On the Reverse-Complement String-Duplication System

Eyar Ben-Tolila and Moshe Schwartz, *Senior Member, IEEE*

***Abstract*— Motivated by DNA storage in living organisms, and by known biological mutation processes, we study the reverse-complement string-duplication system. We fully classify the conditions under which the system has full expressiveness, for all alphabets and all fixed duplication lengths. We then focus on binary systems with duplication length 2 and prove that they have full capacity, yet surprisingly, have zero entropy-rate. Finally, by using binary single burst-insertion correcting codes, we construct codes that correct a single reverse-complement duplication of odd length, over any alphabet. The redundancy (in bits) of the constructed code does not depend on the alphabet size.**

***Index Terms*— String-duplication systems, capacity, entropy rate, error-correcting codes.**

I. Introduction

DNA is a very appealing medium for storing digital information, whose rate of creation is growing exponentially in recent years, causing an expanding gap between information production and data storage capabilities. Compared with current storage technologies, DNA offers densities that are higher in orders of magnitude, and thus can potentially serve as an extremely efficient storage system. Already, a density of $2.15 \cdot 10^{17}$ bytes per gram of DNA molecules has been demonstrated [12], whereas the densest commercially available option [1] is capable of storing only $1.86 \cdot 10^{11}$ bytes per gram of hardware.

Data can be stored in DNA, in vitro or in vivo. While the former will likely provide a higher density, the latter has many advantages. First, in-vivo DNA storage can serve as a protected medium for storing large amounts of data in a compact format for long periods of time [3], [39]. An additional advantage is that data can be disguised as part of the organisms' original DNA, thus providing an added layer of secrecy [8]. Finally, in-vivo DNA storage has further applications such as watermarking genetically modified organisms, and enabling synthetic biology methods [15], [25], [31].

However, storing information in living organisms introduces new types of errors. Among these new error types we find duplication errors, which are motivated by a class of mutations that are common in most organisms and lead to an abundance of repeated sequences in their genomes. Some examples are transposon-driven repeats [20], and tandem repeats, which are believed to be caused by slipped-strand mispairings [24]. These mutation processes take a substring of the DNA and insert a copy of it into the original string. The copy can be inserted anywhere in the string, and might even be reversed and complemented during the insertion [33]. A formal mathematical model for studying these kinds of mutation processes is the notion of string-duplication systems [13]. In such systems, a seed string evolves over time by successive applications of duplication functions.

Previous works studied several properties of these models, among which we mention capacity (the exponential growth rate of the number of mutated strings), entropy rate (the information generated by a stochastic mutation process), and expressiveness (the ability to create any possible target substring). It was shown in [13] that the system with end duplication, which copies a substring of a fixed length k to the end of the string, has full expressiveness and full capacity. In contrast, the system with a fixed length tandem duplication, where the copy is inserted next to its original position, has zero capacity and is never fully expressive, but when the length of the tandem duplication is only lower-bounded, full expressiveness and positive capacity are obtained. In [10], the exact entropy rate of both end- and tandem-duplication systems were found, in the case where the duplication size is $k = 1$ and the alphabet is binary. Furthermore, a noisy scenario was investigated where the duplicated bit has a probability of being complemented. In that case, the exact entropy rate of end-duplication systems was found, as well as bounds on the entropy rate of tandem-duplication systems. In [26], using stochastic approximation methods, upper bounds were found on the entropy rate of tandem duplication with a probability of substitutions.

To protect against these duplications, error-correcting codes for duplication channels were studied as well. A construction correcting any number of tandem duplications of a fixed length k was found in [17]. Other codes that correct a prescribed number of duplications were constructed in [18], [19], [21], [42], addressing several duplication types. Additional codes that are capable of correcting a mixture of tandem duplications and substitutions or edits were described in [34]–[36]. Finally, some Levenshtein-reconstruction problems for duplications were studied in [40], [41].

In this work we study reverse-complement duplication, in which the duplicated copy is reversed and complemented

Manuscript received 22 December 2021; revised 24 April 2022; accepted 7 June 2022. Date of publication 13 June 2022; date of current version 21 October 2022. This work was supported in part by the Israel Science Foundation (ISF) under Grant 270/18. *(Corresponding author: Moshe Schwartz.)*

The authors are with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer Sheva 8410501, Israel (e-mail: eyarb@post.bgu.ac.il; schwartz@ee.bgu.ac.il).

Communicated by A. Wachter-Zeh, Associate Editor for Coding Theory.
Digital Object Identifier 10.1109/TIT.2022.3182873

before being inserted next to its original location. This duplication process has been observed in the genomes of many living organisms [6], [9]. Previous works on this model were severely limited. In [13] the duplicated copy was reversed but not complemented. It was shown there that the system with a fixed-length duplication has full expressiveness and positive capacity. In [10], reverse-complement duplication was studied, but only for a duplication of length $k = 1$. Upper and lower bounds on the entropy rate of the system were found there, but a gap between them remains. Therefore, many aspects of this system are yet to be studied.

We make the following contributions in this paper. First, we fully classify the exact conditions under which the reverse-complement string-duplication system has full expressiveness, for any alphabet and any fixed duplication length. Next, we prove that the binary system with duplication length 2 has full capacity, by carefully characterizing the irreducible strings. We then continue to a probabilistic setting, and show that the same system has zero entropy rate, which is surprising considering the fact that it has full capacity. Thus, while nearly all strings are attainable via mutations, when the mutation process is random, the *probable* mutated outcomes are concentrated in a small set. Finally, we construct error-correcting codes that are capable of fixing a single duplication of odd length, over any alphabet. The coding scheme is built on burst-insertion-correcting codes, and interestingly, has redundancy (in bits) that does not depend on the alphabet size.

The paper is organized as follows. We begin in Section II, by introducing basic concepts and notation that will be used throughout the paper. Then, in Section III, we completely classify the expressive power of the reverse-complement string-duplication system. We continue in Section IV, by studying the capacity and entropy rate of these systems. Error-correcting codes are constructed in Section V. We conclude in Section VI by summarizing our results and discussing open problems.

II. Preliminaries

Throughout this paper we let Σ denote some finite alphabet, the elements of which are called letters. A sequence of letters is a string, e.g., $u = u_0u_1 \dots u_{n-1}$, where $u_i \in \Sigma$ for all i. In this case we say the length of u is n, and we denote it by $|u| = n$. The set of all strings of length n is denoted by Σ^n. The unique string of length 0 is denoted by ε, i.e., $\Sigma^0 = \{\varepsilon\}$. We then define Σ^* to be the set of all strings of finite length, i.e., $\Sigma^* \triangleq \bigcup_{i \geqslant 0} \Sigma^i$. Similarly, we define $\Sigma^+ \triangleq \Sigma^* \setminus \{\varepsilon\}$ to be the set of all non-empty strings of finite length.

Given two strings, $u \in \Sigma^n$ and $v \in \Sigma^m$, we write uv to denote the string of length $n+m$ formed by the concatenation of u and v. For any $\ell \in \mathbb{N}$, we write u^ℓ to denote the string of length ℓn formed by concatenating ℓ copies of u. We define $u^0 = \varepsilon$. We further define

$$u^* \triangleq \left\{u^\ell | \ell \geqslant 0\right\}, \qquad u^+ \triangleq \left\{u^\ell | \ell \geqslant 1\right\}.$$

As is commonly done, if $U \subseteq \Sigma^*$ is a set of strings, we define U^* to be the set of strings formed by finite concatenations of strings from U.

We say that $y \in \Sigma^*$ is a *prefix* of $w \in \Sigma^*$ if there exists $z \in \Sigma^*$ such that $w = yz$. Similarly, y is a *suffix* of w if there exists $x \in \Sigma^*$ such that $w = xy$. Also, y is a *factor* (or *substring*) of w if there exist $x, z \in \Sigma^*$ such that $w = xyz$. We shall say that y is a k-prefix (respectively, k-suffix, k-factor) of w, if it is a prefix (respectively, suffix, factor) of w, and $|y| = k$. If $S \subseteq \Sigma^*$ is a set of strings, its factor set is then defined as

$$\text{Factor}(S) \triangleq \{v \in \Sigma^* | \exists u, w \in \Sigma^* \text{ s.t. } uvw \in S\}.$$

We assume throughout the paper, that a *complement* operation is defined over Σ. More precisely, a complement operation is a bijective map which takes any $a \in \Sigma$ and outputs $\overline{a} \in \Sigma$, $\overline{a} \neq a$, and for which $\overline{\overline{a}} = a$. As a consequence, $|\Sigma|$ must be even, which we assume throughout the paper. We extend the complement operation to strings in the natural way. That is, if $u = u_0u_1 \dots u_{n-1} \in \Sigma^n$, then $\overline{u} = \overline{u_0}\,\overline{u_1} \dots \overline{u_{n-1}}$ is the letter-wise complement string.

Another useful notation we introduce is that of string reversal. Let $u = u_0u_1 \dots u_{n-1} \in \Sigma^n$ be a string of length n. The *reversal* of u is denoted by $u^R \triangleq u_{n-1}u_{n-2} \dots u_0$. Obviously, the reversal and complement operations are independent, and so $\overline{u}^R = \overline{u^R}$.

We now turn to describe the string-duplication framework. We follow the ideas and notation as described in [13]. A *string-duplication rule* is simply a function $T : \Sigma^* \to \Sigma^*$. Following the motivation for this framework, such rules describe, or are inspired by, biological processes that create duplications during DNA replication. The rules that were studied in [13] were tandem duplication, end duplication, interspersed duplication, and palindromic duplication. Let $\mathcal{T} \subseteq \Sigma^{*\Sigma^*}$ denote a set of such duplication rules. For a string $u \in \Sigma^*$, we say that v is an ℓ-descendant of u, if there exist $T_1, \dots, T_\ell \in \mathcal{T}$, not necessarily distinct, such that $v = T_\ell(T_{\ell-1}(\dots T_1(u)\dots))$, and we denote it by $u \Longrightarrow^\ell v$. If $\ell = 1$ we just write $u \Longrightarrow v$. Additionally, for $\ell = 0$ we only have $u \Longrightarrow^0 u$. The set of all ℓ-descendants of u is the set

$$D^\ell(u) \triangleq \left\{v \in \Sigma^* | u \Longrightarrow^\ell v\right\}.$$

The *descendant cone* of u is then defined as all the strings which may be derived from u following a finite number of duplication rules, namely,

$$D^*(u) \triangleq \bigcup_{\ell \geqslant 0} D^\ell(u).$$

We can then write $u \Longrightarrow^* v$ if and only if $v \in D^*(u)$.

A *string-duplication system*, $S(\Sigma, s, \mathcal{T})$, where Σ is a finite alphabet, $s \in \Sigma^*$ is the *seed string*, and $\mathcal{T} \subseteq \Sigma^{*\Sigma^*}$ is a set of duplication rules, is defined as the descendant cone of s, that is

$$S(\Sigma, s, \mathcal{T}) \triangleq D^*(s).$$

Thus, S contains all the strings that may result from s after applying a finite number of arbitrary duplication rules from $\mathcal{T}$.

The dual of the descendant cone is the *ancestor cone*. Here, the ancestor cone of $u \in \Sigma^*$ is defined as

$$A^*(u) \triangleq \{v \in \Sigma^* | u \in D^*(v)\},$$

namely, all the strings of which u is a descendant. We note that $u \in A^*(u)$ always. However, if $A^*(u) = \{u\}$, then we say that u is *irreducible*.

Fix some string-duplication system $S = S(\Sigma, s, \mathcal{T})$. Depending on the application, one can view S in two ways: either as a generative model which describes what possible strings may be derived from s (e.g., see [2], [7], [10], [13], [16], [26]), or as a channel which describes what corrupted versions of the transmitted s may be received (e.g., see [17]–[19], [21], [34], [36], [40]–[42]). While the latter view calls for the construction of suitably tailored error-correcting codes, the former inspires the following properties of S to be studied.

The first property of S is full expressiveness. We say that S is *fully expressive* if for any $v \in \Sigma^*$ there exist $u, w \in \Sigma^*$ (that may depend on v) such that $uvw \in S$. Thus, in a fully expressive system, any finite string appears as a factor of one of the strings in the system (see [13], [16]).

The second property of S is its capacity. We define the *capacity* of S to be

$$\mathsf{cap}(S) \triangleq \limsup_{n\to\infty} \frac{1}{n} \log_2 |S \cap \Sigma^n| .$$

Intuitively, the capacity of S measures the exponential growth rate of the descendant cone of the seed string. One can trivially see that $\mathsf{cap}(S) \leqslant \log_2|\Sigma|$, and if equality holds we say that S has full capacity. Additionally, as mentioned in [13], if S has full capacity then it must be fully expressive.

The last property of S is its entropy rate (see [10]). Unlike the deterministic nature of the previous two properties, here we describe a stochastic process. Denote $S(0) = s$, the seed string. Then, at each step $i = 1, 2, \ldots$, using some probability distribution (that may depend on i) over $\mathcal{T}$, we apply a randomly chosen duplication rule from $\mathcal{T}$ to $S(i-1)$, thus obtaining $S(i)$. Hence, $S(n)$, $n \in \mathbb{N}$, are all random variables. We can then define the entropy of $S(n)$ as

$$H(S(n)) \triangleq -\sum_{w\in\Sigma^*} \Pr(S(n) = w) \log_2 \Pr(S(n) = w).$$

With this, the *entropy rate* of the random process S is defined as

$$h(S) \triangleq \limsup_{n\to\infty} \frac{1}{n} H(S(n)).$$

Loosely speaking, $h(S)$ measures the amount of information generated by an application of a random duplication rule.

We conclude this section by describing the specific set of duplication rules we shall be studying in this paper. Let Σ be a finite alphabet with a complement operation defined on it. The reverse-complement duplication rule, that copies a k-factor starting in position i in the given string, is defined for all $x \in \Sigma^*$ as

$$T^{\mathrm{rc}}_{i,k}(x) \triangleq \begin{cases} uv\overline{v}^R w, & \text{if } x = uvw, |u| = i, |v| = k, \\ x, & \text{otherwise.} \end{cases}$$

We then define the set of duplication rules, for a fixed duplication length $k \in \mathbb{N}$ to be

$$\mathcal{T}^{\mathrm{rc}}_k \triangleq \left\{ T^{\mathrm{rc}}_{i,k} | i \geqslant 0 \right\}.$$

Finally, the *k-uniform reverse-complement string-duplication system* is defined as

$$S^{\mathrm{rc}}_k(s) \triangleq S(\Sigma, s, \mathcal{T}^{\mathrm{rc}}_k).$$

We note that in the notation S^{rc}_k, the dependence on Σ is implicit.

Example 1: Consider the alphabet $\Sigma = \mathbb{Z}_4 = \{0, 1, 2, 3\}$ with complement pairs defined by $\overline{0} = 1$ and $\overline{2} = 3$. Assume that the duplication length is $k = 2$. Then,

$$0123 \Longrightarrow 012\underline{30}3 \Longrightarrow 0123\underline{23}03,$$

where the duplicated factor is underlined. Thus, $01232303 \in S(\Sigma, 0123, \mathcal{T}^{\mathrm{rc}}_2)$.

We would like to note that the reverse-complement string-duplication system is indeed biologically motivated. DNA sequences are strings of bases (or nucleotides). Since there are four possible bases, adenine (A), thymine (T), guanine (G), and cytosine (C), we can think of DNA sequences as strings over $\Sigma = \{A, C, G, T\}$. The bases form two complementary pairs, $\overline{A} = T$, and $\overline{C} = G$. A reverse complement of a section of the DNA molecule might be inserted immediately following the said section, thus creating a *palindromic duplication* (e.g., see [6], [9]). The reverse-complement string-duplication system models this phenomenon. We mention in passing that [13] misused the term "palindromic duplication" to describe the insertion of a reversed, but not complemented, copy of a section of a DNA molecule. To avoid confusion with [13] and papers that cite it, we use the term "reverse-complement duplication".

III. Expressiveness

In this section we study the expressiveness of the k-uniform reverse-complement string-duplication system, $S^{\mathrm{rc}}_k(s)$. We show a simple sufficient and necessary condition on the seed string that implies the system is fully expressive when $k \geqslant 2$. The special case of $k = 1$ is also fully characterized.

We first study the case of $k \geqslant 2$. For any string $s \in \Sigma^*$, and any non-negative integer i, we say $a \in \Sigma$ is the *ith letter from the end of* s if $s = uav$, with $u \in \Sigma^*$ and $v \in \Sigma^i$. The main technical ingredient in proving the expressiveness of $S^{\mathrm{rc}}_k(s)$ is the following lemma, which shows that we can push letters towards the end of the string in a controlled manner.

Lemma 2: Let $k \in \mathbb{N}$, $k \geqslant 2$, $s \in \Sigma^*$, $|s| \geqslant k+1$, and let $i \geqslant 2$ be an integer. If the ith letter from the end of s is a, then there exists $s' \in D^2(s)$, such that a is the $(i-2)$nd letter from the end of s', where descendants are obtained using k-uniform reverse-complement duplications.

Proof: By the lemma's conditions, we can write $s = uvbcw$, where $u, w \in \Sigma^*$, $v \in \Sigma^{k-1}$, $b, c \in \Sigma$, $|w| \leqslant i-2$, and v contains the letter a that is ith from the end of s. Now,

$$s = uvbcw \Longrightarrow uvb\underline{\overline{b}\overline{v}^R}cw \Longrightarrow uv\overline{b}\overline{v}^R c\underline{\overline{c}v}w = s',$$

where for the reader's convenience we underlined the duplicated parts. The claim now follows immediately, since the letter a in v that was ith from the end in s, is $(i-2)$nd from the end of s'. ∎

Assume now that $s = s_0 s_1 \dots s_{n-1}$ is a string of length n, $s_i \in \Sigma$ for all i. For $j = 0, 1$ we define

$$\Sigma_j(s) \triangleq \{s_i | 0 \leqslant i \leqslant n-1, i \equiv j \pmod 2\}.$$

Thus, $\Sigma_0(s)$ (respectively, $\Sigma_1(s)$) is the set of letters of s that appear in even (respectively, odd) positions.

If $A \subseteq \Sigma$ is a subset of letters, we define

$$\overline{A} \triangleq \{\overline{a} | a \in A\}.$$

For $s \in \Sigma^*$ the following is a useful definition:

$$\delta(s) \triangleq \Sigma_0(s) \cup \Sigma_1(\overline{s}).$$

We observe that

$$\delta(\overline{s}) = \Sigma_0(\overline{s}) \cup \Sigma_1(s) = \overline{\delta(s)}. \tag{1}$$

We are now ready to prove the sufficient and necessary condition for $S_k^{\mathrm{rc}}(s)$ to have full expressiveness.

Theorem 3: Let $k \in \mathbb{N}$, $k \geqslant 2$, and let $s \in \Sigma^*$, $|s| \geqslant k$, be a seed string. Then $S_k^{\mathrm{rc}}(s)$ has full expressiveness if and only if:

1) k is odd, and $\delta(s) \cup \delta(\overline{s}) = \Sigma$.
2) k is even, and $\delta(s) = \Sigma$.

Proof: We first prove the "only if" part of the claim. Consider k odd, and assume that $\delta(s) \cup \delta(\overline{s}) \neq \Sigma$. It follows that there exists a letter $a \in \Sigma$ such that neither a, nor $\overline{a}$, appear in s. Hence, a and $\overline{a}$ cannot appear in any of the descendants of s, and $S_k^{\mathrm{rc}}(s)$ is not fully expressive. Now consider k even, and assume $\delta(s) \neq \Sigma$. Thus, there exists a letter $a \in \Sigma$ such that $\delta(s) \subseteq \Sigma \setminus \{a\}$, and by (1), also $\delta(\overline{s}) \subseteq \Sigma \setminus \{\overline{a}\}$. Consider now a k-factor of s that is being duplicated resulting in s', namely,

$$s = uvw \Longrightarrow uv\overline{v}^R w = s'.$$

Since k is even, we can easily see that

$$\Sigma_0(s') \subseteq \Sigma_0(s) \cup \Sigma_1(\overline{s}) = \delta(s) \subseteq \Sigma \setminus \{a\},$$
$$\Sigma_1(s') \subseteq \Sigma_0(\overline{s}) \cup \Sigma_1(s) = \delta(\overline{s}) \subseteq \Sigma \setminus \{\overline{a}\}.$$

Hence, once again we have $\delta(s') \subseteq \Sigma \setminus \{a\}$. By simple induction, this holds not only for s', but for any string in $D^*(s)$. Thus, aa is not a factor of any string in $D^*(s)$, and $S_k^{\mathrm{rc}}(s)$ is not fully expressive.

We move on to prove the "if" part. We now contend that for any descendant $u \in D^*(s)$, and any letter $a \in \Sigma$, we can find a descendant $u' \in D^*(u)$ such that the $(2i)$th letter from the end of u' is a, for some $i \in \mathbb{N}$ (namely, a is found in an even position from the end of the string). We distinguish between two cases depending on the parity of k.

Assume k is odd. The requirement that $\delta(s) \cup \delta(\overline{s}) = \Sigma$ implies that for any letter $a \in \Sigma$, we have that a or $\overline{a}$ appear in s. Trivially, any descendant of $u \in D^*(s)$ also satisfies $\delta(u) \cup \delta(\overline{u}) = \Sigma$, since no letter gets erased in the duplication process. Let $u \in D^*(s)$ be some descendant of s. We have the following cases:

1) If u contains the letter a as the $(2i)$th letter from the end, $i \in \mathbb{N}$, we are done by setting $u' = u$.
2) Otherwise, if a is the last letter of u, we perform two k-suffix reverse-complement duplications starting with u. The resulting u' has a as the $(2k)$th letter from the end.
3) Otherwise, if u contains a as the $(2i-1)$th letter from the end, perform a single k-suffix reverse-complement duplication to obtain u'. Now a is the $(2i-1+k)$th letter from the end of u', and $2i-1+k$ is even, since k is odd.
4) Otherwise, u does not contain a, but only $\overline{a}$. Perform a single reverse-complement duplication on a factor of u that contains $\overline{a}$ to obtain u''. Now u'' contains a, and we repeat the arguments from the first three cases to obtain the desired u'.

Assume k is even. Imagine the letters of s in even positions are colored red, and those in odd positions green. Furthermore, assume any letters inserted due to duplications are colored blue. Since k is even, one can easily see that in any descendant $u \in D^*(s)$, the red letters remain in even positions, and the green letters remain in odd positions. Hence, $\Sigma = \delta(s) \subseteq \delta(u)$, and so $\delta(u) = \Sigma$. We have the following cases:

1) If u contains the letter a as the $(2i)$th letter from the end, $i \in \mathbb{N}$, we are done by setting $u' = u$.
2) Otherwise, if the last letter of u is a, perform a single k-suffix reverse-complement duplication to obtain u'. Then the kth letter from the end of u' is a.
3) Otherwise, since $\delta(u) = \Sigma$, necessarily $\overline{a}$ is the $(2j-1)$st letter from the end, for some $j \in \mathbb{N}$. Perform a single reverse-complement duplication on a factor of u that contains $\overline{a}$ to obtain u'. Now u' contains a as the $(2i)$th letter from the end, for some $i \in \mathbb{N} \cup \{0\}$. If $i = 0$, i.e., a is the last letter of u', perform another k-suffix reverse-complement duplication so a is the kth letter from the end.

We have therefore proved our auxiliary claim, namely, that for any $u \in D^*(s)$ and any $a \in \Sigma$, there exists $u' \in D^*(u)$ such that a is the $(2i)$th letter from the end of u' for some $i \in \mathbb{N}$. We now claim that there exists $u'' \in D^*(u')$ such that u'' ends with a. This is accomplished by repeated application of Lemma 2, provided $|u'| \geqslant k+1$. If, $|u'| = k$, take $u'\overline{u'}^R u' \in D^2(u')$ and apply Lemma 2 repeatedly on it to obtain the desired u''. Note that in all cases we apply Lemma 2 at least once.

To complete the proof of the "if" part, assume we are given a string $a_0 a_1 \dots a_{\ell-1} \in \Sigma^\ell$. We intend to show that there exists a descendant of s whose ℓ-suffix is $a_0 \dots a_{\ell-1}$. By our previous discussion we can derive from s a string that ends with $a_{\ell-1}$,

$$s \Longrightarrow^* v a_{\ell-1}.$$

Since at least one application of Lemma 2 was used in the process, we necessarily have that $\delta(v) \cup \delta(\overline{v}) = \Sigma$ if k is odd, and $\delta(v) = \Sigma$ if k is even. We can therefore repeat the argument, starting with v, to derive a string that ends with $a_{\ell-2}$. In context of the entire derivation starting from s we obtain,

$$s \Longrightarrow^* v a_{\ell-1} \Longrightarrow^* v' a_{\ell-2} a_{\ell-1}.$$

By simple induction, we can repeat the process until

$$s \Longrightarrow^* v'' a_0 a_1 \dots a_{\ell-1},$$

thus completing the proof. ∎

When the duplication-window size is $k = 1$ we have a different situation. Here, full expressiveness depends solely on whether the alphabet is binary or not.

Theorem 4: Let $k = 1$, and let $s \in \Sigma^*$, $|s| \geqslant k$, be a seed string. Then $S_k^{\mathrm{rc}}(s)$ has full expressiveness if and only if $|\Sigma| = 2$.

Proof: In the first direction, assume $|\Sigma| = 2$, and w.l.o.g., suppose $\Sigma = \{0, 1\}$. It is trivial that, starting with 0, we can derive any binary string that starts with 01. First, repeatedly duplicate the last bit to obtain an alternating string $0101\ldots$. Then, extend any run (except for the initial 0) by duplicating the last bit of the preceding run, to obtain $01^{n_1}0^{n_2}1^{n_3}0^{n_4}\ldots$. If the seed string s contains a 0, we are done. Otherwise, duplicate a 1 bit from s to obtain a 0. Thus, $S_1^{\mathrm{rc}}(s)$ is fully expressive when $|\Sigma| = 2$.

For the other direction, assume $|\Sigma| \geqslant 4$. Denote the letters of the seed string by $s = s_0 s_1 \ldots s_{n-1}$, with $s_i \in \Sigma$. Since the size of the duplication window is $k = 1$, it follows that,

$$D^*(s) = \{u_0 u_1 \ldots u_{n-1} | u_i \in D^*(s_i)\}.$$

Furthermore, for all i,

$$D^*(s_i) \subseteq \{s_i, \overline{s_i}\}^*,$$

namely, the strings derived from the letter s_i may contain only the letters s_i and $\overline{s_i}$. Since $|\Sigma| \geqslant 4$, There exist $a, b \in \Sigma$ such that $\{a, \overline{a}\} \neq \{b, \overline{b}\}$. We contend that the string $v = (ab)^{n+1}$ is not a factor of any string in $D^*(s)$. Assume to the contrary that v is a factor of some string $w \in D^*(s)$. Since $|s| = n$, by the pigeonhole principal, it follows that ab or ba is a factor of $u_i \in D^*(s_i) \subseteq \{s_i, \overline{s_i}\}^*$, for some $0 \leqslant i \leqslant n-1$. This contradicts the fact that $\{a, \overline{a}\} \neq \{b, \overline{b}\}$. Hence, $S_1^{\mathrm{rc}}(s)$ is not fully expressive when $|\Sigma| \geqslant 4$. ∎

IV. Capacity and Entropy Rate

In this section we study the capacity and entropy rate of the reverse-complement string-duplication system. Loosely speaking, while the capacity focuses on what is possible [13], [16], the entropy rate looks at what is probable [10]. Thus, the capacity is an upper bound on the entropy rate (see [10]).

Both the capacity and the entropy rate are hard to find exactly, and in previous works it was found exactly only for a select few cases. We mention that the case of $k = 1$ was studied in [10] for a binary alphabet and a seed string $s = 0$. Trivially, when $k = 1$ we have full capacity, $\mathsf{cap}(S_1^{\mathrm{rc}}(0)) = 1$, and by an elaborate combinatorial counting argument it was shown there that $0.8689 \leqslant h(S_1^{\mathrm{rc}}(0)) \leqslant 0.9067$. However, the case $k = 1$ is degenerate in several ways, as reversal is meaningless, and duplication windows never overlap. Thus, a first true step into the reverse-complement string-duplication system requires taking a window size of $k = 2$ at least. We therefore restrict ourselves in this section to the binary alphabet $\Sigma = \mathbb{Z}_2$, a duplication length of $k = 2$, and a seed string $s = 00$. Surprisingly, we show that a huge gap exists in this case between the capacity and the entropy rate.

A. The Capacity

Our strategy for finding the exact capacity of the binary $S_2^{\mathrm{rc}}(00)$ differs from previous approaches to the problem in [13], [16]. We first characterize all the irreducible strings. We then show that we can derive all of them as factors from the seed string with a constant overhead. Finally, we show that this implies full capacity, i.e., $\mathsf{cap}(S_2^{\mathrm{rc}}(00)) = 1$.

In what follows, let us denote the set of irreducible strings with respect to the $S_2^{\mathrm{rc}}(s)$ system by Irr. That is,

$$\mathrm{Irr} \triangleq \{u \in \Sigma^* | A^*(u) = \{u\}\},$$

is the set of strings who are their own sole ancestor. In the following lemma, $(0^20^*1)^*$ plays an important role. We remind that $(0^20^*1)^*$ is the set of finite strings obtained by concatenating strings of the form 0^20^*1, which themselves, are a concatenation of at least two 0's, followed by a single 1. A symmetric statement holds for $(1^21^*0)^*$.

Lemma 5: The set of binary irreducible strings with respect to $S_2^{\mathrm{rc}}(s)$ satisfies

$$\mathrm{Irr} = \mathrm{Factor}\left((0^20^*1)^*\right) \cup \mathrm{Factor}\left((1^21^*0)^*\right).$$

Proof: The irreducible strings are exactly those strings that do not contain any reverse-complement duplication, namely, none of the factors 0011, 1100, 0101, and 1010. Using standard techniques from constrained coding [28], the set of binary strings not containing these factors is exactly generated by reading the labels on paths in the binary De Bruijn graph of order 3, where the four edges corresponding to 0011, 1100, 0101, and 1010, are removed, seen in Figure 1(a). By applying the Moore algorithm to minimize the number of vertices in the graph,[1] we obtain a more compact graph representation, seen in Figure 1(b). The set of strings generated by paths in the latter graph is immediately seen to be equal to $\mathrm{Factor}((0^20^*1)^*) \cup \mathrm{Factor}((1^21^*0)^*)$, as claimed. ∎

As our next step, we prove that we can derive any irreducible string from the seed string 00 using at most a constant overhead, which is independent of the irreducible string.

Lemma 6: Denote $S = S_2^{\mathrm{rc}}(00)$ over $\mathbb{Z}_2$. Then there exists a constant $c \in \mathbb{N}$ such that any irreducible $u \in \mathrm{Irr}$ is a factor of some $w \in S$ where $|w| \leqslant |u| + c$.

Proof: By Lemma 5, $u \in \mathrm{Irr}$ is either $u \in \mathrm{Factor}((0^20^*1)^*)$ or $u \in \mathrm{Factor}((1^21^*0)^*)$. Let us first consider the former, i.e., $u \in \mathrm{Factor}((0^20^*1)^*)$. Our goal is to find a derivation $00 \Longrightarrow^* w$ such that u is a factor of w, with only a constant overhead.

By possibly adding a prefix and a suffix totaling no more than 4 bits, we can find

$$u' = 0^{\ell_1}10^{\ell_2}1\ldots0^{\ell_m}1,$$

$\ell_i \geqslant 2$ for all i, such that u is a factor of u'. It then suffices to prove that we can derive $00 \Longrightarrow^* w$ such that u' is a factor of w. Let us further denote by t the number of indices i such that ℓ_i is even.

[1]The Moore algorithm creates an equivalent graph with a minimal number of vertices that correspond to the equivalence classes of the Myhill-Nerode Theorem (see [30, §3.9 and §3.10]). We do this minimization merely to get a compact form we can easily analyze.

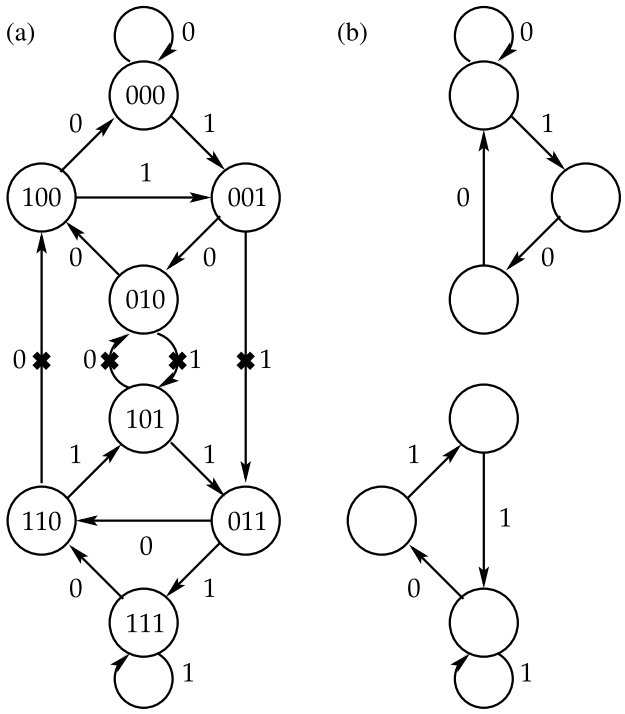

Fig. 1. The graphs used in the proof of Lemma 5: (a) The binary De Bruijn graph of order 3, with the edges corresponding to 0011, 0101, 1010, and 1100 canceled. Labels read along paths in the graph form exactly the set of binary strings not containing the factors 0011, 0101, 1010, and 1100. (b) The minimized graph of (a).

In preparation for finding the desired derivation, we start by deriving

$$00 \Longrightarrow^* 001^{2+2\lceil t/2\rceil} \Longrightarrow 001^{2+2\lceil t/2\rceil}00.$$

Thus, we obtain $001^{2+t}00$ if t is even, and $001^{3+t}00$ if t is odd. We denote

$$y \triangleq 001^{2+2\lceil t/2\rceil}, \qquad z \triangleq 00.$$

Hence, $00 \Longrightarrow^* yz$. We shall continue by induction, at each step updating the y part and the z, with our invariant being that the z part begins with 0 and that the y part ends with sufficiently many 1's (which will be made clear later).

Assume now that we would like to generate a factor of $0^\ell 1$, $\ell \geqslant 2$, from yz. Further assume $y = 001^i$. We distinguish between two cases, depending on the parity of ℓ:

1) If ℓ is even, assume $i \geqslant 3$, and then we repeatedly duplicate the penultimate 11 in y to obtain,

$$yz = 001^{i-3}111z \Longrightarrow^{\ell/2} 001^{i-1}0^\ell 1\ z = y'z'.$$

where $y' = 001^{i-1}$ and $z' = 0^\ell 1z$. We consider y' to be the "new" y, and similarly z', and we observe that y' is shorter than y since a single-bit suffix of 1 has been removed from it.

2) If ℓ is odd, assume $i \geqslant 2$. We make use of the fact that y ends in a 1, and z begins with a 0, and first duplicate this 10 factor. We then duplicate the suffix 11 of y repeatedly to obtain,

$$yz = 001^{i-2}11z \Longrightarrow 001^{i-2}1101z \Longrightarrow^{(\ell-1)/2} 001^i0^\ell 1\ z = y'z',$$

where $y' = y$ and $z' = 0^\ell 1z$. This time, the y prefix remains unchanged.

The process described above can be repeated, namely, we can generate $0^{\ell_m}1$, then $0^{\ell_{m-1}}1$, and so on, until $0^{\ell_1}1$. This is because at each step, the z part begins with a 0, and the y part ends with sufficiently many 1's. More precisely, each time ℓ_1 is even, a single-bit suffix of 1 is removed from the y part, and otherwise, the y part remains the same. Since we have t occurrences of even ℓ_i, and the initial y part has a suffix of $2+t$ 1's, the process terminates with a derivation

$$00 \Longrightarrow^* 001^{2+(t \bmod 2)}0^{\ell_1}1 \ldots 0^{\ell_m}100 \triangleq w.$$

Thus, w contains u' as a factor, with at most 7 extra bits in length. Together with the fact that u' contains u as a factor with at most 4 extra bits in length we get

$$|w| \leqslant |u| + 11.$$

Finally, we need to consider the case of $u \in \text{Factor}\ ((1^21^*0)^*)$. In this case, we first derive $00 \Longrightarrow 0011$. We ignore the 00 prefix, and repeat the proof from the previous case, only starting with 11 to obtain the desired derivation. In this case, we will get 2 more extra bits of overhead, namely,

$$|w| \leqslant |u| + 13.$$

■

We can now prove that the capacity is full.

Theorem 7: Over $\mathbb{Z}_2$,

$$\mathsf{cap}(S_2^{\mathrm{rc}}(00)) = 1.$$

Proof: Denote $S \triangleq S_2^{\mathrm{rc}}(00)$. We first note that all the strings in S have even length. Let $c \in \mathbb{N}$ be the constant from Lemma 6, and let $n \in \mathbb{N}$ be an even number, $n \geqslant c$. We contend that any $v \in \mathbb{Z}_2^{n-c}$ is a factor of some string in $S \cap \mathbb{Z}_2^n$. Indeed, let $u \in \mathrm{Irr}$ be a root of v, i.e., u is irreducible and $u \Longrightarrow^* v$. By Lemma 6, u is a factor of some string in $S \cap \mathbb{Z}_2^{|u|+c}$. Hence, there exist $y, z \in \mathbb{Z}_2^*$ such that $00 \Longrightarrow^* yuz$ and $|yz| = c$. It follows that

$$00 \Longrightarrow^* yuz \Longrightarrow^* yvz,$$

and $|yvz| = n$.

We have reached the conclusion that all 2^{n-c} strings in $\mathbb{Z}_2^{n-c}$ appear as factors of strings in $S \cap \mathbb{Z}_2^n$. Since each string of length n contributes at most $c+1$ distinct $(n-c)$-factors, we have

$$|S \cap \mathbb{Z}_2^n| \geqslant \frac{2^{n-c}}{c+1}. \tag{2}$$

Thus,

$$\mathsf{cap}(S) = \limsup_{n\to\infty} \frac{\log_2 |S \cap \mathbb{Z}_2^n|}{n} \geqslant \lim_{n\to\infty} \frac{\log_2 \frac{2^{n-c}}{c+1}}{n} = 1,$$

where the inequality follows from (2) and the fact that $|S \cap \mathbb{Z}_2^n| = 0$ for odd n. Since trivially $\mathsf{cap}(S) \leqslant 1$, we have $\mathsf{cap}(S) = 1$, as claimed. ■

B. Entropy Rate

Before diving into the technical details of finding the exact entropy rate $h(S_2^{\rm rc}(00))$ in the binary case, we would like to describe an outline of the strategy. Recall that the object under study here is a stochastic process, starting with the seed string 00. In each turn a position in the string is chosen, independently and uniformly, and a reverse-complement duplication of length 2 is performed there. The process then repeats, inducing a probability distribution over outcome strings after n rounds. The limit of the normalized entropy, as $n \to \infty$, gives us the entropy rate.

The approach we take is similar in spirit to [10], [26]: we shall track the frequencies of 2-factors as $n \to \infty$. Unlike previous papers, it seems to be impossible to track the frequencies exactly, and we shall have to resort to an approximate tracking only. Thus, while [10], [26] manage to extract differential equations that govern the evolution of factor frequencies, we shall have to settle for a differential *inclusion*. Once we have a handle on the limit of 2-factor frequencies, we shall use semiconstrained systems [11] to find their exact capacity, which we show is 0. Since the capacity upper bounds the entropy rate, we will reach the conclusion that $h(S_2^{\rm rc}(00)) = 0$.

Let us start the technical discussion. We follow [10], [13], and consider strings to be cyclic, namely, if $u = u_0 u_1 \ldots u_{n-1} \in \mathbb{Z}_2^n$, then u_{i+1} is the letter following u_i, where subscripts are taken modulo n. We shall find it more convenient to work with the derivative of strings rather than with the strings themselves. The *derivative* of u (see [14]) is defined as

$$D(u) \triangleq u_1 - u_0, u_2 - u_1, \ldots, u_{n-1} - u_{n-2}, u_0 - u_{n-1} \in \mathbb{Z}_2^n,$$

where the commas are written simply in order to separate the letters of the string. The derivative is easily seen to be a linear mapping which is two-to-one (mapping a string and its complement to the same derivative).

The stochastic process we are studying, $S = S_2^{\rm rc}(00)$, starts with the seed string $s = 00$, and we denote $S(0) = s$. For each $i = 1, 2, \ldots$, a position in $S(i-1)$ is chosen uniformly and independently at random, and a reverse-complement duplication of length $k = 2$ is performed on this position, resulting in $S(i)$. Thus, each $S(n)$, $n \in \mathbb{N}$, is a random variable representing the outcome of n reverse-complement duplications. Using the definition of the entropy of $S(n)$,

$$H(S(n)) \triangleq -\sum_{w \in \Sigma^{2n+2}} \Pr(S(n) = w) \log_2 \Pr(S(n) = w).$$

With this, the *entropy rate* of the random process S is defined as

$$h(S) \triangleq \limsup_{n \to \infty} \frac{1}{n} H(S(n)).$$

The probabilities $\Pr(S(n) = w)$ are hard to find, and thus, to compute $h(S)$ we resort to an indirect route. This involves statistics of factors of $S(n)$. To that end we give some more definitions. Let $u, v \in \Sigma^*$ be two strings, $|u| \geqslant |v|$. We use $|u|_v$ to denote the number of times v appears as a factor of u, including cyclically. More precisely, if the letters of u are $u = u_0 u_1 \ldots u_{n-1}$, then

$$|u|_v \triangleq \left|\{0 \leqslant i \leqslant n-1 | u_i u_{i+1} \ldots u_{i+k-1} = v\}\right|,$$

where the indices of u are taken modulo n. The frequency of v in u is then defined as

$$\mathrm{fr}_v(u) \triangleq \frac{|u|_v}{|u|}.$$

We can then see that for cyclic strings,

$$\mathrm{fr}_v(u) = \sum_{a \in \Sigma} \mathrm{fr}_{va}(u) = \sum_{a \in \Sigma} \mathrm{fr}_{av}(u).$$

Let us therefore define the set of ℓ-order admissible frequency vectors as

$$\mathcal{Q}(\Sigma^\ell) \triangleq \left\{ (\alpha^v)_{v \in \Sigma^\ell} \in [0,1]^{|\Sigma|^\ell} \;\middle|\; \sum_{v \in \Sigma^\ell} \alpha^v = 1, \forall v' \in \Sigma^{\ell-1} : \sum_{a \in \Sigma} \alpha^{v'a} = \sum_{a \in \Sigma} \alpha^{av'} \right\}.$$

Note that string indices, as the v in α^v, are written in superscript. We also comment that the α's in the vector $(\alpha^v)_{v \in \Sigma^\ell}$ are indexed in lexicographic order. For example, $(\alpha^{00}, \alpha^{01}, \alpha^{10}, \alpha^{11}) \in \mathcal{Q}(\mathbb{Z}_2^2)$ if and only if $\alpha^v \in [0,1]$ for all $v \in \mathbb{Z}_2^2$, $\alpha^{00} + \alpha^{01} + \alpha^{10} + \alpha^{11} = 1$, and also $\alpha^{00} + \alpha^{01} = \alpha^{00} + \alpha^{10}$, as well as $\alpha^{11} + \alpha^{10} = \alpha^{11} + \alpha^{01}$. We will also use partial vectors of $\mathcal{Q}(\Sigma^\ell)$ by replacing entries with a dot. For example, the statement $(\alpha^{00}, \alpha^{01}, \alpha^{10}, \cdot) \in \mathcal{Q}(\mathbb{Z}_2^2)$ is equivalent to stating that there exists α^{11} such that $(\alpha^{00}, \alpha^{01}, \alpha^{10}, \alpha^{11}) \in \mathcal{Q}(\mathbb{Z}_2^2)$.

Our analysis of the stochastic process uses the framework of stochastic approximation [5]. The fundamental tool we employ is that of the differential inclusion limit [5, Sec. 5], and whose main theorem we now recall.

Theorem 8 (Differential Inclusion Limit [5]): Let z_n be a discrete stochastic process in $\mathbb{R}^d$ given by

$$z_{n+1} = z_n + a_n (y_n + M_{n+1}), \quad n \geqslant 0,$$

with a given z_0. Assume all the following conditions hold:

(A1) $y_n \in f(z_n)$ for all $n \geqslant 0$, where f is a set-valued map $f : \mathbb{R}^d \to \mathcal{P}(\mathbb{R}^d)$ (i.e., a mapping of vectors from $\mathbb{R}^d$ to subsets of vectors from $\mathbb{R}^d$) that satisfies:
  (i) For each $z \in \mathbb{R}^d$, $f(z)$ is convex and compact.
  (ii) For all $z \in \mathbb{R}^d$,
  $$\sup_{y \in f(z)} \|y\| < K(1 + \|z\|)$$
  for some constant $K > 0$.
  (iii) f is upper semicontinuous in the sense that if $z_n \to z$, $y_n \to y$, $y_n \in f(z_n)$, for $n \geqslant 1$, then $y \in f(z)$.

(A2) $a_n > 0$ for all $n \geqslant 0$, are fixed positive scalars satisfying
$$\sum_n a_n = \infty, \qquad \sum_n a_n^2 < \infty.$$

(A3) $\{M_n\}$ is a martingale difference sequence w.r.t. the increasing σ-algebras $\mathcal{F}_n = \sigma(z_m, y_m, M_m, m \leqslant n)$, $n \geqslant 0$, i.e.,
$$\mathbb{E}[M_{n+1} | \mathcal{F}_n] = 0,$$
almost surely, $n \geqslant 0$. Additionally, $\{M_n\}$ are square-integrable with
$$\mathbb{E}[\|M_{n+1}\| | \mathcal{F}_n] \leqslant K' \left(1 + \|z_n\|^2\right),$$
almost surely, $n \geqslant 0$, for some constant $K' > 0$.

(A4) The sequence is bounded,

$$\sup_n \|z_n\| < \infty,$$

almost surely.

Then z_n converges almost surely to a closed connected internally chain transitive invariant set of the differential inclusion limit

$$\frac{\mathrm{d}}{\mathrm{d}t} z(t) \in f(z(t)). \tag{3}$$

The following theorem does the heavy lifting in this section, proving that the derivative of $S(n)$ is asymptotically, almost surely, composed nearly entirely of 1's.

Theorem 9: Let $S = S_2^{\mathrm{rc}}(00)$ be the stochastic process described above. Then, almost surely,

$$\lim_{n\to\infty} \mathrm{fr}_1\left(D(S(n))\right) = 1.$$

Proof: While our goal is to prove a claim on the frequency of 1's, we shall need to track the frequencies of other factors as well. More precisely, we shall follow how the frequencies of 0's, 10's, and 11's evolve in $D(S(n))$ as $n \to \infty$. Let us therefore define

$$x_n \triangleq \begin{pmatrix} |D(S(n))|_0 \\ |D(S(n))|_{10} \\ |D(S(n))|_{11} \end{pmatrix},$$
$$z_n \triangleq \begin{pmatrix} \mathrm{fr}_0(D(S(n))) \\ \mathrm{fr}_{10}(D(S(n))) \\ \mathrm{fr}_{11}(D(S(n))) \end{pmatrix} = \frac{1}{2n+2} x_n,$$

where the last equality is due to the fact that $|S(n)| = |D(S(n))| = 2n+2$.

We now write

$$x_{n+1} = x_n + \xi_{n+1},$$

which we can rewrite in the following form:

$$z_{n+1} = z_n + \frac{1}{2n+4}\left(\xi_{n+1} - 2z_n\right). \tag{4}$$

The value of ξ_{n+1} depends on the position of the reverse-complement duplication taken from $S(n)$ to $S(n+1)$. To find the possible values of ξ_{n+1}, consider a string $u \in \mathbb{Z}_2^*$ whose letters are u_i. If a reverse-complement duplication is performed on the ith position then

$$u = \ldots u_i u_{i+1} u_{i+2} \ldots \Longrightarrow \ldots u_i u_{i+1} \overline{u_{i+1} u_i} u_{i+2} \ldots.$$

In the derivative domain this becomes,

$$\begin{aligned} &\ldots u_{i+1} - u_i, u_{i+2} - u_{i+1}, u_{i+3} - u_{i+2} \ldots \longrightarrow \\ &\ldots u_{i+1} - u_i, 1, u_i - u_{i+1}, u_{i+2} - \overline{u_i}, u_{i+3} - u_{i+2} \ldots, \end{aligned} \tag{5}$$

where $\longrightarrow$ is used instead of $\Longrightarrow$ to emphasize that this is a reverse-complement duplication in the derivative domain. Tabulating all the possible cases of (5) gives us Table I.

We further manipulate (4) to obtain,

$$\begin{aligned} z_{n+1} = z_n + \frac{1}{2n+4}(&\mathbb{E}[\xi_{n+1}|\mathcal{F}_n] - 2z_n \\ &+ \xi_{n+1} - \mathbb{E}[\xi_{n+1}|\mathcal{F}_n]), \end{aligned}$$

where $\mathcal{F}_n$ is the σ-algebra generated by $\sigma(z_m, \xi_m, m \leqslant n)$. We first observe that

$$M_{n+1} \triangleq \xi_{n+1} - \mathbb{E}[\xi_{n+1}|\mathcal{F}_n]$$

is a martingale difference sequence w.r.t. $\mathcal{F}_n$ since

$$\begin{aligned} \mathbb{E}[M_{n+1}|\mathcal{F}_n] &= \mathbb{E}[\xi_{n+1} - \mathbb{E}[\xi_{n+1}|\mathcal{F}_n]|\mathcal{F}_n] \\ &= \mathbb{E}[\xi_{n+1}|\mathcal{F}_n] - \mathbb{E}[\xi_{n+1}|\mathcal{F}_n] = 0. \end{aligned}$$

Next, we study

$$y_n \triangleq \mathbb{E}[\xi_{n+1}|\mathcal{F}_n] - 2z_n. \tag{6}$$

Determining $\mathbb{E}[\xi_{n+1}|\mathcal{F}_n]$ seems difficult. However, if we denote[2]

$$\begin{aligned} \alpha_n^{000} &\triangleq \mathbb{E}[\mathrm{fr}_{000}(D(S(n)))|\mathcal{F}_n], \\ \alpha_n^{001} &\triangleq \mathbb{E}[\mathrm{fr}_{001}(D(S(n)))|\mathcal{F}_n], \\ \alpha_n^{010} &\triangleq \mathbb{E}[\mathrm{fr}_{010}(D(S(n)))|\mathcal{F}_n], \\ \alpha_n^{011} &\triangleq \mathbb{E}[\mathrm{fr}_{011}(D(S(n)))|\mathcal{F}_n], \end{aligned}$$

then by using Table I we could write,

$$\begin{aligned} \mathbb{E}[\xi_{n+1}|\mathcal{F}_n] &= \alpha_n^{000}\begin{pmatrix} 0 \\ +2 \\ 0 \end{pmatrix} + \alpha_n^{001}\begin{pmatrix} 0 \\ +1 \\ +1 \end{pmatrix} \\ &\quad + \alpha_n^{010}\begin{pmatrix} +2 \\ 0 \\ 0 \end{pmatrix} + \alpha_n^{011}\begin{pmatrix} +2 \\ +1 \\ -1 \end{pmatrix} \\ &\quad + (1 - \alpha_n^{000} - \alpha_n^{001} - \alpha_n^{010} - \alpha_n^{011})\begin{pmatrix} 0 \\ 0 \\ +2 \end{pmatrix} \\ &= \begin{pmatrix} 2\alpha_n^{010} + 2\alpha_n^{011} \\ 2\alpha_n^{000} + \alpha_n^{001} + \alpha_n^{011} \\ 2 - 2\alpha_n^{000} - \alpha_n^{001} - 2\alpha_n^{010} - 3\alpha_n^{011} \end{pmatrix}. \end{aligned}$$

We make the observation that, given $z_n = (z_n^0, z_n^{10}, z_n^{11})^{\mathsf{T}}$,

$$\alpha_n^{000} + \alpha_n^{001} + \alpha_n^{010} + \alpha_n^{011} = z_n^0. \tag{7}$$

Looking back at (6), we can therefore write,

$$\begin{aligned} y_n &= \begin{pmatrix} 2\alpha_n^{010} + 2\alpha_n^{011} \\ 2\alpha_n^{000} + \alpha_n^{001} + \alpha_n^{011} \\ 2 - 2\alpha_n^{000} - \alpha_n^{001} - 2\alpha_n^{010} - 3\alpha_n^{011} \end{pmatrix} - 2\begin{pmatrix} z_n^0 \\ z_n^{10} \\ z_n^{11} \end{pmatrix} \\ &= \begin{pmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ -2 & 0 & -2 \end{pmatrix}\begin{pmatrix} z_n^0 \\ z_n^{10} \\ z_n^{11} \end{pmatrix} + \begin{pmatrix} 2\alpha_n^{010} + 2\alpha_n^{011} \\ 2\alpha_n^{000} + \alpha_n^{001} + \alpha_n^{011} \\ 2 + \alpha_n^{001} - \alpha_n^{011} \end{pmatrix}. \end{aligned}$$

Let us now define the function $f : \mathbb{R}^3 \to \mathcal{P}(\mathbb{R}^3)$, sending vectors from $\mathbb{R}^3$ to subsets of $\mathbb{R}^3$, in the following way,

$$\begin{aligned} f\begin{pmatrix} z^0 \\ z^{10} \\ z^{11} \end{pmatrix} \triangleq \Bigg\{ &\begin{pmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ -2 & 0 & -2 \end{pmatrix}\begin{pmatrix} z^0 \\ z^{10} \\ z^{11} \end{pmatrix} \\ &+ \begin{pmatrix} 2\alpha^{010} + 2\alpha^{011} \\ 2\alpha^{000} + \alpha^{001} + \alpha^{011} \\ 2 + \alpha^{001} - \alpha^{011} \end{pmatrix} \Bigg| \\ &(\alpha^{000}, \alpha^{001}, \alpha^{010}, \alpha^{011}, \cdot, \cdot, \cdot, \cdot) \in \mathcal{Q}(\mathbb{Z}_2^3) \Bigg\}. \end{aligned}$$

[2]We use superscripts here to remind us of the relevant factors, and not to represent powers.

TABLE I

THE CASES OF (5), SHOWING THE RELEVANT FACTORS OF THE DERIVATIVE BEFORE AND AFTER THE REVERSE-COMPLEMENT DUPLICATION, AS WELL AS THE VECTOR ξ_{n+1} REPRESENTING THE CHANGE IN THE NUMBER OF OCCURRENCES OF THE FACTORS 0, 10, AND 11.

before dup.	000	001	010	011	100	101	110	111
after dup.	01010	01011	01000	01001	11100	11101	11110	11111
ξ_{n+1}	$\begin{pmatrix}0\\+2\\0\end{pmatrix}$	$\begin{pmatrix}0\\+1\\+1\end{pmatrix}$	$\begin{pmatrix}+2\\0\\0\end{pmatrix}$	$\begin{pmatrix}+2\\+1\\-1\end{pmatrix}$	$\begin{pmatrix}0\\0\\+2\end{pmatrix}$	$\begin{pmatrix}0\\0\\+2\end{pmatrix}$	$\begin{pmatrix}0\\0\\+2\end{pmatrix}$	$\begin{pmatrix}0\\0\\+2\end{pmatrix}$

It then follows that y_n from (6) satisfies,

$$y_n \in f(z_n).$$

By a simple check we can verify that the requirements of Theorem 8 are satisfied. Thus, by employing it we can say that z_n converges almost surely to the limit of a function satisfying (3). In our case, that means solving

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}z^0\\z^{10}\\z^{11}\end{pmatrix} = \begin{pmatrix}-2&0&0\\0&-2&0\\-2&0&-2\end{pmatrix}\begin{pmatrix}z^0\\z^{10}\\z^{11}\end{pmatrix} + \begin{pmatrix}2\alpha^{010}+2\alpha^{011}\\2\alpha^{000}+\alpha^{001}+\alpha^{011}\\2+\alpha^{001}-\alpha^{011}\end{pmatrix}$$

for some constants $(\alpha^{000},\alpha^{001},\alpha^{010},\alpha^{011},\cdot,\cdot,\cdot,\cdot) \in \mathcal{Q}(\mathbb{Z}_2^3)$, and with the initial condition

$$z^0(0)=1, \qquad z^{10}(0)=0, \qquad z^{11}(0)=0,$$

since the seed string is 00. The solution is then

$$\begin{aligned}z^0(t) &= (\alpha^{010}+\alpha^{011})(1-e^{-2t})+e^{-2t},\\ z^{10}(t) &= \left(\alpha^{000}+\frac{1}{2}\alpha^{001}+\frac{1}{2}\alpha^{011}\right)(1-e^{-2t}),\\ z^{11}(t) &= \left(1+\frac{1}{2}\alpha^{001}-\alpha^{010}-\frac{3}{2}\alpha^{011}\right)(1-e^{-2t})\\ &\quad + e^{-2t}\cdot 4t(\alpha^{010}+\alpha^{011}-1).\end{aligned}$$

Thus, almost surely,

$$\begin{aligned}\lim_{n\to\infty}\begin{pmatrix}\mathrm{fr}_0(D(S(n)))\\ \mathrm{fr}_{10}(D(S(n)))\\ \mathrm{fr}_{11}(D(S(n)))\end{pmatrix} &= \lim_{t\to\infty}\begin{pmatrix}z^0(t)\\z^{10}(t)\\z^{11}(t)\end{pmatrix}\\ &= \begin{pmatrix}\alpha^{010}+\alpha^{011}\\ \alpha^{000}+\frac{1}{2}\alpha^{001}+\frac{1}{2}\alpha^{011}\\ 1+\frac{1}{2}\alpha^{001}-\alpha^{010}-\frac{3}{2}\alpha^{011}\end{pmatrix} \triangleq \begin{pmatrix}z^0_\infty\\z^{10}_\infty\\z^{11}_\infty\end{pmatrix}.\end{aligned} \tag{8}$$

Since for any cyclic binary string u we have

$$\mathrm{fr}_0(u)+\mathrm{fr}_{10}(u)+\mathrm{fr}_{11}(u)=1,$$

this relation also holds in the limit, and therefore,

$$z^0_\infty + z^{10}_\infty + z^{11}_\infty = 1.$$

Substituting the values from (8) we thus have

$$\alpha^{000}+\alpha^{001}=0.$$

Since $\alpha^{000},\alpha^{001}\in[0,1]$, we reach the conclusion that

$$\alpha^{000}=\alpha^{001}=0,$$

and we obtain

$$\begin{pmatrix}z^0_\infty\\z^{10}_\infty\\z^{11}_\infty\end{pmatrix} = \begin{pmatrix}\alpha^{010}+\alpha^{011}\\ \frac{1}{2}\alpha^{011}\\ 1-\alpha^{010}-\frac{3}{2}\alpha^{011}\end{pmatrix}. \tag{9}$$

The information in (9) is not yet sufficient to prove the claim. The main technical obstacle is that Theorem 8 does not completely determine all the parameters. However, by reusing it in a slightly different manner we can improve our result. We go all the way back to our observation (7). We now make another observation, which is that the frequencies of 01 and 10 in a cyclic binary string must be equal. Hence,

$$\begin{aligned}\alpha^{010}_n+\alpha^{011}_n &= \mathbb{E}[\mathrm{fr}_{010}(D(S(n)))|\mathcal{F}_n]\\ &\quad + \mathbb{E}[\mathrm{fr}_{011}(D(S(n)))|\mathcal{F}_n]\\ &= \mathbb{E}[\mathrm{fr}_{01}(D(S(n)))|\mathcal{F}_n]\\ &= \mathbb{E}[\mathrm{fr}_{10}(D(S(n)))|\mathcal{F}_n] = z^{10}_n.\end{aligned}$$

Continuing down the same path as before, we now have,

$$\begin{aligned}y_n &= \begin{pmatrix}2\alpha^{010}_n+2\alpha^{011}_n\\ 2\alpha^{000}_n+\alpha^{001}_n+\alpha^{011}_n\\ 2-2\alpha^{000}_n-\alpha^{001}_n-2\alpha^{010}_n-3\alpha^{011}_n\end{pmatrix} - 2\begin{pmatrix}z^0_n\\z^{10}_n\\z^{11}_n\end{pmatrix}\\ &= \begin{pmatrix}-2&2&0\\1&-2&0\\-2&0&-2\end{pmatrix}\begin{pmatrix}z^0_n\\z^{10}_n\\z^{11}_n\end{pmatrix} + \begin{pmatrix}0\\ \alpha^{000}_n-\alpha^{010}_n\\ 2+\alpha^{001}_n-\alpha^{011}_n\end{pmatrix},\end{aligned}$$

and by defining

$$\begin{aligned}g\begin{pmatrix}z^0\\z^{10}\\z^{11}\end{pmatrix} \triangleq \Bigg\{ &\begin{pmatrix}-2&2&0\\1&-2&0\\-2&0&-2\end{pmatrix}\begin{pmatrix}z^0\\z^{10}\\z^{11}\end{pmatrix}\\ &+\begin{pmatrix}0\\ \alpha^{000}-\alpha^{010}\\ 2+\alpha^{001}-\alpha^{011}\end{pmatrix}\ \Bigg|\\ &(\alpha^{000},\alpha^{001},\alpha^{010},\alpha^{011},\cdot,\cdot,\cdot,\cdot)\in\mathcal{Q}(\mathbb{Z}_2^3)\Bigg\},\end{aligned}$$

it then follows that y_n from (6) satisfies,

$$y_n \in g(z_n).$$

Once again we use Theorem 8, however this time we obtain that almost surely,

$$\begin{pmatrix}z^0_\infty\\z^{10}_\infty\\z^{11}_\infty\end{pmatrix} = \begin{pmatrix}\beta^{000}-\beta^{010}\\ \beta^{000}-\beta^{010}\\ 1-\beta^{000}+\frac{1}{2}\beta^{001}+\beta^{010}-\frac{1}{2}\beta^{011}\end{pmatrix}, \tag{10}$$

for some $(\beta^{000},\beta^{001},\beta^{010},\beta^{011},\cdot,\cdot,\cdot,\cdot)\in\mathcal{Q}(\mathbb{Z}_2^3)$. We crucially note that the first two components of (10) are equal.

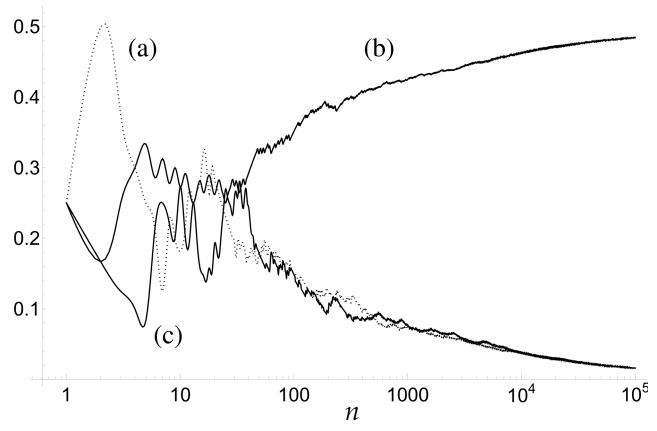

Fig. 2. An example simulation of $S_2^{\mathrm{rc}}(00)$ showing (a) $\mathrm{fr}_{00}(S(n))$, (b) $\mathrm{fr}_{01}(S(n)) = \mathrm{fr}_{10}(S(n))$, and (c) $\mathrm{fr}_{11}(S(n))$.

Carrying this knowledge back to (9) we obtain the equation

$$\alpha^{010} + \alpha^{011} = \frac{1}{2}\alpha^{011}.$$

Since $\alpha^{010}, \alpha^{011} \in [0,1]$, we necessarily have

$$\alpha^{010} = \alpha^{011} = 0.$$

Putting this back into (9), we reach the conclusion that, almost surely,

$$\begin{pmatrix} z_\infty^0 \\ z_\infty^{10} \\ z_\infty^{11} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

thus proving our claim. ∎

Returning from the derivative domain is simple.

Corollary 10: Let $S = S_2^{\mathrm{rc}}(00)$ be the stochastic process described above. Then, almost surely,

$$\lim_{n\to\infty} \begin{pmatrix} \mathrm{fr}_{00}(S(n)) \\ \mathrm{fr}_{01}(S(n)) \\ \mathrm{fr}_{10}(S(n)) \\ \mathrm{fr}_{11}(S(n)) \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}.$$

Proof: In a cyclic binary string, the frequencies of 01 and 10 are equal. Additionally, they are the only factors creating 1's in the derivative. Thus, the conclusion follows from Theorem 9. ∎

An illustration of the frequencies of 2-factors in $S(n)$, as $n \to \infty$, is shown in Figure 2. Notice in this figure how $\mathrm{fr}_{01}(S(n)) = \mathrm{fr}_{10}(S(n))$ tends to $\frac{1}{2}$, hence in the derivative, $\mathrm{fr}_1(D(S(n))$ tends to 1. We now conclude this section by proving that the entropy rate of $S_2^{\mathrm{rc}}(00)$ is 0. We recall the following useful bound [26, Th. 11], and the remark following it.

Lemma 11 ([26]): Let S be a stochastic duplication system, as defined above. Assume that

$$\lim_{n\to\infty} \left(\mathrm{fr}_v(S(n))\right)_{v\in\Sigma^\ell} = (\alpha^v)_{v\in\Sigma^\ell},$$

almost surely. Then,

$$h(S) \leqslant -\sum_{v\in\Sigma^\ell} \alpha^v \log_2\left(\frac{\alpha^v}{\mu^v}\right),$$

where μ^v is the marginal on the first $\ell-1$ coordinates, namely, if $v = v_0 v_1 \ldots v_{\ell-1}$, with $v_i \in \Sigma$ for all i, then

$$\mu^v = \sum_{a\in\Sigma} \alpha^{v_0\ldots v_{\ell-2}a}.$$

Corollary 12: Over $\mathbb{Z}_2$,

$$h(S_2^{\mathrm{rc}}(00)) = 0.$$

Proof: Simply combine Lemma 11 with Corollary 10 to obtain an upper bound of 0 on the entropy rate. A lower bound of 0 is trivial. ∎

V. Single-Duplication-Correcting Codes

In this section we switch gears, and consider the reverse-complement duplication as a source of noise. To protect the information against such a noise mechanism, we would like to design efficient error-correcting codes. We first formally define error-correcting codes for the reverse-complement duplication channel. We then show how to construct single-duplication-correcting codes, with odd duplication length, by altering known code constructions for single-burst-insertion correction.

Definition 13: An $(n, M, t)_k^{\mathrm{rc}}$ reverse-complement-duplication code is a set $C \subseteq \Sigma^n$ of size $|C| = M$, such that for every $c_1 \neq c_2 \in C$, $D^t(c_1) \cap D^t(c_2) = \emptyset$, all with respect to the reverse-complement string-duplication system, with duplication length k. The redundancy of the code (in bits) is defined as $\log_2(|\Sigma|^n/|C|)$.

Thus, in a code capable of correcting t reverse-complement duplications of length k, no two distinct codewords have the same t-descendant. Our focus here is on single-error-correcting codes, namely, $t = 1$.

Trivially a reverse-complement duplication is a special case of an insertion. Hence, a simple "off-the-shelf" solution to finding such a code is to employ a general insertion-correcting code. The first code solving this problem in the literature is the renowned Varshamov-Tenengolts (VT) code [22], [38]. It is a binary code which addresses the case of a single insertion or deletion, i.e., $k = 1$. The redundancy of the binary VT code is $\log_2 n + o(1)$. This code was extended to a q-ary alphabet in [37], with a code of redundancy $\log_2 n + \log_2 q + o(1)$. The binary VT codes were also extended to binary codes capable of correcting (general) k insertions or deletions. For example, recently, [32] (see also references therein) constructed binary k-deletion-correcting codes with redundancy $8k \log_2 n + o(\log n)$.

Taking another step forward, we observe that a single reverse-complement duplication does not insert k symbols in random positions, but rather as a single burst of length k. Thus, to correct a single reverse-complement duplication we may use a ready-made code capable of correcting a single burst of insertions of length k. A binary code that corrects a single burst of insertions or deletions of length $k = 2$, was first constructed in [23]. This code has redundancy $\log_2(n) + 1 + o(1)$. Recently, [29] (see also references therein) constructed binary codes that correct a single burst of length (general) k insertions or deletions, with redundancy at most $\log_2 n + (k-1)\log_2 \log_2 n + k - \log_2 k$.

Thus, in light of the above, the advantage of the construction we are about to propose is two-fold. First, except for the degenerate case of burst-length 1 (i.e., a single inserted symbol), the known constructions are only over the binary alphabet. Our construction works for arbitrary alphabets. Second, the redundancy of our construction does not depend on the size of the alphabet.

To present our construction we require the notion of a complement-preserving mapping.

Definition 14: Let Σ be any alphabet of even size, with a complement operation as defined in Section II. A mapping $\beta : \Sigma \to \mathbb{Z}_2$ is said to be *complement preserving* if for all $a \in \Sigma$,

$$\beta(\overline{a}) = \overline{\beta(a)}.$$

Complement-preserving mappings always exist, and may be chosen to be easily computable. For example, if $\Sigma = \mathbb{Z}_q$, with $\overline{2i} = 2i + 1$ for all i, then we may take $\beta(a) = a \bmod 2$. We also extend β to act on strings in the natural way, namely, $\beta(a_0 a_1 \ldots a_{n-1}) \triangleq \beta(a_0)\beta(a_1)\ldots\beta(a_{n-1})$, with $a_i \in \Sigma$ for all i. We can now give our construction.

Construction A: Let $C' \subseteq \mathbb{Z}_2^n$ be a binary code capable of correcting a single burst insertion of odd length k. Let Σ be an alphabet of even size, and $\beta : \Sigma \to \mathbb{Z}_2$ a complement-preserving mapping. We construct the following code:

$$C \triangleq \{c \in \Sigma^n | \beta(c) \in C'\}.$$

To prove that this construction is indeed capable of correcting a single reverse-complement duplication, one might expect the following straightforward decoding algorithm: upon receiving a string $w \in \Sigma^{n+k}$, the receiver computes $\beta(w)$ and then it runs the decoding algorithm for the binary burst-insertion-correcting code C'. The receiver then finds out what k-factor is to be removed from w to obtain the transmitted codeword. This however is insufficient, as the following example shows. Suppose $\Sigma = \mathbb{Z}_4$ (with $\overline{0} = 1$ and $\overline{2} = 3$), and $k = 3$. Consider the received string $w = 1332202221$, for which we have $\beta(w) = 1110000001$. Assume further that the decoder of C' determines a 000 factor was inserted, and is therefore to be removed. Unfortunately, there are four possible factors 000 in $\beta(w)$, the removal of any of which results in the same string $\beta(c) = 1110001$. However, in the string w over $\mathbb{Z}_4$ these four factors are 220, 202, 022 and 222. It is not immediately clear which of them is to be removed to obtain the transmitted c. We solve this problem by using the fact that the inserted burst forms a reverse-complement duplication. In this case, only 220 forms a reverse-complement duplication since it is preceded by 133.

We cite the useful Lyndon-Schützenberger Lemma, and then proceed to prove the correctness of the code construction. For the following, recall that ε denotes the unique empty word.

Lemma 15 ([27, Lemma 2]): Let $x, y, z \in \Sigma^*$, $x \neq \varepsilon$. If $xy = yz$, then there exist $u, v \in \Sigma^*$ and $\ell \geqslant 0$ such that $x = uv$, $y = (uv)^\ell u$, and $z = vu$.

Lemma 16: Let Σ be any alphabet of even size. Consider the reverse-complement string-duplication system over Σ, with duplication length k that is odd. If $w' \implies w$ (i.e., w is obtained from w' using a single reverse-complement duplication operation), then there is only a single way of obtaining w from w' using a reverse-complement duplication of length k.

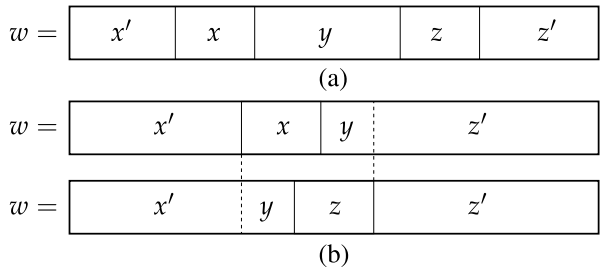

Fig. 3. The two cases of w in the proof of Theorem 17.

Proof: Throughout the proof, we adopt the notation that if $a \in \Sigma^n$ is a string, its letters are denoted $a_0 a_1 \ldots a_{n-1}$. Assume to the contrary that two k-factors of $w \in \Sigma^{n+k}$ in distinct positions, denoted $x, z \in \Sigma^k$, may be removed to obtain $w' \in \Sigma^n$. We contend that with the additional knowledge that both x and z form a reverse-complement duplication, a contradiction must be reached. We distinguish between two cases, depending on whether the positions of x and z overlap, as depicted in Figure 3.

Case I: Assume x and z do not overlap (see Figure 3(a)). Since the removal of x and the removal of z result in w', we necessarily have $xy = yz$. By Lemma 15 there exist u, $v \in \Sigma^*$ and an integer $\ell \geqslant 0$ such that

$$x = uv, \qquad y = (uv)^\ell u, \qquad z = vu. \tag{11}$$

According to (11), the k-factor preceding z in w is vu. Since z is a reverse-complement duplication of length k, its preceding k-factor must be the reverse complement of z. Thus, by (11) we get,

$$z = vu = (\overline{vu})^R = \overline{z}^R.$$

Since $|z| = k$ is odd, we have an odd-length string that is equal to its own reverse complement. The middle letter of that string, z_i, $i = (k-1)/2$, must therefore satisfy $z_i = \overline{z_i}$, a contradiction.

Case II: Assume that x and z overlap (see Figure 3(b)). In this case, by Lemma 15 we must have $x = uv$, $y = u$, and $z = vu$. Since x is a reverse-complement duplication, it is preceded by its reverse complement. Thus, the following is a factor of w:

$$\overbrace{\overline{v}^R\,\overline{u}^R}^{\overline{x}^R}\ \overbrace{u\,\underbrace{v}_{z}\,u}^{x}$$

Since k is odd, and $k = |u| + |v|$, we have that $|u| - |v|$ is odd, and in particular, $|u| \neq |v|$. If $|u| = 0$ then x and z are in the exact same location, which contradicts our assumption that they are not. If $|v| = 0$ then x and z do not overlap. Hence, in what follows we assume $|u|, |v| > 0$. We now use the fact that z is also a reverse-complement duplication, and hence, must be preceded by its reverse complement. We divide our discussion depending on the relation between $|u|$ and $|v|$.

1) $0 < |v| < |u|$: In order for z to be a reverse-complement duplication the following equations must hold,

$$v_0 v_1 \dots v_{|v|-1} = \overline{u_{|u|-1} u_{|u|-2} \dots u_{|u|-|v|}}, \quad (12)$$
$$u_0 u_1 \dots u_{|u|-|v|-1} = \overline{u_{|u|-|v|-1} u_{|u|-|v|-2} \dots u_0}. \quad (13)$$

Recall that $|u| - |v|$ is odd. Thus, (13) shows a string of odd length that equals its reverse complement. In particular, for its middle letter we get $u_i = \overline{u_i}$, $i = (|u| - |v| - 1)/2$, a contradiction.

2) $0 < |u| < |v| < 2|u|$: For z to form a reverse-complement duplication, the following equations must hold,

$$v_0 v_1 \dots v_{|u|-1} = \overline{u_{|u|-1} u_{|u|-2} \dots u_0}, \quad (14)$$
$$v_{|u|} v_{|u|+1} \dots v_{|v|-1} = u_0 u_1 \dots u_{|v|-|u|-1}, \quad (15)$$
$$u_0 u_1 \dots u_{2|u|-|v|-1} = u_{|v|-|u|} \dots u_{|u|-1}, \quad (16)$$
$$u_{2|u|-|v|} \dots u_{|u|-1} = v_0\ v_1 \dots v_{|v|-|u|-1}. \quad (17)$$

From (14) and (17) we get that

$$u_{2|u|-|v|} \dots u_{|u|-1} = \overline{u_{|u|-1} u_{|u|-2} \dots u_{2|u|-|v|}}.$$

Once again we have a string of odd length that is equal to its own reverse complement, a contradiction.

3) $|v| = 2|u|$: This time for z to form a reverse-complement duplication we must have,

$$v_0 v_1 \dots v_{|u|-1} = \overline{u_{|u|-1} u_{|u|-2} \dots u_0}, \quad (18)$$
$$v_{|u|} v_{|u|+1} \dots v_{|v|-1} = u_0 u_1 \dots u_{|u|-1}, \quad (19)$$
$$u_0 u_1 \dots u_{|u|-1} = v_0 v_1 \dots v_{|u|-1}. \quad (20)$$

Recall that $|u| - |v|$ is odd, so since $|v|$ is even in this case, we have that $|u|$ is odd. From (18) and (20) it follows that $u = \overline{u}^R$, which is again a contradiction since u has odd length.

4) $2|u| < |v|$: For z to form a reverse-complement duplication we necessarily have,

$$v_0 v_1 \dots v_{|u|-1} = \overline{u_{|u|-1} u_{|u|-2} \dots u_0}, \quad (21)$$
$$v_{|u|} v_{|u|+1} \dots v_{2|u|-1} = u_0 u_1 \dots u_{|u|-1}, \quad (22)$$
$$v_{2|u|} v_{2|u|+1} \dots v_{|v|-1} = v_0 v_1 \dots v_{|v|-2|u|-1}, \quad (23)$$
$$u_0 u_1 \dots u_{|u|-1} = v_{|v|-2|u|} \dots v_{|v|-|u|-1}. \quad (24)$$

By repeated applications of (23) we may reduce the indices of v modulo $2|u|$. Denote by $I \triangleq \{|v| - 2|u|, |v| - 2|u| + 1, \dots, |v| - |u| - 1\}$ the $|u|$ consecutive indices of v appearing on the right-hand side of (24). They may be reduced modulo $2|u|$ to obtain $I \bmod 2|u|$. We contend that the following intersection

$$J = (I \bmod 2|u|) \cap \{0, 1, \dots, |u| - 1\}$$

is not empty. If this intersection were empty, then necessarily $I \bmod 2|u| = \{|u|, |u + 1|, \dots, 2|u| - 1\}$. But that would mean that

$$|v| - 2|u| \equiv |u| \pmod{2|u|},$$

and therefore

$$|v| - |u| \equiv 0 \pmod{2|u|},$$

which implies that $|v| - |u|$ is even, a contradiction. Denote

$$s \triangleq (|v| - 2|u|) \bmod 2|u|,$$
$$t \triangleq (|v| - |u|) \bmod 2|u|.$$

Thus, either $J = \{0, 1, \dots, t - 1\}$ or $J = \{s, s + 1, \dots, |u| - 1\}$.
Assume the former case holds, i.e., $J = \{0, 1, \dots, t-1\}$. Observe that

$$|J| = t \equiv |v| - |u| \equiv 1 \pmod 2.$$

Then, using only the indices in J with (21) and (24) we have

$$u_{|u|-t} \dots u_{|u|-1} = v_0 \dots v_{t-1} = \overline{u_{|u|-1} \dots u_{|u|-t}}.$$

We thus obtained a string of odd length that equals its reverse complement, a contradiction.
If we assume the latter case holds, i.e., $J = \{s, s + 1, \dots, |u| - 1\}$, then

$$|J| = |u| - s \equiv |u| - |v| \equiv 1 \pmod 2.$$

Then, using only the indices in J with (21) and (24) we have

$$u_0 \dots u_{|u|-s-1} = v_s \dots v_{|u|-1} = \overline{u_{|u|-s-1} \dots u_0}.$$

Once again, we obtained a string of odd length that equals its reverse complement, a contradiction.

After having considered all cases and reaching a contradiction in all of them, we are forced to deduce that no x and z factors, in distinct positions, are possible. That means that there is a unique k-factor in w' whose reverse-complement duplication results in w. ■

Theorem 17: The code C from Construction A can correct a single reverse-complement duplication of length k.

Proof: Assume that a codeword of C was transmitted, and a single reverse-complement duplication of length k occurred which resulted in $w \in \Sigma^{n+k}$. We then compute $\beta(w) \in \mathbb{Z}_2^{n+k}$ and use the decoding procedure of C' on it. This decoding procedure identifies at least one k-factor of $\beta(w)$ whose removal results in a codeword from C'. However, by Lemma 16, this factor is unique. Hence, removing the corresponding factor from w produces the correct transmitted codeword. ■

Using Theorem 17, we may use known binary codes that are capable of correcting a single burst insertion of odd length k, as the component code C' in Construction A. The resulting q-ary codes are capable of correcting a single reverse-complement duplication of length k. For example, taking C' to be the binary VT code [22], [38], we can construct a q-ary $(n, M, 1)_1^{\mathrm{rc}}$ code with redundancy $\log_2 n + o(1)$ for *any* even q. This is better than using the q-ary VT code [37] with redundancy $\log_2 n + \log_2 q + o(1)$. For odd $k \geqslant 3$ we can use the binary burst-insertion-correcting code of [29] to create a q-ary $(n, M, 1)_1^{\mathrm{rc}}$ code for *any* even q, with redundancy at most $\log_2 n + (k - 1) \log_2 \log_2 n + k - \log_2 k$, which does not depend on q. If we are willing to increase the redundancy slightly, while gaining some flexibility with the duplication

length, we can use the codes of [4] as the basis for our construction. These binary codes of length n, can correct any deletions or insertions that are confined to a range of K positions, with redundancy $\log n+O(K\log^2(K\log n))$. Thus, as long as the actual duplication length, k, is odd and does not exceed K, we will be able to correctly decode. We also mention that the encoding complexity of our code construction is exactly that of the underlying binary code. Hence, if for example we use the binary code of [4], we can encode in time $O(n\cdot\text{poly}(K\log n))$.

By using Lemma 16 and a standard ball-packing argument, we can also prove a simple lower bound on the redundancy.

Theorem 18: Let C be an $(n,M,1)_k^{\text{rc}}$, k odd, over an alphabet Σ of size q. Then,

$$M\leqslant\frac{q^{n+k}}{n-k+1},$$

namely, the code has redundancy at least $\log_2(n-k+1)-k\log_2 q$.

Proof: By Lemma 16, from each codeword $c\in C$ there are exactly $n-k+1$ distinct reverse-complement duplications, namely $|D^1(c)|=n-k+1$. By the definition of a code, if $c'\in C$, $c'\neq c$, then $D^1(c)\cap D^1(c')=\emptyset$. Since all the mutated strings are of length $n+k$, by simple ball packing we obtain $|C|=M\leqslant q^{n+k}/(n-k+1)$, as well as the claimed redundancy bound. ∎

As we can see, the bound on the code redundancy from Theorem 18 is lower than the redundancy we can currently obtain using our construction. However, the main gap is due to the gap between the redundancy of the binary burst-insertion-correcting code, and the lower bound on the redundancy of such codes [23].

To conclude this section, we would like to point out a peculiarity of codes that correct reverse-complement duplications. It is well known that codes that correct insertions can also correct the same amount of deletions, even when burst insertions/deletions are concerned (e.g., see [22], [29]). If insertion-correcting codes correspond to duplication-correcting codes, then deletion-correcting codes correspond to deduplication-correcting codes, which we now define.

Definition 19: An $(n,M,t)_k^{\text{rc}}$ reverse-complement-deduplication code is a set $C\subseteq\Sigma^n$ of size $|C|=M$, such that for every $c_1\neq c_2\in C$, $A^t(c_1)\cap A^t(c_2)=\emptyset$, all with respect to the reverse-complement string-duplication system, with duplication length k.

Hence, deduplication-correcting codes have non-intersecting t-ancestors, whereas duplication-correcting codes have non-intersecting t-descendants.

The peculiarity we would like to point out is that duplication-correcting codes are not necessarily deduplication-correcting codes, and vice versa. The following counter examples show this.

Example 20: Consider the following code,

$$C=\{00110,00011\}\subseteq\mathbb{Z}_2^5.$$

This is a $(5,2,1)_2^{\text{rc}}$ duplication-correcting code since,

$$D^1(00110)=\{0011110,0010110,0011000,0011010\},$$
$$D^1(00011)=\{0011011,0001111,0001011,0001100\},$$

hence, $D^1(00110)\cap D^1(00011)=\emptyset$. However, C is *not* a $(5,2,1)_2^{\text{rc}}$ deduplication-correcting code since

$$000\in A^1(00110)\cap A^1(00011).$$

Example 21: Consider the following code,

$$C=\{1100,1111\}\subseteq\mathbb{Z}_2^4.$$

This is a $(4,2,1)_2^{\text{rc}}$ deduplication-correcting code since,

$$A^1(1100)=\{11\},$$
$$A^1(1111)=\emptyset,$$

hence, $A^1(00110)\cap A^1(00011)=\emptyset$. Observe that $A^1(1111)=\emptyset$ since 1111 is irreducible. We now also note that C is *not* a $(4,2,1)_2^{\text{rc}}$ duplication-correcting code since

$$110011\in D^1(00110)\cap D^1(00011).$$

VI. Conclusion

In this paper, we studied the reverse-complement string-duplication system. When viewed as a generative system, we fully classified the cases in which $S_k^{\text{rc}}(s)$ has full expressiveness. Interestingly, these differ depending on whether k is even or odd, and with $k=1$ being a special case altogether. We then focused on the binary case with $k=2$ and a seed string 00. We proved that the capacity in this case is full, i.e., $\mathsf{cap}(S_2^{\text{rc}}(00))=1$, but the entropy rate vanishes, i.e., $h(S_2^{\text{rc}}(00))=0$. We switched gears to look at the duplications as a noise source, and constructed q-ary codes that correct a single reverse-complement duplication from binary burst-insertion-correcting codes. The construction works for odd duplication lengths, and produces codes whose redundancy (in bits) does not depend on the alphabet size.

We find the fact that $h(S_2^{\text{rc}}(00))=0$ particularly surprising. When looking at previously known results [10], we have $\mathsf{cap}(S_1^{\text{rc}}(0))=1$ and $0.8689\leqslant h(S_1^{\text{rc}}(0))\leqslant 0.9067$. In stark contrast, when the duplication length is increased from $k=1$ to $k=2$ we still have $\mathsf{cap}(S_2^{\text{rc}}(00))=1$, but $h(S_2^{\text{rc}}(00))=0$. To the best of our knowledge, this is the first case of a duplication system that is fully expressive, with full capacity, but vanishing entropy rate. Why that is the case when $k=2$, and whether it remains so for larger values of k, is still unknown. Additionally, from Figure 2 it appears as if the convergence to the limit is quite slow.

Many open questions remain, and we mention a few. First, we observe that our proof that the binary $S_2^{\text{rc}}(00)$ has full capacity is tailored to this case, and relies on the classification of irreducible strings. The use of irreducible strings to prove full capacity is new, and we suspect that the same approach may work for any alphabet size, and any duplication length. However, the classification of irreducible strings in each case becomes increasingly complex. It would be interesting to find a more general approach to the problem of determining $\mathsf{cap}(S_k^{\text{rc}}(s))$ over any alphabet. A similar open question pertains to the problem of finding the entropy rate $h(S_k^{\text{rc}}(s))$ for general alphabets. The stochastic approximation method we used calls for a decision on what factors to follow. It is still unknown how to generalize this to cases other than $S_2^{\text{rc}}(00)$.

Finally, the construction for duplication-correcting codes that we provided only works for odd duplication lengths. It is interesting to find a construction for even k, and to find out whether it offers the same savings in redundancy compared with burst-insertion-correcting codes. In addition, the construction only works for a single duplication, which parallels the fact that only single-burst-insertion-correcting codes are known. These problems, and others, are left for future work.

REFERENCES

[1] *Nimbus Data ExaDrive DC Datasheet*. Accessed: 2020. [Online]. Available: https://nimbusdata.com/docs/ExaDrive-DC-Datasheet.pdf

[2] N. Alon, J. Bruck, F. F. Hassanzadeh, and S. Jain, "Duplication distance to the root for binary sequences," *IEEE Trans. Inf. Theory*, vol. 63, no. 12, pp. 7793–7803, Dec. 2017.

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

[4] R. Bitar, S. K. Hanna, N. Polyanskii, and I. Vorobyev, "Optimal codes correcting localized deletions," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Melbourne, VIC, Australia, Jul. 2021, pp. 1991–1996.

[5] V. Borkar, *Stochastic Approximation—A Dynamical System Viewpoint*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[6] D. K. Butler, D. Gillespie, and B. Steele, "Formation of large palindromic DNA by homologous recombination of short inverted repeat sequences in Saccharomyces cerevisiae," *Genetics*, vol. 161, no. 3, pp. 1065–1075, Jul. 2002.

[7] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Deciding the confusability of words under tandem repeats in linear time," *ACM Trans. Algorithms*, vol. 15, no. 3, pp. 42:1–42:22, 2019.

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

[9] S. J. Diede, H. Tanaka, D. A. Bergstrom, M.-C. Yao, and S. J. Tapscott, "Genome-wide analysis of palindrome formation," *Nature Genet.*, vol. 42, no. 4, p. 279, 2010.

[10] O. Elishco, F. F. Hassanzadeh, M. Schwartz, and J. Bruck, "The entropy rate of some Pólya string models," *IEEE Trans. Inf. Theory*, vol. 65, no. 12, pp. 8180–8193, Dec. 2019.

[11] O. Elishco, T. Meyerovitch, and M. Schwartz, "Semiconstrained systems," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 811–824, Apr. 2016.

[12] Y. Erlich and D. Zielinski, "DNA Fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, 2017.

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

[14] S. W. Golomb, *Shift Register Sequences*. San Francisco, CA, USA: Holden-Day, 1967.

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

[16] S. Jain, F. F. Hassanzadeh, and J. Bruck, "Capacity and expressiveness of genomic tandem duplication," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6129–6138, Oct. 2017.

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

[18] M. Kovačević, "Zero-error capacity of duplication channels," *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 6735–6742, Oct. 2019.

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

[20] E. S. Lander *et al.*, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, no. 6822, pp. 860–921, 2001.

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

[22] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals (in Russian)," *Doklady Akademii Nauk SSR*, vol. 163, no. 4, pp. 845–848, 1965.

[23] V. I. Levenshtein, "Asymptotically optimum binary code with correction for losses of one or two adjacent bits," (in Russian), *Problemy Kibernetiki*, vol. 19, pp. 293–298, 1967.

[24] G. Levinson and G. A. Gutman, "Slipped-strand mispairing: A major mechanism for DNA sequence evolution," *Molecular Biol. Evol.*, vol. 4, no. 3, pp. 203–221, 1987.

[25] M. Liss *et al.*, "Embedding permanent watermarks in synthetic genes," *PLoS ONE*, vol. 7, no. 8, Aug. 2012, Art. no. e42465.

[26] H. Lou, M. Schwartz, J. Bruck, and F. Farnoud, "Evolution of k-mer frequencies and entropy in duplication and substitution mutation systems," *IEEE Trans. Inf. Theory*, vol. 66, no. 5, pp. 3171–3186, May 2020.

[27] R. C. Lyndon and M. P. Schützenberger, "The equation $a^M = b^N c^P$ in a free group," *Michigan Math. J.*, vol. 9, no. 4, pp. 289–298, 1962.

[28] B. H. Marcus, R. M. Roth, and P. H. Siegel, *An Introduction to Coding for Constrained Systems* (Lecture Notes). The Univ. of British Columbia, Oct. 2001. [Online]. Available: https://www.math.ubc.ca/~marcus/Handbook

[29] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 1971–1985, Apr. 2017.

[30] J. Shallit, *A Second Course in Formal Languages and Automata Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[31] S. L. Shipman, J. Nivala, J. D. Macklis, and G. M. Church, "CRISPR-Cas encoding of digital movie into the genomes of a population of living bacteria," *Nature*, vol. 547, pp. 345–349, Jul. 2017.

[32] J. Sima and J. Bruck, "On optimal k-deletion correcting codes," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 3360–3375, Jun. 2021.

[33] G. R. Smith, "Meeting DNA palindromes head-to-head," *Genes Develop.*, vol. 22, no. 19, pp. 2612–2620, Oct. 2008.

[34] Y. Tang and F. Farnoud, "Error-correcting codes for noisy duplication channels," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 3452–3463, Jun. 2021.

[35] Y. Tang and F. Farnoud, "Error-correcting codes for short tandem duplication and edit errors," *IEEE Trans. Inf. Theory*, vol. 68, no. 2, pp. 871–880, Feb. 2022.

[36] Y. Tang, Y. Yehezkeally, M. Schwartz, and F. Farnoud, "Single-error detection and correction for duplication and substitution channels," *IEEE Trans. Inf. Theory*, vol. 66, no. 11, pp. 6908–6919, Nov. 2020.

[37] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 5, pp. 766–769, Sep. 1984.

[38] R. R. Varshamov and G. M. Tenengolts, "Code which correct single asymmetric errors," (in Russian), *Avtomatika i Telemekhanika*, vol. 26, no. 2, pp. 288–292, 1965.

[39] 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.

[40] Y. Yehezkeally and M. Schwartz, "Reconstruction codes for DNA sequences with uniform tandem-duplication errors," *IEEE Trans. Inf. Theory*, vol. 66, no. 5, pp. 2658–2668, May 2020.

[41] Y. Yehezkeally and M. Schwartz, "Uncertainty of reconstruction with list-decoding from uniform-tandem-duplication noise," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4276–4287, Jul. 2021.

[42] M. Zeraatpisheh, M. Esmaeili, and T. A. Gulliver, "Construction of tandem duplication correcting codes," *IET Commun.*, vol. 13, no. 15, pp. 2217–2225, Sep. 2019.

Eyar Ben-Tolila received the B.Sc. degree in mathematics, the B.Sc. degree in electrical engineering, and the M.Sc. degree in electrical engineering from the Ben-Gurion University of the Negev, Israel. He was a Graduate Student with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev. His research interests include coding for DNA storage, information theory, and deep learning.

Moshe Schwartz (Senior Member, IEEE) received the B.A. *(summa cum laude)*, M.Sc., and Ph.D. degrees from the Computer Science Department, Technion—Israel Institute of Technology, Haifa, Israel, in 1997, 1998, and 2004 respectively.

He was a Fulbright Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, University of California San Diego, and a Post-Doctoral Researcher with 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). He is currently a Professor with 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 2009 IEEE Communications Society Best Paper Award in Signal Processing and Coding for Data Storage and the 2020 NVMW Persistent Impact Prize. He served as an Associate Editor of coding techniques and coding theory for the IEEE TRANSACTIONS ON INFORMATION THEORY from 2014 to 2021, and since 2021, he has been serving as an Area Editor of coding and decoding for the IEEE TRANSACTIONS ON INFORMATION THEORY. He has been an Editorial Board Member for the *Journal of Combinatorial Theory, Series A* since 2021.