is a unique-root pair, then so is $(\Sigma', U)$, for all $\Sigma' \subseteq \Sigma$. Similarly, if $(\Sigma, U)$ is a non-unique-root pair, then so is $(\Sigma', U)$, for all $\Sigma \subseteq \Sigma'$.

The following sequence of lemmas will provide the basis for a full classification of unique-root pairs.

*Lemma 36: Let $\Sigma = \{a\}$ be an alphabet with only a single letter. Let $U \subseteq \mathbb{N}$, and denote $k = \min(U)$. Then $(\Sigma, U)$ is a unique-root pair if and only if $k|m$ for all $m \in U$.*

*Proof:* If $k|m$ for all $m \in U$, then any sequence $a^n$, $n \in \mathbb{N}$ has a unique root

$$a^n \overset{*}{\underset{U}{\Longleftarrow}} a^{k+(n \bmod k)},$$

where in the expression above $n \bmod k$ denotes the unique integer from $\{1, 2, \dots, k\}$ with the same residue modulo $k$ as $n$.

In the other direction, if there exists $m \in U$ such that $k \nmid m$, let us consider the sequence $a^{k+2m}$. By first deduplicating a length $m$ sequence, and then via as many deduplications of length $k$ as needed, we obtain

$$a^{k+2m} \underset{U}{\Longleftarrow} a^{k+m} \overset{*}{\underset{U}{\Longleftarrow}} a^{k+(m \bmod k)} = x.$$

However, by only deduplicating length $k$ sequences, we also get

$$a^{k+2m} \overset{*}{\underset{U}{\Longleftarrow}} a^{k+(2m \bmod k)} = y.$$

Both $x$ and $y$ are irreducible since $1 \leq |x|, |y| \leq k$. However, since $m \not\equiv 0 \pmod{k}$, we have

$$m \not\equiv 2m \pmod{k},$$

and therefore $x \neq y$, and $a^{k+2m}$ has two distinct roots. ■

*Lemma 37: Let $\Sigma$ be an alphabet, $|\Sigma| \geq 2$, $km > 1$, and $U = \{k, k+m\} \cup V$, where $V \subseteq \mathbb{N} \setminus \{1, 2, \dots, k+m\}$. Then $(\Sigma, U)$ is a non-unique-root pair.*

*Proof:* By Lemma 36 and monotonicity, if $k \nmid m$, then $(\Sigma, U)$ is already a non-unique-root pair, and we are done. Thus, for the rest of the proof we assume $m = \ell k$, for some $\ell \in \mathbb{N}$.

Let $a, b \in \Sigma$ be two distinct letters, and let $v_1 v_2 \dots v_{k+m} \in \Sigma^{k+m}$ be a sequence defined as follows:

$$v_i = \begin{cases} a & i < k+m \text{ and } \lceil i/k \rceil \text{ is odd,} \\ b & i < k+m \text{ and } \lceil i/k \rceil \text{ is even,} \\ v_m & i = k+m. \end{cases}$$

Consider now the sequence

$$z = v_1\, v_2 \dots v_{k+m} v_1\, v_2 \dots v_{k+m} v_{m+1} \dots v_{k+m-1}.$$

We can write $z$ as

$$\begin{aligned} z &= (v_1\, v_2 \dots v_{k+m-1} v_m)^2\, v_{m+1} \dots v_{k+m-1} \\ &= v_1\, v_2 \dots v_{k+m-1} v_m v_1\, v_2 \dots v_{m-1} (v_m v_{m+1} \dots v_{k+m-1})^2. \end{aligned}$$

As is evident, there are two squares in $z$, one of which is of length $2k+2m$ and the other is of length $2k$. Deduplicating the square of length $2k+2m$ in $z$ first gives

$$\begin{aligned} z &\underset{U}{\Longleftarrow} v_1\ v_2 \dots v_{k+m-1} v_m v_{m+1} \dots v_{k+m-1} \\ &\underset{U}{\Longleftarrow} v_1\ v_2 \dots v_{k+m-1} = y. \end{aligned}$$

Deduplicating the square of length $2k$ first gives

$$z \underset{U}{\Longleftarrow} v_1\ v_2 \dots v_{k+m-1} v_m v_1\ v_2 \dots v_{k+m-1} = x.$$

We note that $|x| = 2k+2m-1$ and $|y| = k+m-1$. Thus, if further deduplications are possible, they must be deduplications of length $k$, since both $x$ and $y$ are too short to allow deduplications of other allowed lengths from $U$. We observe that $y$ is certainly irreducible, since it is made up of alternating blocks of $a$'s and $b$'s of length $k$. However, it is conceivable that $x$ may be further deduplicated to obtain $y$.

We recall $m = \ell k$. Depending on the parity of $\ell$, we have two cases. If $\ell$ is even, we can write explicitly

$$\begin{aligned} y &= (a^k b^k)^{\ell/2} a^{k-1}, \\ x &= (a^k b^k)^{\ell/2} a^{k-1} b\ (a^k b^k)^{\ell/2} a^{k-1}. \end{aligned}$$

The sequence $x$ may be further deduplicated, by noting the square $ba^{k-1}ba^{k-1}$, to obtain

$$x \underset{U}{\Longleftarrow} (a^k b^k)^{\ell} a^{k-1} = x'.$$

We easily observe that $x'$ is irreducible, and $x' \neq y$ since their lengths differ, $|y| = (\ell+1)k-1$, $|x| = (2\ell+1)k-1$, and $\ell \geq 1$.

If $\ell$ is odd, we explicitly write

$$\begin{aligned} y &= (a^k b^k)^{(\ell-1)/2} a^k b^{k-1}, \\ x &= (a^k b^k)^{(\ell-1)/2} a^k b^{k-1} a\ (a^k b^k)^{(\ell-1)/2} a^k b^{k-1}. \end{aligned}$$

We recall our requirement that $km > 1$, which translates to $k \geq 1$, $\ell \geq 1$ and odd, but not $k = \ell = 1$. If $k \neq 1$ and $\ell \neq 1$, we easily see that $x$ is irreducible, $x \neq y$. If $\ell = 1$ and $k \neq 1$, we have $x = a^k b^{k-1} a^{k+1} b^{k-1}$ which is again irreducible, and $x \neq y$. The final case is $k = 1$ and $\ell \neq 1$, in which

$$x = (ab)^{(\ell-1)/2} a^2 (ab)^{(\ell-1)/2} a \overset{*}{\underset{U}{\Longleftarrow}} (ab)^{\ell-1} a = x'$$

by twice deduplicating the square $a^2$. However, $y = (ab)^{(\ell-1)/2} a$, and $y \neq x$ since $|y| = 1 + (\ell-1)/2$ and $|x| = \ell$, while $\ell \geq 3$. ■

*Lemma 38: For any alphabet* $\Sigma$, $|\Sigma| \geq 3$, *and for any* $V \subseteq \mathbb{N} \setminus \{1,2,3\}$, $V \neq \emptyset$, *if* $U = \{1,2\} \cup V$, *then* $(\Sigma, U)$ *is a non-unique-root pair.*

*Proof:* Let $a, b, c \in \Sigma$ be distinct symbols, and let $m = \min(V)$. Consider the sequence

$$z = ab^{m-3}caab^{m-3}ca.$$

We now have the following two distinct roots,

$$\begin{aligned} z &\underset{U}{\Longleftarrow} ab^{m-3}ca \overset{*}{\underset{U}{\Longleftarrow}} abca, \\ z &\underset{U}{\Longleftarrow} ab^{m-3}cab^{m-3}ca \overset{*}{\underset{U}{\Longleftarrow}} abcabca. \end{aligned}$$

■

*Lemma 39: For any alphabet* $\Sigma$, $|\Sigma| \geq 3$, *and for any* $V \subseteq \mathbb{N} \setminus \{1,2,3\}$, $V \neq \emptyset$, *if* $U = \{1,2,3\} \cup V$, *then* $(\Sigma, U)$ *is a non-unique-root pair.*

*Proof:* Let $a, b, c \in \Sigma$ be 3 distinct symbols. Consider the sequence

$$z = ab^{m-3}cbab^{m-3}cbc,$$

where $m = \min(V)$. We now have the following two distinct roots,

$$\begin{aligned} z &\underset{U}{\Longleftarrow} ab^{m-3}cbc \overset{*}{\underset{U}{\Longleftarrow}} abcbc \underset{U}{\Longleftarrow} abc, \\ z &\underset{U}{\Longleftarrow} ab^{m-3}cbab^{m-3}c \overset{*}{\underset{U}{\Longleftarrow}} abcbabc. \end{aligned}$$

■

We are now in a position to provide a full classification of unique-root pairs.

*Theorem 40: Let* $\Sigma \neq \emptyset$ *be an alphabet, and* $U \subseteq \mathbb{N}$, $U \neq \emptyset$, *a set of tandem-duplication lengths. Denote* $k = \min(U)$. *Then* $(\Sigma, U)$ *is a unique-root pair if and only if it matches one of the following cases:*

| | |
|---|---|
| $\lvert\Sigma\rvert = 1$ | $U \subseteq k\mathbb{N}$ |
| $\lvert\Sigma\rvert = 2$ | $U = \{k\}$ |
| | $U \supseteq \{1,2\}$ |
| $\lvert\Sigma\rvert \geq 3$ | $U = \{k\}$ |
| | $U = \{1,2\}$ |
| | $U = \{1,2,3\}$ |

*Proof:* The case of $|\Sigma| = 1$ is given by Lemma 36. The case of $|U| = 1$ was proved in [17], and with an alternative proof, in Section III. The case of $|\Sigma| = 2$ and $\{1,2\} \not\subseteq U$, was proved in Lemma 37. It is also folklore that having $|\Sigma| = 2$ and $\{1,2\} \subseteq U$ gives a unique-root pair, since we can always deduplicate runs of symbols to single letters, and then deduplicate pairs, to obtain one of only six possible roots: $a$, $b$, $ab$, $ba$, $aba$, $bab$. The choice of root depends only on the first letter of the word, its last letter, and when they're the same, on the existence of a different letter inside. No deduplication actions change those, regardless of the length of the deduplication.

When $|\Sigma| \geq 3$, the unique-root property for $U = \{1,2\}$ and $U = \{1,2,3\}$ was established in Corollary 25 and Theorem 24, respectively. The non-unique-root property for the other cases was proved in Lemma 37, Lemma 38, and Lemma 39. ■

## VI. Conclusion

We provided error-correcting codes, and in some cases, exact capacity, for the tandem-duplication channel. These codes mostly rely on unique-root pairs of alphabets and duplication lengths, which we also investigated. Several interesting questions remain open. In particular, we do not know yet how to construct general $(n, M; *)_U$ over $\Sigma$, especially when we do not necessarily have unique roots. We also mention the interesting combinatorial problem of counting the number of distinct roots of a string, and finding strings of a given length with as many roots as possible.

## Appendix

We provide a short proof of (4). We need to estimate the largest eigenvalue of $A_q(k)$ from (3), i.e., to estimate the largest root $\lambda$ of its characteristic polynomial

$$\chi_{A_q(k)}(x) = \frac{x^{k+2} - qx^{k+1} + q - 1}{x-1}.$$

Since this largest root is strictly greater than 1, we can alternatively find the largest root of the polynomial

$$f(x) = x^{k+2} - qx^{k+1} + q - 1.$$

We shall require the following simple bounds. Taking the first term in the Taylor expansion of $e^x$, and the error term, we have for all $x > 0$,

$$e^x = 1 + xe^{x'},$$

for some $x' \in [0, x]$. Since $x > 0$ and $e^x$ is increasing, we have

$$e^x = 1 + xe^{x'} \le 1 + xe^x,$$

or alternatively,

$$1 - e^x \ge -xe^x. \tag{7}$$

Similarly, taking the first two terms of the Taylor expansion, for all $x > 0$, we get the well-known bound

$$e^x > 1 + x. \tag{8}$$

We return to the main proof. In the first direction, let us first examine what happens when we set

$$x = qe^{-\frac{q-1}{q^{k+2}}}.$$

Then

$$\begin{aligned} f(x) &= q^{k+2}e^{-\frac{q-1}{q^{k+2}}(k+2)} - q^{k+2}e^{-\frac{q-1}{q^{k+2}}(k+1)} + q - 1 \\ &= q^{k+2}e^{-\frac{q-1}{q^{k+2}}(k+2)}\left(1 - e^{\frac{q-1}{q^{k+2}}}\right) + q - 1 \\ &\overset{(a)}{\ge} (q-1)\left(1 - e^{-\frac{q-1}{q^{k+2}}(k+1)}\right) \\ &> 0, \end{aligned}$$

where (a) follows by an application of (7).

In the other direction, we examine the value of $f(x)$ when we set

$$x = qe^{-\frac{q-1}{q^{k+2}}\alpha},$$

where $\alpha$ is a constant depending on $q$ and $k$. To specify $\alpha$ we recall $W(z)$, $z \ge -\frac{1}{e}$, denotes the Lambert $W$-function, defined by

$$W(z)e^{W(z)} = z.$$

We define

$$\alpha = \frac{W\left(-\frac{q-1}{q^{k+2}}(k+2)\right)}{-\frac{q-1}{q^{k+2}}(k+2)} = e^{-W\left(-\frac{q-1}{q^{k+2}}(k+2)\right)}.$$

Except for $k = 1$ and $q = 2$, for all other values of the parameters we have

$$-\frac{q-1}{q^{k+2}}(k+2) \ge -\frac{1}{e},$$

rendering the use of the $W$ function valid. We also note that for these parameters we have $\alpha \ge 1$.

Let us calculate $f(x)$,

$$\begin{aligned} f(x) &= q^{k+2}e^{-\frac{q-1}{q^{k+2}}(k+2)\alpha} - q^{k+2}e^{-\frac{q-1}{q^{k+2}}(k+1)\alpha} + q - 1 \\ &= q^{k+2}e^{-\frac{q-1}{q^{k+2}}(k+2)\alpha}\left(1 - e^{\frac{q-1}{q^{k+2}}\alpha}\right) + q - 1 \\ &\overset{(a)}{<} (q-1)\left(1 - \alpha e^{-\frac{q-1}{q^{k+2}}(k+2)\alpha}\right) \\ &\overset{(b)}{=} (q-1)(1-1) = 0, \end{aligned}$$

where (a) follows by an application of (8), and (b) follows by substituting the value of $\alpha$.

In summary, $f(x)$ is easily seen to be decreasing in the range $[1, (k+1)q/(k+2)]$, and increasing in the range $[(k+1)q/(k+2), \infty)$, and therefore, its unique largest root $\lambda$ is in the range

$$qe^{-\frac{q-1}{q^{k+2}}\alpha} \le \lambda \le qe^{-\frac{q-1}{q^{k+2}}}.$$

It is easy to verify that $\alpha = 1 + o(1)$, where $o(1)$ denotes a function decaying to 0 as $k \to \infty$. Hence,

$$\lambda = qe^{-\frac{q-1}{q^{k+2}}(1+o(1))},$$

and therefore

$$\begin{aligned} \mathsf{cap}(\mathrm{RLL}_q(0,k)) &= \log_2 \lambda \\ &= \log_2 q - \frac{(q-1)\log_2 e}{q^{k+2}}(1 + o(1)). \end{aligned}$$

## Acknowledgments

The authors would like to thank the associate editor and the anonymous reviewers for improving the presentation of this paper.

## References

[1] M. Arita and Y. Ohashi, "Secret signatures inside genomic DNA," *Biotechnol. Progr.*, vol. 20, no. 5, pp. 1605–1607, Jan. 2004.

[2] F. Balado, "Capacity of DNA data embedding under substitution mutations," *IEEE Trans. Inf. Theory*, vol. 59, no. 2, pp. 928–941, Feb. 2013.

[3] C. T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots," *Nature*, vol. 399, no. 6736, pp. 533–534, Jun. 1999.

[4] L. Dolecek and V. Anantharam, "Repetition error correcting sets: Explicit constructions and prefixing methods," *SIAM J. Discrete Math.*, vol. 23, no. 4, pp. 2120–2146, 2010.

[5] F. Farnoud, M. Schwartz, and J. Bruck, "A stochastic model for genomic interspersed duplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Jun. 2015, pp. 1731–1735.

[6] F. Farnoud, M. Schwartz, and J. Bruck, "The capacity of string-duplication systems," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 811–824, Feb. 2016.

[7] J. W. Fondon and H. R. Garner, "Molecular origins of rapid and continuous morphological evolution," *Nat. Acad. Sci.*, vol. 101, no. 52, pp. 18058–18063, Aug. 2004.

[8] D. Haughton and F. Balado, "BioCode: Two biologically compatible algorithms for embedding data in non-coding and coding regions of DNA," *BMC Bioinform.*, vol. 14, no. 1, p. 121, Apr. 2013.

[9] D. Heider and A. Barnekow, "DNA-based watermarks using the DNA-Crypt algorithm," *BMC Bioinform.*, vol. 8, no. 1, pp. 1–10, May 2007.

[10] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. 3rd ed. Englewood Cliffs, NJ, USA: Prentice Hall, 2004.

[11] K. A. S. Immink, *Coding Techniques for Digital Recorders*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.

[12] S. Jain, F. Farnoud, and J. Bruck, "Capacity and expressiveness of genomic tandem duplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Jun. 2015, pp. 1946–1950.
[13] D. C. Jupiter, T. A. Ficht, J. Samuel, Q.-M. Qin, and P. De Figueiredo, "DNA watermarking of infectious agents: Progress and prospects," *PLoS Pathog*, vol. 6, no. 6, p. e1000950, Jun. 2010.
[14] A. Kato and K. Zeger, "On the capacity of two-dimensional run-length constrained channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1527–1540, Jul. 1999.
[15] E. S. Lander *et al.*, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, no. 6822, pp. 860–921, 2001.
[16] P. Leupold, "Duplication roots," in *Proc. Int. Conf. Develop. Lang. Theory*, vol. 4588, pp. 290–299, Jul. 2007.
[17] P. Leupold, C. Martin-Vide, and V. Mitrana, "Uniformly bounded duplication languages," *Discrete Appl. Math.*, vol. 146, no. 3, pp. 301–310, Mar. 2005.
[18] D. Lind and B. H. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge, U.K.: Cambridge Univ. Press, 1985.
[19] M. Liss *et al.*, "Embedding permanent watermarks in synthetic genes," *PLoS ONE*, vol. 7, no. 8, p. e42465, Aug. 2012.
[20] N. I. Mundy and A. J. Helbig, "Origin and evolution of tandem repeats in the mitochondrial DNA control region of shrikes (*Lanius* spp.)," *J. Molecular Evol.*, vol. 59, no. 2, pp. 250–257, Aug. 2004.
[21] R. M. Roth and P. H. Siegel, "Lee-metric BCH codes and their application to constrained and partial-response channels," *IEEE Trans. Inf. Theory*, vol. 40, no. 4, pp. 1083–1096, Jul. 1994.
[22] K. Usdin, "The biological effects of simple tandem repeats: Lessons from the repeat expansion diseases," *Genome Res.*, vol. 18, no. 7, pp. 1011–1019, 2008.
[23] J. H. Van Lint and R. M. Wilson, *A Course in Combinatorics*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2001.
[24] P. C. Wong, K.-K. Wong, and H. Foote, "Organic data memory using the DNA approach," *Commun. ACM*, vol. 46, no. 1, pp. 95–98, Jan. 2003.
[25] N. Yachie, Y. Ohashi, and M. Tomita, "Stabilizing synthetic data in the DNA of living organisms," *Syst. Synth. Biol.*, vol. 2, nos. 1–2, pp. 19–25, Jun. 2008.
[26] S. M. H. T. Yazdi, H. M. Kiah, E. R. Garcia, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Trans. Molecular, Biol. Multi-Scale Commun.*, vol. 1, no. 3, pp. 230–248, Sep. 2015, [Online]. Available: http://arxiv.org/abs/1507.01611.

**Siddharth Jain** (S'15) is a Ph.D. Candidate in the department of Electrical Engineering at Caltech.

His research interests include information and coding theory, machine learning, information theoretic and statistical analysis of genomic data, pattern recognition, data compression and computational biology.

Siddharth received Bachelors and Masters degree from Indian Institute of Technology (IIT) Kanpur, India in 2013. He was awarded the proficiency medal at IIT Kanpur for excellent academic performance in Electrical Engineering.

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

He received his M.S. degree in Electrical and Computer Engineering from the University of Toronto in 2008. From the University of Illinois at Urbana-Champaign, he received his M.S. degree in mathematics and his Ph.D. in Electrical and Computer Engineering in 2012 and 2013, respectively. His research interests include the information-theoretic and probabilistic analysis of genomic evolutionary processes; rank aggregation and gene prioritization; and coding for flash memory and DNA storage.

Dr. Farnoud 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.

**Moshe Schwartz** (M'03–SM'10) is an associate professor at the Department 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 2010 IEEE Communications Society Best Student Paper Award in Signal Processing and Coding for Data Storage.

**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.