Low-redundancy codes for correcting multiple short-duplication and edit errors

Yuanyuan Tang, *Student Member, IEEE*, Shuche Wang, *Student Member, IEEE*, Hao Lou, *Student Member, IEEE*, Ryan Gabrys, *Member, IEEE*, and Farzad Farnoud, *Member, IEEE*

***Abstract*—Due to its higher data density, longevity, energy efficiency, and ease of generating copies, DNA is considered a promising technology for satisfying future storage needs. However, a diverse set of errors including deletions, insertions, duplications, and substitutions may arise in DNA at different stages of data storage and retrieval. The current paper constructs error-correcting codes for simultaneously correcting short (tandem) duplications and at most p edits, where a short duplication generates a copy of a substring with length ≤ 3 and inserts the copy following the original substring, and an edit is a substitution, deletion, or insertion. Compared to the state-of-the-art codes for duplications only, the proposed codes correct up to p edits (in addition to duplications) at the additional cost of roughly $8p(\log_q n)(1+o(1))$ symbols of redundancy, thus achieving the same asymptotic rate, where $q \geq 4$ is the alphabet size and p is a constant. Furthermore, the time complexities of both the encoding and decoding processes are polynomial when p is a constant with respect to the code length.**

***Index Terms*—DNA data storage, error-correcting codes, short tandem duplications, edit errors, redundancy, time complexity.**

I. Introduction

With recent advances in sequencing and synthesis, deoxyribonucleic acid (DNA) is considered a promising candidate for satisfying future data storage needs [3], [4]. In particular, experiments in [3], [5]–[9] demonstrate that data can be stored on and subsequently retrieved from DNA. Compared to traditional data storage media, DNA has the advantages of higher data density, longevity, energy efficiency, and ease of generating copies [3], [9]. However, a diverse set of errors may occur at different stages of the data storage and retrieval processes, such as deletions, insertions, duplications, and substitutions. Many recent works, such as [9]–[26], have been devoted to protecting the data against these errors. The current paper constructs error-correcting codes for duplication and edit errors, where an edit error is an insertion, deletion, or substitution.

A *(tandem) duplication* in a DNA sequence generates a copy of a substring and then inserts it directly following the original substring [10], where the duplication length is the length of the copy. For example, given ACTG, a tandem duplication may generate ACTCTG, where CTCT is a *(tandem) repeat* of length 4 (i.e., twice the length of the duplication). Bounded-length duplications are those whose lengths are at most a given constant. In particular, we refer to duplications of length at most 3 as *short duplications*. Correcting fixed-length duplications [10], [12]–[14], [27] and bounded-length duplications [10], [25], [28]–[31] have both been studied recently. In particular, the code in [10], which has a polynomial-time encoder, provides the highest possible asymptotic rate for correcting any number of short duplications [29]. For an alphabet of size 4, corresponding to DNA data storage, this rate is $\log 2.6590$ and as the alphabet size q increases, the rate is approximately $\log(q-1)$ [32].

For channels with both duplication and substitution errors, *restricted* substitutions [14], [27], which occur only in duplicated copies, and *unrestricted* substitutions [14], [31]–[33], which may occur anywhere, have been studied. The closest work to the current paper, [32], constructed error-correcting codes for short duplications and at most one (unrestricted) edit. However, compared to the codes in [10] for only duplications, the codes in [32] incur an asymptotic rate loss when $q = 4$ in order to correct the additional edit. The current paper provides codes for correcting any number of short duplications and at most p (unrestricted) edits with no asymptotic rate penalty compared to correcting short duplications only, where p and the alphabet size q are constants.

One of the challenging aspects of correcting multiple types of errors, even when optimal codes for individual error types exist, is that codes for each type may utilize incompatible strategies. In particular, correcting duplications relies on constrained codes (local constraints) while edits are corrected using error-correcting codes with codewords that satisfy certain global constraints. Combining these strategies is not straightforward as encoding one set of constraints may violate the other, or alter how errors affect the data. Our strategy, which can be viewed as modified concatenation described in [34], is to first encode user data as a constrained sequence $\boldsymbol{x}$, which does not contain any repeats of length ≤ 6 (such sequences are called irreducible). Then using *syndrome compression*, we compute and append to $\boldsymbol{x}$ a "parity" sequence $\boldsymbol{r}$ to help correct errors that occur in $\boldsymbol{x}$. Syndrome compression has recently

This work was supported in part by NSF grants under grant CIF 1816409 and CIF 1755773.

This paper was presented in part at the IEEE ISIT2021 [1] and IEEE ISIT2022 [2].

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

Shuche Wang is with the Institute of Operations Research and Analytics, National University of Singapore, Singapore, (email: shuche.wang@u.nus.edu).

Hao Lou is with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA, 22903, USA, (email: hl2nu@virginia.edu).

Ryan Gabrys is with the Calit2, University of California-San Diego, U.S.A., (email: rgabrys@ucsd.edu).

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

been used to provide explicit constructions for correcting a wide variety of errors with redundancy as low as roughly twice the Gilbert-Varshamov bound [35]–[38].

Another challenge arises from the interaction between the errors. When both short duplications and edits are present, a single edited symbol may be duplicated many times and affect an unbounded segment. However, when the input is an irreducible sequence, after removing all tandem copies with length ≤ 3 from the output, the effects of short duplications and at most p edits can be localized in at most p substrings, each with length ≤ 17 [32]. Using the structure of these localized alterations, we describe the set of strings that can be confused with $\boldsymbol{x}$ and bound its size, allowing us to leverage syndrome compression to reduce redundancy.

A third challenge is ensuring that the appended vector $\boldsymbol{r}$ is itself protected against errors and can be decoded correctly. We do this by introducing a higher-redundancy MDS-based code over irreducible sequences. After decoding the appended vector, we use it to recover the data by eliminating incorrect confusable inputs.

Compared to the explicit code for short duplications only [10], the proposed code corrects $\leq p$ edits in addition to the duplications at the extra cost of roughly $8p(\log_q n)(1+o(1))$ symbols of redundancy for $q \geq 4$, and achieves the same asymptotic code rate. We note that the state-of-the-art redundancy for correcting p edits is no less than $4p\log_q n(1+o(1))$ [37]. Time complexities of both the encoding and decoding processes are polynomial when p is a constant.

For simplicity of exposition, we first consider the channel with short duplications and *substitutions* only and construct codes for it. Then, in Subsection IV-B, we show that the same codes can correct short duplications and *edit* errors. We note that short duplications and edits may occur in any order. Henceforth, the term duplication refers to short duplications only.

The paper is organized as follows. Section II presents the notation and preliminaries. In Section III, we derive an upper bound on the size of the confusable set for an irreducible string, which is a key step of the syndrome compression technique used to construct our error-correcting codes. Then, Section IV presents the code construction as well as a discussion of the redundancy and the encoding/decoding complexities, under the assumption that the syndrome information can be recovered correctly by an auxiliary error-correcting code, which is described in Section V. Finally, Section VI concludes the main results.

II. Notation and Preliminaries

Let $\Sigma_q = \{0, 1, 2, \ldots, q-1\}$ represent a finite alphabet of size q and Σ_q^n the set of all strings of length n over Σ_q. Furthermore, let Σ_q^* be the set of all finite strings over Σ_q, including the empty string Λ. Given two integers a, b with $a \leq b$, the set $\{a, a+1, \ldots, b\}$ is shown as $[a, b]$. We simplify $[1, b]$ as $[b]$. For an integer $a \geq 1$, we define $b \bmod^+ a$ as the integer in $[a]$ whose remainder when divided by a is the same as that of b. Unless otherwise stated, logarithms are to the base 2.

We use bold symbols to denote strings over Σ_q, i.e., $\boldsymbol{x}, \boldsymbol{y}_j \in \Sigma_q^*$. The entries of a string are represented by plain typeface, e.g., the ith elements of $\boldsymbol{x}, \boldsymbol{y}_j \in \Sigma_q^*$ are $x_i, y_{ji} \in \Sigma_q$, respectively. For two strings $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, let $\boldsymbol{xy}$ denote their concatenation. Given four strings $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \Sigma_q^*$, if $\boldsymbol{x} = \boldsymbol{uvw}$, then $\boldsymbol{v}$ is called a substring of $\boldsymbol{x}$. Furthermore, we let $|\boldsymbol{x}|$ represent the length of a string $\boldsymbol{x} \in \Sigma_q^n$, and let $\|S\|$ denote the size (the number of elements) of a set S.

A *(tandem) duplication* of length k is the operation of generating a copy of a substring and inserting it directly following the substring, where k is the length of the copy. For example, for $\boldsymbol{x} = \boldsymbol{uvw}$ with $|\boldsymbol{v}| = k$, a (tandem) duplication may generate $\boldsymbol{uvvw}$, where $\boldsymbol{vv}$ is called a *(tandem) repeat* with length $2k$. A duplication of length at most 3 is called *a short duplication*. Unless otherwise stated, the term duplication is used to refer to short duplications in the rest of the paper. For example, given $\boldsymbol{x} = 213012 \in \Sigma_4^*$, a sequence of duplications may produce

$$\begin{aligned}\boldsymbol{x} = 213012 &\to 213\underline{213}012 \to 21321303\underline{0}12 \\ &\to 213\underline{2}21303012 = \boldsymbol{x}',\end{aligned} \tag{1}$$

where the duplicated copies are marked with underlines. We call $\boldsymbol{x}'$ a *descendant* of $\boldsymbol{x}$, i.e., a string generated from $\boldsymbol{x}$ by a sequence of duplications. Furthermore, for a string $\boldsymbol{x} \in \Sigma_q^*$, let $\mathcal{D}(\boldsymbol{x}) \subseteq \Sigma_q^*$ be the set of all descendants generated from $\boldsymbol{x}$ by an arbitrary number of duplications. Note that, unless $\boldsymbol{x} = \Lambda$, $\mathcal{D}(\boldsymbol{x})$ is an infinite set.

A *deduplication* of length k replaces a repeat $\boldsymbol{vv}$ by $\boldsymbol{v}$ with $|\boldsymbol{v}| = k$. In the rest of the paper, unless otherwise stated, dedulications are assumed to be of length at most 3. For example, the string $\boldsymbol{x}$ in (1) can be recovered from $\boldsymbol{x}'$ by three deduplications.

The set of *irreducible strings* of length n over Σ_q, denoted $\mathrm{Irr}_q(n)$, consists of strings without repeats $\boldsymbol{vv}$, where $|\boldsymbol{v}| \leq 3$. Furthermore, $\mathrm{Irr}_q(*)$ represents all irreducible strings of finite length over Σ_q. The *duplication root* of $\boldsymbol{x}'$ is an irreducible string $\boldsymbol{x}$ such that $\boldsymbol{x}'$ is a descendant of $\boldsymbol{x}$. Equivalently, $\boldsymbol{x}$ can be obtained from $\boldsymbol{x}'$ by performing all possible deduplications. Any string $\boldsymbol{x}'$ has one and only one duplication root [10][1], denoted $R(\boldsymbol{x}')$. The uniqueness of the root implies that if $\boldsymbol{x}''$ is a descendant of $\boldsymbol{x}'$, we have $R(\boldsymbol{x}') = R(\boldsymbol{x}'')$. For a set S of strings, we define $R(S) = \{R(\boldsymbol{s}) : \boldsymbol{s} \in S\}$ as the set of the duplication roots of the elements of S.

Besides duplications, we also consider edit errors. An edit may be a substitution, which replaces a symbol by another one from the same alphabet; a deletion, which removes a symbol; or an insertion, which inserts a symbol from the same alphabet. Continuing the example in (1), two substitutions and two duplications applied to $\boldsymbol{x}'$ may produce

$$\begin{aligned}\boldsymbol{x}' = 213221303012 &\to 213211303012 \\ &\to 21321\underline{321}1303012 \to 213213211\underline{32}3012 \\ &\to 213213211323\underline{32}3012 = \boldsymbol{x}'',\end{aligned}$$

where the substituted symbols are marked in red. Let $\mathcal{D}^{\leq p}(\boldsymbol{x}) \subseteq \Sigma_q^*$ represent the set of strings derived from $\boldsymbol{x}$ by an

[1]Note that this statement only applies to duplications of length at most 3. For duplications of length at most 4, the root is not necessarily unique.

arbitrary number of duplications and at most p substitutions. In the example above, we have $\boldsymbol{x}'' \in \mathcal{D}^{\leq 2}(\boldsymbol{x})$. Note that the alphabet over which $\mathcal{D}^{\leq p}(\boldsymbol{x})$ is defined affects its contents. For example, for $\boldsymbol{x} = 012$, $\mathcal{D}^{\leq 1}(\boldsymbol{x})$ contains 013 if the alphabet is Σ_4 but not if the alphabet is Σ_3. Unless $\boldsymbol{x} = \Lambda$, $\mathcal{D}^{\leq p}(\boldsymbol{x})$ is infinite.

We define a *substring edit* in a string $\boldsymbol{x} \in \Sigma_q^*$ as the operation of replacing a substring $\boldsymbol{u}$ with a string $\boldsymbol{v} \in \Sigma_q^*$, where at least one of $\boldsymbol{u}, \boldsymbol{v}$ is nonempty. The length of the substring edit is $\max\{|\boldsymbol{u}|, |\boldsymbol{v}|\}$. An *$L$-substring edit* is one whose length is at most L. For example, given $\boldsymbol{x} = 0123456$, a 4-substring edit can generate the sequence $\boldsymbol{y} = 0\underline{789}56$ or the sequence $\boldsymbol{z} = 01\underline{89}23456$, where the inserted strings are underlined. Furthermore, a *burst deletion* in $\boldsymbol{x} \in \Sigma_q^*$ is defined as removing a substring $\boldsymbol{v}$ of $\boldsymbol{x}$, where $|\boldsymbol{v}|$ is the length of the burst deletion.

Given a sequence $\boldsymbol{x} \in \Sigma_q^n$, we define the binary matrix $\mathcal{U}(\boldsymbol{x})$ of $\boldsymbol{x}$ with dimensions $\lceil \log q \rceil \times n$ as

$$\begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{\lceil \log q \rceil,1} & u_{\lceil \log q \rceil,2} & \cdots & u_{\lceil \log q \rceil,n} \end{bmatrix}, \tag{2}$$

where the jth column of $\mathcal{U}(\boldsymbol{x})$ is the binary representation of the jth symbol of $\boldsymbol{x}$ for $j \in [n]$. The ith row of $\mathcal{U}(\boldsymbol{x})$ is denoted as $\mathcal{U}_i(\boldsymbol{x})$ for $i \in \lceil \log q \rceil$.

The redundancy of a code $\mathcal{C} \subseteq \Sigma_q^n$ of length n is defined as $n - \log_q \|\mathcal{C}\|$ symbols, and its rate as $\frac{1}{n} \log \|\mathcal{C}\|$ bits per symbol. Asymptotic rate is the limit superior of the rate as the length n grows.

In order to construct error-correcting codes by applying the syndrome compression technique [35], we first introduce some auxiliary definitions and a theorem.

Suppose $q \geq 3$ is a constant. We start with the definition of confusable sets for a given channel and a given set of strings $S \subseteq \Sigma_q^n$. In our application, S is the set of irreducible strings, upon which the proposed codes will be constructed.

Definition 1. *A* confusable set $B(\boldsymbol{x}) \subseteq S$ *of* $\boldsymbol{x} \in S$ *consists of all* $\boldsymbol{y} \in S$*, excluding* $\boldsymbol{x}$*, such that* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *can produce the same output when passed through the channel.*

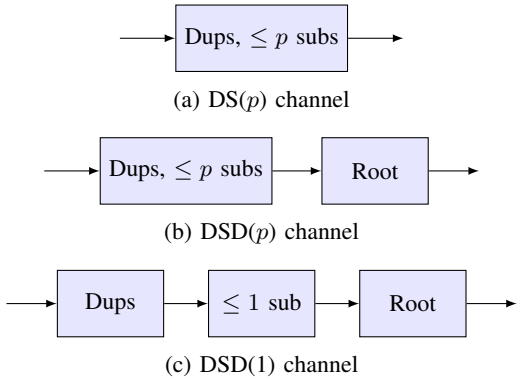
Definition 2. *Let* $\mathcal{R}(n)$ *be an integer function of* n*. A* labeling function for the confusable sets $B(\boldsymbol{x}), \boldsymbol{x} \in S$*, is a function*

$$f : \Sigma_q^n \to \Sigma_{2^{\mathcal{R}(n)}}$$

such that, for any $\boldsymbol{x} \in S$ *and* $\boldsymbol{y} \in B(\boldsymbol{x})$*,* $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$*.*

Theorem 3. *(c.f. [35, Theorem 5]) Let* $f : \Sigma_q^n \to \Sigma_{2^{\mathcal{R}(n)}}$*, where* $\mathcal{R}(n) = o(\log \log n \cdot \log n)$*, be a labeling function for the confusable sets* $B(\boldsymbol{x}), \boldsymbol{x} \in S$*. Then there exists an integer* $a \leq 2^{\log \|B(\boldsymbol{x})\| + o(\log n)}$ *such that for all* $\boldsymbol{y} \in B(\boldsymbol{x})$*, we have* $f(\boldsymbol{x}) \not\equiv f(\boldsymbol{y}) \bmod a$*.*

The above definitions and theorem are used in our code construction based on syndrome compression, presented in Section IV. The construction and analysis rely on the confusable sets for the channel, discussed in the next section.



(a) DS(p) channel

(b) DSD(p) channel

(c) DSD(1) channel

Figure 1: Any error-correcting code for channel (b) is also an error-correcting code for channel (a). The confusable set for a channel obtained by concatenating p copies of channel (c) contains the confusable set for channel (b).

III. Confusable Sets for Channels with Short Duplication and Substitution Errors

In this section, we study the size of confusable sets of input strings passing through channels with an arbitrary number of duplications and at most p substitutions. This quantity will be used to derive a Gilbert-Varshamov bound and, in the next section, to construct our error-correcting codes.

Since the duplication root is unique, and duplications and deduplications do not alter the duplication root of the input, $\mathrm{Irr}_q(n)$ is a code capable of correcting duplications. The decoding process simply removes all tandem repeats. In other words, if we append a root block, which deduplicates all repeats and produces the root of its input, to the channel with duplication errors, any irreducible sequence passes through this concatenated channel with no errors. This approach produces codes with the same asymptotic rate as that of [10], achieving the highest possible asymptotic rate.

Similar to [32], we extend this strategy to design codes for the channel with duplication and at most p substitution errors, denoted the DS(p) channel and shown in Figure 1a. Note that the duplications and substitutions can occur in any order. We take the code to be a subset of irreducible strings and find the code for a new channel obtained by concatenating a root block to the channel with duplication and substitution errors, denoted as the DSD(p) channel and shown in Figure 1b. Clearly, any error-correcting code for DSD(p) is also an error-correcting code for the DS(p) channel.

We now define the confusable sets over $\mathrm{Irr}_q(n)$ for the DSD(p) channel and bound its size, which is needed to construct the code and determine its rate.

Definition 4. *For* $\boldsymbol{x} \in \mathrm{Irr}_q(n)$*, let*

$$B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathrm{Irr}_q(n) : \boldsymbol{y} \neq \boldsymbol{x}, \\ R(\mathcal{D}^{\leq p}(\boldsymbol{x})) \cap R(\mathcal{D}^{\leq p}(\boldsymbol{y})) \neq \varnothing\} \tag{3}$$

denote the irreducible-confusable set *of* $\boldsymbol{x}$*.*

Note that the DSD(1) channel can be represented as shown in Figure 1c. This is because the sequence of errors consists of duplications, substitutions, more duplications, and finally

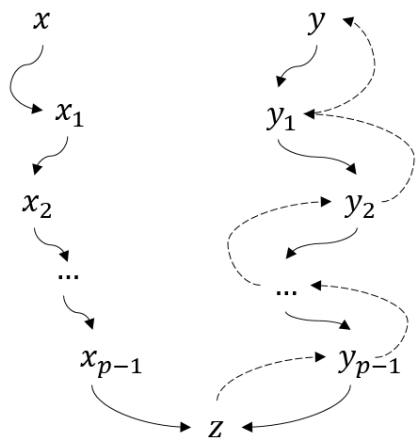


Figure 2: A sequence $\boldsymbol{z} = \boldsymbol{x}_p = \boldsymbol{y}_p$ that can be obtained from both $\boldsymbol{x}$ and $\boldsymbol{y}$ through channels resulting from the concatenation of p DSD(1) channels, each shown by a solid arrow. The dashed arrows represent the reverse relationships and each $\boldsymbol{y}_{i-1}$ can be obtained by passing $\boldsymbol{y}_i$ through a DSD(1) channel.

all deduplications. Hence, duplications that occur after the substitutions are all deduplicated and we may equivalently assume they have not occurred. Next, observe that the confusable set for the concatenation of p DSD(1) channels contains the confusable set for a DSD(p) channel. In other words, if the same string is input to the concatenation of p DSD(1) channels and to a DSD(p) channel, the set of possible outputs of the former is a superset of the set of possible outputs of the latter. Hence, we have the following lemma.

Lemma 5. *An error-correcting code for the concatenation of p DSD(1) channels is also an error-correcting code for the DSD(p) channel.*

We can thus focus on this concatenated channel. The advantage of considering DSD(1) is that it is reversible in the sense that if $\boldsymbol{v}$ can be obtained from an input $\boldsymbol{u}$, then $\boldsymbol{u}$ can be obtained from the input $\boldsymbol{v}$, and this simplifies our analysis. In particular, we have $\boldsymbol{u} \in R(\mathcal{D}^{\leq 1}(\boldsymbol{v}))$ and $\boldsymbol{v} \in R(\mathcal{D}^{\leq 1}(\boldsymbol{u}))$.

Figure 2 shows a confusable string $\boldsymbol{z}$ obtainable from irreducible sequences $\boldsymbol{x} \in \mathrm{Irr}_q(n)$ and $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$, after passing through p DSD(1) channels, each represented by a solid arrow. More precisely, $\boldsymbol{x}_i \in R(\mathcal{D}^{\leq 1}(\boldsymbol{x}_{i-1}))$ and $\boldsymbol{y}_i \in R(\mathcal{D}^{\leq 1}(\boldsymbol{y}_{i-1}))$, where $\boldsymbol{x} = \boldsymbol{x}_0, \boldsymbol{y} = \boldsymbol{y}_0, \boldsymbol{z} = \boldsymbol{x}_p = \boldsymbol{y}_p$. Furthermore, $\boldsymbol{y}_{i-1} \in R(\mathcal{D}^{\leq 1}(\boldsymbol{y}_i))$. Hence, $\boldsymbol{y}$ can be generated from $\boldsymbol{x}$ by concatenating the solid-line path from $\boldsymbol{x}$ to $\boldsymbol{z}$ and the dashed-line path from $\boldsymbol{z}$ to $\boldsymbol{y}$, i.e., $\boldsymbol{x} \to \boldsymbol{x}_1 \to \cdots \to \boldsymbol{z} \to \boldsymbol{y}_{p-1} \to \cdots \to \boldsymbol{y}$, where each $\to$ represents a DSD(1) channel. Considering the number of possibilities in each step gives the following lemma.

Lemma 6. *For $\boldsymbol{x} \in \mathrm{Irr}_q(n)$,*

$$\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\| \leq \max_{\boldsymbol{x}_i, \boldsymbol{y}_i} \prod_{i=0}^{p-1} \|R(\mathcal{D}^{\leq 1}(\boldsymbol{x}_i))\| \prod_{i=1}^{p} \|R(\mathcal{D}^{\leq 1}(\boldsymbol{y}_i))\|$$

where the maximum for $\boldsymbol{x}_i$ (resp. $\boldsymbol{y}_i$) is over sequences that can result from $\boldsymbol{x}$ (resp. $\boldsymbol{y}$) passing through the concatenation of i DSD(1) channels.

It thus suffices to find $\|R(\mathcal{D}^{\leq 1}(\boldsymbol{x}))\|$ for $\boldsymbol{x} \in \mathrm{Irr}_q(*)$. As

$$\begin{aligned}\|R(\mathcal{D}^{\leq 1}(\boldsymbol{x}))\| &\leq \|R(\mathcal{D}^{1}(\boldsymbol{x}))\| + \|R(\mathcal{D}(\boldsymbol{x}))\| \\ &= \|R(\mathcal{D}^{1}(\boldsymbol{x}))\| + 1,\end{aligned}$$

we find an upper bound on $\|R(\mathcal{D}^1(\boldsymbol{x}))\|$, in Lemma 8, using the following lemma from [32].

Lemma 7. *[32, Lemma 3] Let $\boldsymbol{x}$ be any string of length at least 5 and $\boldsymbol{x}' \in \mathcal{D}(\boldsymbol{x})$. For any decomposition of $\boldsymbol{x}$ as*

$$\boldsymbol{x} = \ \boldsymbol{r}\ ab\ c\ de\ \boldsymbol{s},$$

for $a, b, c, d, e \in \Sigma_q$ and $\boldsymbol{r}, \boldsymbol{s} \in \Sigma_q^*$, there is a decomposition of $\boldsymbol{x}'$ as

$$\boldsymbol{x}' = \boldsymbol{u}\ ab\ \boldsymbol{w}\ de\ \boldsymbol{v}$$

such that $\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{v} \in \Sigma_q^*$, $\boldsymbol{u}ab \in \mathcal{D}(\boldsymbol{r}ab)$, $ab\boldsymbol{w}de \in \mathcal{D}(abcde)$, and $de\boldsymbol{v} \in \mathcal{D}(de\boldsymbol{s})$.

Lemma 8. *For an irreducible string $\boldsymbol{x} \in \Sigma_q^n$,*

$$\|R(\mathcal{D}^1(\boldsymbol{x}))\| \leq n \max_{\boldsymbol{t} \in \Sigma_q^5} \|R(\mathcal{D}^1(\boldsymbol{t}))\|.$$

Proof: Given an irreducible string $\boldsymbol{x} \in \mathrm{Irr}_q(n)$, let $\boldsymbol{x}' \in \mathcal{D}(\boldsymbol{x})$ be obtained from $\boldsymbol{x}$ through duplications and $\boldsymbol{x}''$ obtained from $\boldsymbol{x}'$ by a substitution. For a given $\boldsymbol{x}$, $\|R(\mathcal{D}^1(\boldsymbol{x}))\|$ equals the number of possibilities for $R(\boldsymbol{x}'')$ as $\boldsymbol{x}''$ varies. Note that duplications that occur after the substitution do not affect the root. So we have assumed that the substitution is the last error before the root is found.

Decompose $\boldsymbol{x}$ as $\boldsymbol{x} = \boldsymbol{r}abcde\boldsymbol{s}$ with $\boldsymbol{r}, \boldsymbol{s} \in \mathrm{Irr}_q(*)$ and $a, b, c, d, e \in \Sigma_q$, so that the substituted symbol in $\boldsymbol{x}'$ is a copy of c. Note that if $|\boldsymbol{x}| < 5$ or if a copy of one of its first two symbols or its last two symbols is substituted, then we can no longer write $\boldsymbol{x}$ as described. To avoid considering these cases separately, we may append two dummy symbols to the beginning of $\boldsymbol{x}$ and two dummy symbols to the end of $\boldsymbol{x}$, where the four dummy symbols are distinct and do not belong to Σ_q, and prove the result for this new string. Since these dummy symbols do not participate in any duplication, substitution, or deduplication events, the proof is also valid for the original $\boldsymbol{x}$.

By Lemma 7, we can write

$$\begin{aligned}\boldsymbol{x} &= \ \boldsymbol{r}\ ab\ c\ de\ \boldsymbol{s} \\ \boldsymbol{x}' &= \boldsymbol{u}\ ab\ \boldsymbol{w}\ de\ \boldsymbol{v}, \\ \boldsymbol{x}'' &= \boldsymbol{u}\ ab\ \boldsymbol{z}\ de\ \boldsymbol{v},\end{aligned} \tag{4}$$

where $\boldsymbol{u}ab \in \mathcal{D}(\boldsymbol{r}ab)$, $ab\boldsymbol{w}de \in \mathcal{D}(abcde)$, $de\boldsymbol{v} \in \mathcal{D}(de\boldsymbol{s})$, and $\boldsymbol{z}$ is obtained from $\boldsymbol{w}$ by substituting a copy of c. From (4), $R(\boldsymbol{x}'') = R(\boldsymbol{r}R(ab\boldsymbol{z}de)\boldsymbol{s})$, where $R(ab\boldsymbol{z}de)$ starts with ab and ends with de (which may fully or partially overlap).

To determine $\|R(\mathcal{D}^1(\boldsymbol{x}))\|$, we count the number of possi-

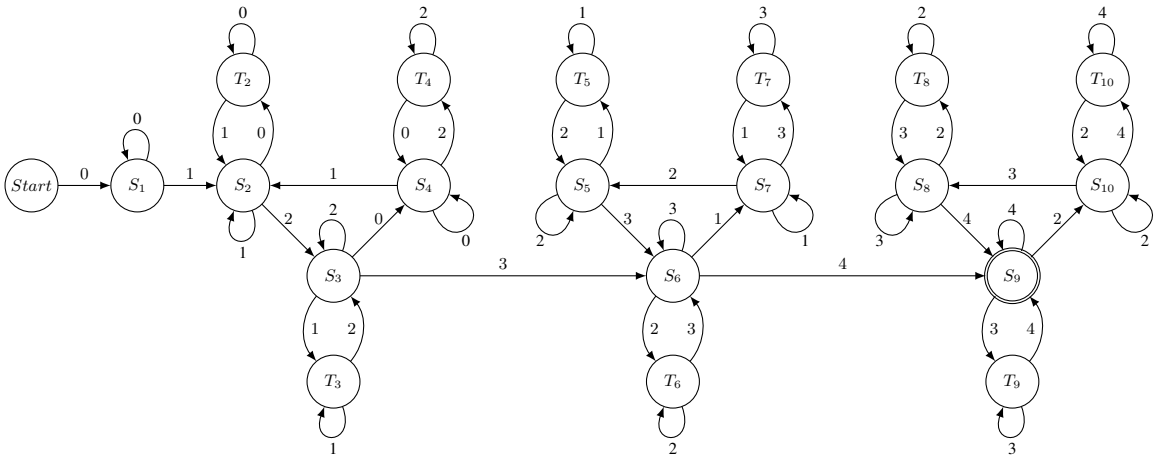


Figure 3: Finite automaton for the regular language $D^*(01234)$ based on [28].

bilities for $R(\boldsymbol{x}'')$ as $\boldsymbol{x}''$ varies. Considering the decomposition of $\boldsymbol{x}''$ into $\boldsymbol{u}ab\boldsymbol{z}de\boldsymbol{v}$ given in (4), we note that if $R(ab\boldsymbol{z}de)$ is given, then $R(\boldsymbol{x}'') = R(\boldsymbol{r}R(ab\boldsymbol{z}de)\boldsymbol{s})$ is uniquely determined. So to find an upper bound, it suffices to count the number of possibilities for $R(ab\boldsymbol{z}de)$. We thus have

$$\|R(\mathcal{D}^1(\boldsymbol{x}))\| \leq \sum \|\{R(ab\boldsymbol{z}de) : ab\boldsymbol{z}de \in \mathcal{D}^1(abcde)\}\|,$$

where the sum is over the choices of c in $\boldsymbol{x}$, or equivalently the decompositions of $\boldsymbol{x}$ into $\boldsymbol{r}abcde\boldsymbol{s}$, in (4). As there are n choices for c, we have

$$\|R(\mathcal{D}^1(\boldsymbol{x}))\| \leq n \max_{\boldsymbol{t}\in\Sigma_q^5} \|R(\mathcal{D}^1(\boldsymbol{t}))\|.$$

■

The next lemma provides a bound on $\|R(\mathcal{D}^1(\boldsymbol{t}))\|$ for $\boldsymbol{t} \in \Sigma_q^5$ by identifying the "worst case". The proof is given in Appendix A.

Lemma 9. *Given $q \geq 3$, we have*

$$\max_{\boldsymbol{t}\in\Sigma_q^5} \|R(\mathcal{D}^1(\boldsymbol{t}))\| \leq \|R(\mathcal{D}^1(01234))\|,$$

where $\mathcal{D}^1(01234) \subseteq \Sigma_{q+4}^*$ (the substituted symbol can be replaced with another symbol from Σ_{q+4}).

As shown in [28], $\mathcal{D}(01234)$ is a regular language whose words can be described as paths from 'Start' to S_9 in the finite automaton given in Figure 3. Then $\mathcal{D}^1(01234)$ is equivalent to the set of paths from 'Start' to S_9 but with the label on one edge substituted. We will use this observation to bound $\|R(\mathcal{D}^1(01234))\|$ in Lemma 11. The next lemma establishes a symmetric property of the automaton that will be useful in Lemma 11. Lemma 10 is proved by showing that there is a bijective function $h : U \to V$ between U and V and between $R(U)$ and $R(V)$. Specifically, for $\boldsymbol{u} = u_1 \cdots u_n$, we let $\boldsymbol{v} = h(\boldsymbol{u}) = \bar{u}_n \bar{u}_{n-1} \ldots \bar{u}_1$, where for $a \in \{0, 1, 2, 3, 4\}$, $\bar{a} = 4 - a$. A detailed proof is given in Appendix B.

Lemma 10. *Let U and V be the sets of labels of all paths from Start to any state and from any state to S_9, respectively, in the finite automaton of Figure 3. Then $\|U\| = \|V\|$ and $\|R(U)\| = \|R(V)\|$.*

Lemma 11. *For $\hat{q} \geq 5$ and $\mathcal{D}^1(01234) \subseteq \Sigma_{\hat{q}}^*$, where the substitution replaces a symbol with any symbol from $\Sigma_{\hat{q}}$, we have*

$$\|R(\mathcal{D}^1(01234))\| \leq 22^2(\hat{q} - 1).$$

Proof: Based on [28], recall that $\mathcal{D}(01234)$ is a regular language whose words can be described as paths from 'Start' to S_9 in the finite automaton given in Figure 3, where the word associated with each path is the sequence of the edge labels. Let $\boldsymbol{x}' \in \mathcal{D}(01234)$ and let $\boldsymbol{x}''$ be generated from $\boldsymbol{x}'$ by a substitution. Assume $\boldsymbol{x}' = \boldsymbol{u}w\boldsymbol{v}$ and $\boldsymbol{x}'' = \boldsymbol{u}\hat{w}\boldsymbol{v}$, where $\boldsymbol{u}, \boldsymbol{v} \in \Sigma_5^*, w \in \Sigma_5$ and $\hat{w} \in \Sigma_{\hat{q}} \setminus \{w\}$. So there are $\hat{q} - 1$ choices for $\hat{w}$. The string $\boldsymbol{u}$ represents a path from 'Start' to some state s_u and the string $\boldsymbol{v}$ represents a path from some state s_v to S_9 in the automaton, where there is an edge with label w from s_u to s_v.

As $\boldsymbol{x}'' = \boldsymbol{u}\hat{w}\boldsymbol{v}$, we have $R(\boldsymbol{x}'') = R(R(\boldsymbol{u})\hat{w}R(\boldsymbol{v}))$, where $R(\boldsymbol{u})$ is an irreducible string represented by a path from "Start" to state s_u, and $R(\boldsymbol{v})$ is an irreducible string represented by a path from s_v to S_9. Define U and V as in Lemma 10. We thus have $\|R(\mathcal{D}^1(\boldsymbol{x}))\| \leq ||R(U)|| \times (\hat{q}-1) \times ||R(V)|| = ||R(U)||^2 \times (\hat{q} - 1)$. By inspection, we can show that

$$\begin{aligned} R(U) = \{&\Lambda, 0, 01, 01201, 012, 0120, 010, 012010, \\ &0121, 01202, 0123, 01232, 01231, 012313, 012312, \\ &0123121, 01234, 012343, 012342, 0123424, \\ &0123423, 01234232\}, \end{aligned}$$

and hence $||R(U)|| = 22$, completing the proof. ■

Theorem 12. *For an irreducible string $\boldsymbol{x} \in \Sigma_q^n$, with $q \geq 3$,*

$$\|R(\mathcal{D}^{\leq 1}(\boldsymbol{x}))\| \leq 968nq + 1.$$

Proof: From Lemmas 8, 9, and 11, it follows that $\|R(\mathcal{D}^1(\boldsymbol{x}))\| \leq 22^2 n(\hat{q} - 1) \leq 2q \cdot 22^2 n = 968nq$ with $\hat{q} = q + 4$. Furthermore, $\|R(\mathcal{D}^{\leq 1}(\boldsymbol{x}))\| \leq \|R(\mathcal{D}^1(\boldsymbol{x}))\| + 1$. ■

We can now use Theorem 12 along with Lemma 6, to find a

bound on $\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\|$. To do so, we need to bound the size of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ shown in Figure 2, for which the following theorem is of use. The theorem is a direct extension of [32, Theorem 5] and thus requires no proof. An example demonstrating the theorem is given in Appendix E.

Theorem 13 (c.f. [32, Theorem 5])**.** *Given strings $\boldsymbol{x} \in \Sigma_q^n$ and $\boldsymbol{v} \in \mathcal{D}^{\leq p}(\boldsymbol{x})$, $R(\boldsymbol{v})$ can be obtained from $R(\boldsymbol{x})$ by at most p $\mathcal{L}$-substring edits, where $\mathcal{L} = 17$.*

It follows from the theorem that for $1 \leq i \leq p$,

$$|\boldsymbol{x}_i| \leq n + p\mathcal{L}, \ \ |\boldsymbol{y}_i| \leq n + p\mathcal{L}. \tag{5}$$

The next theorem then follows from Lemmas 6 and 12.

Theorem 14. *Let $\boldsymbol{x} \in \mathrm{Irr}_q(n) \subseteq \Sigma_q^n$ be an irreducible string of length n with $q \geq 3$. The irreducible-confusable set $B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$ of $\boldsymbol{x}$ satisfies*

$$\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\| \leq (968q(n + p\mathcal{L}) + 1)^{2p}.$$

The size of the confusable sets will be used for our code construction. It also allows us to derive a Gilbert-Varshamov (GV) bound, as follows.

Theorem 15. *There exists a code of length n capable of correcting any number of duplications and at most p substitutions with size at least*

$$\frac{\|\mathrm{Irr}_q(n)\|}{(968q(n + p\mathcal{L}) + 1)^{2p}}.$$

We will show in Lemma 23 that the size of the code with the highest asymptotic rate for correcting duplications only is essentially $\|\mathrm{Irr}_q(n)\|$. Assuming that p and q are constants, this GV bound shows that a code exists for additionally correcting up to p substitution errors with extra redundancy of approximately $2p \log_q n$ symbols. The two constructions presented in the next section have extra redundancies of $4p \log_q n$ and $8p \log_q n$, which are only small constant factors away from this existential bound.

IV. Low-redundancy error-correcting codes

As stated in Section III, our code for correcting duplications and substitutions is a subset of irreducible strings of a given length. In this section, we construct this subset by applying the syndrome compression technique [35], where we will make use of the size of the irreducible-confusable set $\|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\|$ derived in Section III. In this section, unless otherwise stated, we assume that both $q \geq 4$ and p are constant.

We begin by presenting the code constructions for correcting duplications and *substitutions* in Subsection IV-A, assuming the existence of appropriate labeling functions used to produce the syndrome information and an auxiliary error-correcting code used to protect it. The labeling functions will be discussed in Subsection IV-C, while the auxiliary ECC is presented in Section V. In Subsection IV-B, we show that the proposed codes can in fact correct duplications and *edits*. The redundancy of the codes and the computational complexities of their encoding and decoding are discussed in Subsections IV-D and IV-E, respectively.

A. Code constructions

We first present a code in Construction A that assumes an error-free side channel is available, where the length of the sequence passing through the side channel is logarithmic in the length of the sequence passing through the main channel. We then present the main result of this section, Construction B, which does not make such an assumption and is intended for a single noisy channel. Construction A helps motivate certain components of Construction B and make its proof of correctness more clear. In addition, it may have potential practical applications. For example, in a DNA storage system, metadata of the data stored on DNA may be stored on silicon-based devices such as disk or flash. Due to the maturity of these technologies, they may provide a nearly error-free channel, suitable for storing a small amount of side information.

1) Channels with error-free side channels: In the construction below, $\boldsymbol{x}$ is transmitted through the noisy channel, while $\boldsymbol{r}$, which encodes the information $(a, f(\boldsymbol{x}) \bmod a)$ is transmitted through an error-free channel.

Construction A. *Let n, p, q be positive integers. Furthermore, let f be a (labeling) function and, for each $\boldsymbol{x} \in \mathrm{Irr}_q(n)$, $a_{\boldsymbol{x}}$ be a positive integer, such that for any $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$, $f(\boldsymbol{x}) \not\equiv f(\boldsymbol{y}) \bmod a_{\boldsymbol{x}}$. Define*

$$\mathcal{C}_n^A = \{(\boldsymbol{x}, \boldsymbol{r}) : \boldsymbol{x} \in \mathrm{Irr}_q(n), \boldsymbol{r} = (a_{\boldsymbol{x}}, f(\boldsymbol{x}) \bmod a_{\boldsymbol{x}})\},$$

where $\boldsymbol{r}$ is assumed to be the q-ary representation of $(a_{\boldsymbol{x}}, f(\boldsymbol{x}) \bmod a_{\boldsymbol{x}})$.

We consider the length of this code to be $N = n + |\boldsymbol{r}|$. As will be observed in (8), $|\boldsymbol{r}| = O(\log_q n)$ and so the sequence carried by the side channel is logarithmic in length. Recall that the existence of the labeling functions is discussed in Subsection IV-C.

Theorem 16. *The code in Construction A, assuming the labeling function f and $a_{\boldsymbol{x}}$ (for each $\boldsymbol{x} \in \mathrm{Irr}_q(n)$) exist, can correct any number of duplications and at most p substitutions applied to $\boldsymbol{x}$, provided that $\boldsymbol{r}$ is transmitted through an error-free channel.*

Proof: Let the retrieved word from storing $\boldsymbol{x}$ be $\boldsymbol{v} \in R(\mathcal{D}^{\leq p}(\boldsymbol{x}))$. Note that $a_{\boldsymbol{x}}$ and $f(\boldsymbol{x}) \bmod a_{\boldsymbol{x}}$ can be recovered error-free from $\boldsymbol{r}$. By definition, for all $\boldsymbol{y} \neq \boldsymbol{x}$ that could produce the same $\boldsymbol{v}$, we have $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$. But then, $f(\boldsymbol{y}) \not\equiv f(\boldsymbol{x}) \bmod a_{\boldsymbol{x}}$, and so we can determine $\boldsymbol{x}$ by exhaustive search. ∎

2) Channels with no side channels: To better illustrate the construction with no side channels, let us first observe what the issues are with simply concatenating $\boldsymbol{x}$ and $\boldsymbol{r}$ and forming codewords of the form $\boldsymbol{xr}$.

- The code in Construction A relies on a sequence $\boldsymbol{v} \in R(\mathcal{D}^{\leq p}(\boldsymbol{x}))$ but if $\boldsymbol{xr}$ is stored, the output of the channel is a sequence $\boldsymbol{w} \in R(\mathcal{D}^{\leq p}(\boldsymbol{xr}))$. As the boundary between $\boldsymbol{x}$ and $\boldsymbol{r}$ becomes unclear after duplication and substitution errors, it is difficult to find $\boldsymbol{v} \in R(\mathcal{D}^{\leq p}(\boldsymbol{x}))$ from $\boldsymbol{w} \in R(\mathcal{D}^{\leq p}(\boldsymbol{xr}))$. To address this, instead of finding $\boldsymbol{v}$, we find a sufficiently long prefix, as discussed

in Lemma 17. This will also require us to modify the labeling function.
- The decoding process requires the information encoded in $\boldsymbol{r}$, which is now subject to errors. We will address this by using a high-redundancy code that can protect this information, introduced in Lemma 18 and discussed in detail in Subsection V-C.
- The codewords need to be irreducible. This is discussed in Lemma 19.

For integers p, j, denote by $\mathcal{D}^{\leq p}_{\leq j}(\boldsymbol{x})$ the set of strings that can be obtained by deleting a suffix of length at most j from some $\boldsymbol{v} \in R(\mathcal{D}^{\leq p}(\boldsymbol{x}))$. Note that $\mathcal{D}^{\leq p}_{\leq j}(\boldsymbol{x}) \subseteq \mathrm{Irr}_q(*)$.

Lemma 17. *Let $\boldsymbol{x}$ be an irreducible string of length n and $\boldsymbol{r}$ any string such that $\boldsymbol{xr}$ is irreducible. Let $\boldsymbol{w} \in R(\mathcal{D}^{\leq p}(\boldsymbol{xr}))$ and $\boldsymbol{s}$ be the prefix of $\boldsymbol{w}$ of length $n-p\mathcal{L}$. Then $\boldsymbol{s} \in \mathcal{D}^{\leq p}_{\leq 2p\mathcal{L}}(\boldsymbol{x})$.*

The lemma is proved in Appendix C.

By choosing the first $n-p\mathcal{L}$ elements of $\boldsymbol{w} \in R(\mathcal{D}^{\leq p}(\boldsymbol{xr}))$, we find $\boldsymbol{s} \in \mathcal{D}^{\leq p}_{\leq 2p\mathcal{L}}(\boldsymbol{x})$, which is a function of only $\boldsymbol{x}$ rather than $\boldsymbol{xr}$. But in doing so, we have introduced an additional error, namely deleting a suffix of length at most $2p\mathcal{L}$. As a result, we need to replace the labeling function f with a stronger labeling function f' that, in addition to handling both substitutions and duplications, can handle deleting a suffix of $\boldsymbol{x}$. More precisely, f' is a labeling function for the confusable set

$$B^{\leq p,\leq 2p\mathcal{L}}_{\mathrm{Irr}}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathrm{Irr}_q(n) : \boldsymbol{y} \neq \boldsymbol{x}, \mathcal{D}^{\leq p}_{\leq 2p\mathcal{L}}(\boldsymbol{x}) \cap \mathcal{D}^{\leq p}_{\leq 2p\mathcal{L}}(\boldsymbol{y}) \neq \varnothing\}. \tag{6}$$

The details of determining f' will be discussed in Section IV-C. Assuming the existence of the labeling function, $\boldsymbol{r}$ encodes $(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$, where for $\boldsymbol{x} \in \mathrm{Irr}_q(\boldsymbol{x})$, $a'_{\boldsymbol{x}}$ is chosen such that

$$f'(\boldsymbol{x}) \not\equiv f'(\boldsymbol{y}) \bmod a'_{\boldsymbol{x}}, \forall \boldsymbol{y} \in B^{\leq p,\leq 2p\mathcal{L}}_{\mathrm{Irr}}(\boldsymbol{x}).$$

To address the second difficulty raised above, i.e., protecting the information encoded in $\boldsymbol{r}$, we use an auxiliary high-redundancy code given in Section V. The following lemma, which is proved in Subsection V-C, provides an encoder for this code.

Lemma 18. *Let $\boldsymbol{\sigma} = 01020$. There exists an encoder $\mathcal{E}_1 : \Sigma_2^L \to \mathrm{Irr}_q(L')$ such that i) $\boldsymbol{\sigma}\mathcal{E}_1(\boldsymbol{u}) \in \mathrm{Irr}_q(*)$ and ii) for any string $\boldsymbol{x} \in \mathrm{Irr}_q(*)$ with $\boldsymbol{x\sigma}\mathcal{E}_1(\boldsymbol{u}) \in \mathrm{Irr}_q(*)$, we can recover $\boldsymbol{u}$ from any $\boldsymbol{w} \in R(\mathcal{D}^{\leq p}(\boldsymbol{x\sigma}\mathcal{E}_1(\boldsymbol{u})))$. Asymptotically, $L' \leq \frac{L}{\log(q-2)}(1+o(1))$.*

We use $\mathcal{E}_1(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$ to denote $\mathcal{E}_1(\boldsymbol{u})$, where $\boldsymbol{u}$ is a binary sequence representing the pair $(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$. For $\boldsymbol{x} \in \mathrm{Irr}_q(n)$, by letting $\boldsymbol{r} = \mathcal{E}_1(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$, we can construct codewords of the form $\boldsymbol{x\sigma r}$. But such codewords would not necessarily be irreducible. Irreducibility can be ensured by adding a buffer $\boldsymbol{b_x}$ between $\boldsymbol{x}$ and $\boldsymbol{\sigma r}$, as described by the next lemma, proved in Appendix D.

Lemma 19. *For $q \geq 3$ and any irreducible string $\boldsymbol{x}$ over Σ_q, there is a string $\boldsymbol{b_x}$ of length c_q such that $\boldsymbol{xb_x\sigma}$ is irreducible. Furthermore, $c_3 = 13$, $c_4 = 7$, $c_5 = 6$, and $c_q = 5$ for $q \geq 6$.*

The lemma implies that $\boldsymbol{xb_x\sigma r}$ is irreducible. This is because any substring of length at most 6 is either in $\boldsymbol{xb_x\sigma}$ or in $\boldsymbol{\sigma r}$ but cannot span both as $|\boldsymbol{\sigma}| = 5$. But $\boldsymbol{xb_x\sigma}$ and $\boldsymbol{\sigma r}$ are both irreducible, as shown in Lemma 19 and Lemma 18i, respectively.

We are now ready to present the code construction.

Construction B. *Let f' be a labeling function for the confusable sets $B^{\leq p,\leq 2p\mathcal{L}}_{\mathrm{Irr}}(\boldsymbol{x}), \boldsymbol{x} \in \mathrm{Irr}_q(n)$. Furthermore, for each $\boldsymbol{x}$, let $a'_{\boldsymbol{x}}$ be an integer such that $f'(\boldsymbol{x}) \not\equiv f'(\boldsymbol{y}) \bmod a'_{\boldsymbol{x}}$ for $\boldsymbol{y} \in B^{\leq p,\leq 2p\mathcal{L}}_{\mathrm{Irr}}(\boldsymbol{x})$. Let*

$$\mathcal{C}^B_n = \{\boldsymbol{xb_x\sigma r} : \boldsymbol{x} \in \mathrm{Irr}_q(n), \boldsymbol{r} = \mathcal{E}_1(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})\}.$$

Note that for simplicity, we index the code by the length of $\boldsymbol{x}$ rather than the length of the codewords $\boldsymbol{xb_x\sigma r}$, i.e., n in $\mathcal{C}^B_n$ refers to the length of $\boldsymbol{x}$. The length of $\boldsymbol{r}$ is discussed in Subsection IV-D below.

Theorem 20. *The code in Construction B can correct any number of short duplications and at most p substitutions.*

Proof: Let the retrieved word be $\boldsymbol{w} \in R(\mathcal{D}^{\leq p}(\boldsymbol{xb_x\sigma r}))$. From Lemma 18, given $\boldsymbol{w}$, we can find $a'_{\boldsymbol{x}}$ and $f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}}$. By Lemma 19, $\boldsymbol{xb_x\sigma r}$ is irreducible. Then, by Lemma 17, the $(n-p\mathcal{L})$-prefix of $\boldsymbol{w}$, denoted $\boldsymbol{s}$, satisfies $\boldsymbol{s} \in \mathcal{D}^{\leq p}_{\leq 2p\mathcal{L}}(\boldsymbol{x})$. By definition, for all $\boldsymbol{y} \neq \boldsymbol{x}$ that could produce the same $\boldsymbol{s}$, we have $\boldsymbol{y} \in B^{\leq p,\leq 2p\mathcal{L}}_{\mathrm{Irr}}(\boldsymbol{x})$. But then, $f'(\boldsymbol{y}) \not\equiv f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}}$, and so we can determine $\boldsymbol{x}$ by exhaustive search. ∎

B. Extension to edit errors

We now show that the codes in Constructions A and B are able to correct an arbitrary number of duplications and at most p *edit* errors, where an edit error may be a deletion, an insertion, or a substitution.

Define the DED(1) and DED(p) channels analogously to the DSD(1) and DSD(p) channels by replacing substitutions with edit errors. Any error-correcting code for a concatenation of p DED(1) channels is also an error-correcting code for DED(p).

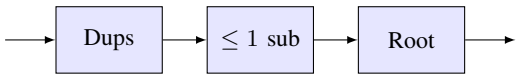
Additionally, any error-correcting code for a DSD(1) channel is also an error-correcting code for the DED(1) channel. This is because any input-output pair $(\boldsymbol{x}, \boldsymbol{y})$ for DED(1), shown in Figure 4b, is also an input-output pair for the DSD(1) channel, shown in Figure 4a. This claim is proved in [32, Corollary 12], where it was shown that a deletion can be represented as a substitution and a deduplication, e.g., $abc \to ac$ as $abc \to aac \to ac$, and an insertion as a duplication and a substitution, e.g., $abc \to abdc$ as $abc \to abbc \to abdc$.

Since $\mathcal{C}^A$ and $\mathcal{C}^B$ can correct errors arising from a concatenation of p DSD(1) channels, they can also correct errors arising from a concatenation of p DED(1) channels and thus a DED(p) channel, leading to the following theorem.

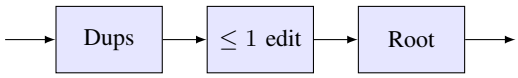
Theorem 21. *The codes in Constructions A and B can correct any number of duplications and at most p edit errors.*

C. The labeling function

In this subsection, we first present the labeling function f such that $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$ for $\boldsymbol{y} \in B^{\leq p}_{\mathrm{Irr}}(\boldsymbol{x})$, used in Construction A. By Theorem 13, $\boldsymbol{z} \in R(\mathcal{D}^{\leq p}(\boldsymbol{x})) \cap R(\mathcal{D}^{\leq p}(\boldsymbol{y}))$



(a) DSD(1) channel



(b) DED(1) channel

Figure 4: Any error-correcting code for channel (a) is also an error-correcting code for channel (b).

can be obtained from $\boldsymbol{x}$ and from $\boldsymbol{y}$ by at most $2p\mathcal{L}$ indels. Hence, it suffices to find f such that $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$ if there is a string $\boldsymbol{z}$ that can be obtained from both $\boldsymbol{x}$ and $\boldsymbol{y}$ through $2p\mathcal{L}$ indels. Note that since we are utilizing syndrome compression, choosing a more "powerful" labeling function does not increase the redundancy, which is still primarily controlled by $\max_{\boldsymbol{x}\in\mathrm{Irr}_q(n)} \|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\|$. We use the next theorem for binary sequences to find f.

Theorem 22. *[38] There exists a labeling function $g : \{0,1\}^n \to \Sigma_{2^{\mathcal{R}(t,n)}}$ such that for any two distinct strings $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ confusable under at most t insertions, deletions, and substitutions, we have $g(\boldsymbol{c}_1) \neq g(\boldsymbol{c}_2)$, where $\mathcal{R}(t,n) = [(t^2+1)(2t^2+1) + 2t^2(t-1)]\log n + o(\log n)$.*

Since $\boldsymbol{z} \in R(\mathcal{D}^{\leq p}(\boldsymbol{x}))$ can be obtained from $\boldsymbol{x}$ via at most $2p\mathcal{L}$ indels, $\mathcal{U}_i(\boldsymbol{z})$ can be derived from $\mathcal{U}_i(\boldsymbol{x})$ by at most $2p\mathcal{L}$ indels, for $i \in [\lceil \log q \rceil]$. Based on Theorem 22 and the work [38], by letting $t = 2p\mathcal{L}$, we can obtain a labeling function g for recovering $\mathcal{U}_i(\boldsymbol{x})$ from $\mathcal{U}_i(\boldsymbol{z})$ under at most $2p\mathcal{L}$ indels. Therefore, $f : \Sigma_q^n \to \Sigma_{2^{\lceil \log q \rceil \mathcal{R}(t,n)}}$,

$$f(\boldsymbol{x}) = \sum_{i=1}^{\lceil \log q \rceil} 2^{\mathcal{R}(t,n)(i-1)} g(\mathcal{U}_i(\boldsymbol{x})), \tag{7}$$

where $t = 2p\mathcal{L}$, is a labeling function for the confusable sets $B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$, $\boldsymbol{x} \in \mathrm{Irr}_q(n)$. For each $\boldsymbol{x}$, a value $a_{\boldsymbol{x}}$ needs to be also determined such that $f(\boldsymbol{x}) \not\equiv f(\boldsymbol{y}) \bmod a_{\boldsymbol{x}}$ for $\boldsymbol{y} \in B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})$. The existence of such $a_{\boldsymbol{x}}$, satisfying $\log a_{\boldsymbol{x}} \leq \log \|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\| + o(\log n)$, is guaranteed by Theorem 3 provided that p is a constant (ensuring that $p^4 = o(\log \log n)$). The labeling function f and integers $a_{\boldsymbol{x}}$ are used in Construction A.

In a similar manner, we can construct f' as a labeling function for $B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x}), \boldsymbol{x} \in \mathrm{Irr}_q(n)$ and integers $a'_{\boldsymbol{x}}$, by setting $t = 4p\mathcal{L}$ to account for the deletion of length at most $2p\mathcal{L}$. This time, for all $\boldsymbol{x} \in \mathrm{Irr}_q(n)$, $\log a'_{\boldsymbol{x}} \leq \log \|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| + o(\log n)$. The labeling function f' and integers $a'_{\boldsymbol{x}}$ are used in Construction B.

D. The redundancy of the error-correcting codes

In this section, we study the rate and the redundancy of the codes proposed in Constructions A and B and compare these to those of the state-of-the-art short-duplication-correcting code given in [10], which has the highest possible asymptotic rate. For an alphabet of size q, the asymptotic rate of this code for short duplications is $\log \lambda$, where λ is the largest positive real root of $x^3 - (q-2)x^2 - (q-3)x - (q-2) = 0$ [25].

The following lemma shows that the code proposed in [10] essentially has size $\mathrm{Irr}_q(N)$, where N is the length of the code, a fact that will be helpful for comparing the redundancies of the codes proposed here with this baseline.

Lemma 23. *Let $\mathcal{C}_N^D$ be the code of length N over alphabet Σ_q introduced by [10] for correcting any number of duplication errors. For $q \geq 4$,*

$$\|\mathrm{Irr}_q(N)\| \leq \|\mathcal{C}_N^D\| \leq \frac{q-2}{q-3}\|\mathrm{Irr}_q(N)\|.$$

Proof: As shown in [10], $\|\mathcal{C}_N^D\| = \sum_{i=1}^N \|\mathrm{Irr}_q(i)\|$. Based on [32, Lemma 14], given $\boldsymbol{u} \in \mathrm{Irr}_q(N-1)$, there are at least $q-2$ choices for $a \in \Sigma_q$ such that $\boldsymbol{x} = \boldsymbol{u}a \in \mathrm{Irr}_q(N)$. Thus, $(q-2)\|\mathrm{Irr}_q(N-1)\| \leq \|\mathrm{Irr}_q(N)\|$ and, consequently, $\|\mathrm{Irr}_q(N-i)\| \leq \frac{\|\mathrm{Irr}_q(N)\|}{(q-2)^i}$. Then we have

$$\frac{\sum_{i=1}^N \|\mathrm{Irr}_q(i)\|}{\|\mathrm{Irr}_q(N)\|} \leq \sum_{j=0}^{N-1} \frac{1}{(q-2)^j} \leq \frac{q-2}{q-3}. \qquad \blacksquare$$

We now compare the redundancy of the code $\mathcal{C}^A$ of Construction A with the code $\mathcal{C}^D$ of [10] for correcting only duplications. The length N of $\mathcal{C}_n^A$ is $N = n + |\boldsymbol{r}|$, where

$$\begin{aligned} |\boldsymbol{r}| &= 2\log_q a_{\boldsymbol{x}} \leq 2\log_q \|B_{\mathrm{Irr}}^{\leq p}(\boldsymbol{x})\| + o(\log_q n) \\ &\leq 4p\log_q n + o(\log_q n) \end{aligned} \tag{8}$$

symbols. Hence, $N = n + 4p\log_q n + o(\log_q n)$. Then, the difference in redundancies between $\mathcal{C}_n^A$ and $\mathcal{C}_N^D$, both of length N, is

$$\log_q \|\mathcal{C}_N^D\| - \log_q \|\mathcal{C}_n^A\| = \log_q \frac{\|\mathrm{Irr}_q(N)\|}{\|\mathrm{Irr}_q(n)\|} + O(1) \tag{9}$$

$$\leq \log_q q^{N-n} + O(1) \tag{10}$$

$$\leq 4p\log_q n + o(\log_q n), \tag{11}$$

where the equality follows from Lemma 23 and the first inequality from the fact that $\|\mathrm{Irr}_q(i+1)\| \leq q\|\mathrm{Irr}_q(i)\|$. Noting that $\log_q n = \log_q N + o(\log_q N)$ yields the following theorem.

Theorem 24. *For constants $q \geq 4$ and p, the redundancy of the code $\mathcal{C}_n^A$ of length N is larger than the redundancy of the code $\mathcal{C}_N^D$ of the same length by at most $4\log_q N + o(\log_q N)$.*

We now turn our attention to comparing the redundancy of $\mathcal{C}_n^B$ of length N with $\mathcal{C}_N^D$. Here, $N - n = |\boldsymbol{r}| + O(1) = |\mathcal{E}_1(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})| + O(1)$. Similar to (9), the extra redundancy is then $|\boldsymbol{r}| + O(1)$, which through $a'_{\boldsymbol{x}}$ depends on $\|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\|$, investigated in the next lemma. The proof of the lemma is in Appendix F.

Lemma 25. *For $\boldsymbol{x} \in \mathrm{Irr}_q(n)$ with $q \geq 3$,*

$$\|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| \leq q^{4p\mathcal{L}}(n + p\mathcal{L})^{2p}.$$

Lemma 26. *For constants $q \geq 4$ and p, and $\boldsymbol{x} \in \mathrm{Irr}_q(n)$, the length $|\boldsymbol{r}|$ of $\boldsymbol{r} = \mathcal{E}_1(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$ satisfies*

$$|\boldsymbol{r}| \leq 8p\log_q n + o(\log_q n).$$

Proof: From the previous subsection, assuming p is a constant, we have that $\log a'_{\boldsymbol{x}} \leq \log \|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| + o(\log n) \leq 2p\log n + o(\log n)$. Since $(f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}}) \leq a'_{\boldsymbol{x}}$, we need $4p\log n + o(\log n)$ bits to represent the pair $(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$. Then, by Lemma 18, $|\mathcal{E}_1(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}}))| \leq 4p\log n(1+o(1))/\log(q-2)$. The lemma follows from $\frac{\log q}{\log(q-2)} \leq 2$ for $q \geq 4$. ∎

Using Lemma 26, the next theorem gives the extra redundancy of correcting p substitutions compared to [10] and shows that there is no relative asymptotic rate penalty.

Theorem 27. *For constants $q \geq 4$ and p, the redundancy of the code $\mathcal{C}_n^B$ of length N is larger than the redundancy of the code $\mathcal{C}_N^D$ of the same length by at most $8\log_q N + o(\log_q N)$. The codes have the same asymptotic rate, which, for $q = 4$, equals $\log 2.6590$.*

E. Time complexity of encoding and decoding

Suppose $q \geq 4$ is a constant. The time complexities of both the encoding and decoding processes are polynomial in the lengths of the stored and retrieved sequences, respectively. The encoding process consists of four main parts:

1) Generating $\boldsymbol{x} \in \mathrm{Irr}_q(n)$ by the state-splitting algorithm, which has polynomial-time complexity [25].
2) Determining $\boldsymbol{b}_{\boldsymbol{x}}$ such that $\boldsymbol{x}\boldsymbol{b}_{\boldsymbol{x}}\boldsymbol{\sigma} \in \mathrm{Irr}_q(*)$, which has constant time complexity as the relevant subgraph of the De Bruijn graph (see Appendix D) has a constant size (no more than q^5 vertices).
3) Determining $a'_{\boldsymbol{x}}$ and $f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}}$. This is done in three steps, with polynomial time complexity. i) Given $\boldsymbol{x} \in \mathrm{Irr}_q(n)$, we find the elements of a set $\hat{B} \supseteq B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})$ whose size satisfies the upper bound given in Lemma 25. Specifically, given $\boldsymbol{x}$ we find all sequences that can be obtained from it through $\leq p$ *short substring substitutions*, one deletion of a suffix of length $\leq 2p\mathcal{L}$, one insertion of a suffix of length $\leq 2p\mathcal{L}$, and another $\leq p$ short substring substitutions, where in each short substring substitution step, we replace a substring $abcde \in \mathrm{Irr}_q(5)$ by another irreducible substring from the set $R(\mathcal{D}^1(abcde))$ and then deduplicate all copies. The total time complexity of this step is $O(n^{2p})$ as each element of $\hat{B}$ is obtained by a bounded number of operations and $\|\hat{B}\| = O(n^{2p})$. ii) Since computing $f'(\cdot)$ from [38] has time complexity $O(n\log n)$, computing it for all elements of $\hat{B}$ takes $O(n^{3p}\log n)$ steps. iii) Computing the remainder of these values modulo the $\leq 2^{\log O(n^{2p})}$ possible values for $a'_{\boldsymbol{x}}$ also has polynomial complexity.
4) Generating $\boldsymbol{r} = \mathcal{E}_1(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$ using the encoder $\mathcal{E}_1$ for the code in Construction E, which has complexity polynomial in $|\boldsymbol{r}|$ based on Subsection V-D. Hence, by Lemma 26, the complexity is at most polynomial.

Therefore, when p is a constant, the time complexity of the encoding process is polynomial with respect to N (as well as n).

Decoding requires finding the root of the retrieved word, which is linear in its length; decoding $a'_{\boldsymbol{x}}$ and $f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}}$, which is polynomial as discussed in Subsection V-D; and determining $\boldsymbol{x}$ through a brute-force search among all inputs that can lead to the same $(n-p\mathcal{L})$-prefix of the root of the retrieved sequence. Similar to the discussion about finding $\hat{B}$ above, the brute-force search is polynomial in n. Hence, decoding is polynomial in the length of the retrieved sequence.

V. Auxiliary High-Redundancy Error-Correcting Codes

Based on lemma 18 in Section IV, the error-correcting codes for short duplications and at most p substitutions with low redundancy rely on an error-correcting code to protect the syndrome information $(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$, where $(a'_{\boldsymbol{x}}, f'(\boldsymbol{x}) \bmod a'_{\boldsymbol{x}})$ is considered as a binary sequence. Therefore, this section focuses on constructing error-correcting codes that can protect this information from short duplications and at most p substitutions. We will also present the rate of the proposed codes in Section V-B, followed by the proof of Lemma 18 used in the previous section.

While in the previous section, we used syndrome compression with a labeling function designed to handle indel errors, in this section, the errors are viewed as substring edits in irreducible sequences, as described in Theorem 13. An example for Theorem 13 is given in Appendix E.

A. Code construction

To construct codes correcting at most p $\mathcal{L}$-substring edits in irreducible sequences, similar to [32], we divide the codewords into message blocks, separated by markers, while maintaining irreducibility, such that an $\mathcal{L}$-substring edit only affects a limited number of message blocks. In the case of $p = 1$ studied in [32], it was shown that if the markers appear in the correct positions in the retrieved word, then at most two of the message blocks are substituted. For $p > 1$ however, even if all markers are in the correct positions, it is not guaranteed that a limited number of message blocks are substituted, making it challenging to correct more than one error.

We start by recalling an auxiliary construction from [32].

Construction C. *[32, Construction 6] Let l, m, N_B be positive integers with $m > l \geq 5$ and $\boldsymbol{\sigma} \in \mathrm{Irr}_q(l)$. Also, let $\mathcal{B}_{\boldsymbol{\sigma}}^m$ denote the set of sequences B of length m such that $\boldsymbol{\sigma}B\boldsymbol{\sigma}$ is irreducible and has exactly two occurrences of $\boldsymbol{\sigma}$. Define*

$$\mathcal{C}_{\boldsymbol{\sigma}} = \{B_1\boldsymbol{\sigma}B_2\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_{N_B} : B_i \in \mathcal{B}_{\boldsymbol{\sigma}}^m\}.$$

The irreducibility of $\boldsymbol{\sigma}B_i\boldsymbol{\sigma}$ ensures that the codewords are irreducible.

We denote the output of the channel by $\boldsymbol{y}$. Define a *block* in $\boldsymbol{y}$ as a maximal substring that does not overlap with any $\boldsymbol{\sigma}$. Furthermore, define an *m-block* in $\boldsymbol{y}$ as a block of length m. Note that m-blocks can be either message blocks in $\boldsymbol{x}$ or new blocks created by substring edits.

Having divided each codeword into N_B message blocks and $N_B - 1$ separators, we study in the next lemma how message blocks are affected by the errors.

Lemma 28. *Let $\boldsymbol{x} \in \mathcal{C}_{\boldsymbol{\sigma}}$, $m > \mathcal{L}$, and $\boldsymbol{y}$ be generated from $\boldsymbol{x}$ through at most p $\mathcal{L}$-substring edits. Then there are less*

than (N_B+p) m-blocks in $\boldsymbol{y}$. Furthermore, there are at least N_B-2p error-free m-blocks in $\boldsymbol{y}$ which appear in $\boldsymbol{x}$ in the same order. More precisely, there are blocks $B_{i_1}, B_{i_2}, \ldots, B_{i_k}$ in $\boldsymbol{y}$, where $k \geq N_B-2p$, each B_{i_j} is a message block in $\boldsymbol{x}$, and any two blocks B_{i_j} and $B_{i_{j'}}$ have the same relative order of appearance in $\boldsymbol{x}$ and in $\boldsymbol{y}$.

Proof: First suppose $\boldsymbol{y}$ has $\geq (N_B+p)$ message blocks. This implies that the length of $\boldsymbol{y}$ is at least $(N_B+p)m+(N_B+p-1)l$, which is larger than the length of $\boldsymbol{x}$ by $pm+(p-1)l$. But this is not possible as $m > \mathcal{L}$ and the total length of inserted substrings is at most $p\mathcal{L}$.

Furthermore, if $m > \mathcal{L}$, each $\mathcal{L}$-substring edit alters i) a message block in $\boldsymbol{x}$, ii) a message block and a marker $\boldsymbol{\sigma}$, or iii) two message blocks and the marker between them. Hence at least N_B-2p message blocks of $\boldsymbol{x}$ appear in $\boldsymbol{y}$ without being changed. ∎

If the positions of the error-free m-blocks described in Lemma 28 in $\boldsymbol{y}$ were known, a Reed-Solomon (RS) code of length N_B and dimension N_B-2p could be used to recover codewords in $\mathcal{C}_{\boldsymbol{\sigma}}$. This however is not the case since the blocks can be shifted by substring edits. In order to determine the positions of the error-free m-blocks, we introduce another auxiliary construction based on Construction C by combining message blocks into *message groups*, where the message blocks in each group have different "colors".

Construction D. *For an integer T, we partition $\mathcal{B}^m_{\boldsymbol{\sigma}}$ into T parts $\mathcal{B}^m_{\boldsymbol{\sigma}}(j), j \in [T]$. The elements of $\mathcal{B}^m_{\boldsymbol{\sigma}}(j)$ are said to have* color *j. Let N_B be a positive integer that is divisible by T. We define the code*

$$\mathcal{C}_{(\boldsymbol{\sigma},T)} = \left\{B_1\boldsymbol{\sigma}B_2\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_{N_B} \in \mathcal{C}_{\boldsymbol{\sigma}} : B_i \in \mathcal{B}^m_{\boldsymbol{\sigma}}(i \bmod^+ T)\right\},$$

where $\mathcal{C}_{\boldsymbol{\sigma}}$ has parameters m, l with $m > \mathcal{L}$ and $m > l \geq 5$.

We divide the message blocks $B_1, \ldots, B_{N_B}$ in each $\boldsymbol{x} \in \mathcal{C}_{(\boldsymbol{\sigma},T)}$ into $\hat{N} = N_B/T$ message groups*, where the k-th message group is $S_k = (B_{(k-1)T+1}, \ldots, B_{kT-1}, B_{kT})$. Note that the message blocks in each message group have colors $1, 2, \ldots, T$ in order.*

For example, if $N_B = 12, T = 3, \hat{N} = 4$, then in a codeword

$$\boldsymbol{x} = B_1\boldsymbol{\sigma}B_2\boldsymbol{\sigma}B_3\boldsymbol{\sigma}B_4\boldsymbol{\sigma}B_5\boldsymbol{\sigma}B_6\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}B_{10}\boldsymbol{\sigma}B_{11}\boldsymbol{\sigma}B_{12},$$

the first group is (B_1, B_2, B_3) and the second group is (B_4, B_5, B_6). Furthermore, message blocks in both groups have colors $(1, 2, 3)$. The colors in the message group will help us identify the true positions of the message blocks.

Definition 29. *For $\boldsymbol{x} \in \mathcal{C}_{(\boldsymbol{\sigma},T)}$ and $\boldsymbol{y}$ derived from $\boldsymbol{x}$ through at most p $\mathcal{L}$-substring edits, let the i-th m-block in $\boldsymbol{y}$ be denoted by B'_i. A T-*group *in $\boldsymbol{y}$ is a substring $B'_{k+1}\boldsymbol{\sigma}B'_{k+2}\cdots\boldsymbol{\sigma}B'_{k+T}$ such that the m-block B'_{k+j} has color j.*

The next lemma characterizes how error-free message groups (those that do not suffer any substring edits but may be shifted) appear in $\boldsymbol{y}$.

Lemma 30. *Suppose $\boldsymbol{x} \in \mathcal{C}_{(\boldsymbol{\sigma},T)}$ and let $\boldsymbol{y}$ be obtained from $\boldsymbol{x}$ through at most p $\mathcal{L}$-substring edits. For $r \in [\hat{N}]$, if the r-th message group in $\boldsymbol{x}$ is not affected by any substring edit errors, then it will appear as a T-group after b m-blocks in $\boldsymbol{y}$, where $b \in [(r-1)T-2p, (r-1)T+p-1]$.*

Proof: Since $m > \mathcal{L}$, each $\mathcal{L}$-substring edit can affect at most two message blocks and thus at most two message groups. Hence, there are at least $\hat{N}-2p$ message groups that do not suffer any substring edits.

Let the r-th message group S_r in $\boldsymbol{x}$ be free of substring edits. Given that the colors of its message blocks are not altered, it will appear as a T-group in $\boldsymbol{y}$. Since each substring edit alters at most two message blocks, among the $(r-1)T$ message blocks appearing before S_r in $\boldsymbol{x}$, at most $2p$ do not appear in $\boldsymbol{y}$. Furthermore, the substring edits add at most $p\mathcal{L}$ to the length of $\boldsymbol{x}$. Since $m > \mathcal{L}$, this means that at most $p-1$ new m-blocks are created in $\boldsymbol{y}$. Hence, $b \in [(r-1)T-2p, (r-1)T+p-1]$. ∎

The previous lemma guarantees the presence of error-free, but possibly shifted, T-groups, and provides bounds on their position in $\boldsymbol{y}$. In the following theorem, we use these facts to show that these T-groups can be synchronized and the errors can be localized.

Theorem 31. *Let $\mathcal{C}_{(\boldsymbol{\sigma},T)}$ be a code in Construction D and suppose $T \geq 3p$ and $\hat{N} \geq 4p+1$. There is a decoder* Dec *such that, for any $\boldsymbol{x} \in \mathcal{C}_{(\boldsymbol{\sigma},T)}$ and $\boldsymbol{y}$ derived from $\boldsymbol{x}$ through at most p $\mathcal{L}$-substring edits, $\boldsymbol{v} = \mathrm{Dec}(\boldsymbol{y})$ suffers at most t substitutions and e erasures of message groups, where $t+e \leq 2p$.*

Proof: We start by identifying all T-groups in $\boldsymbol{y}$. Note that no two T-groups can overlap. Let $\boldsymbol{v} = (S'_1, \ldots, S'_{\hat{N}})$ be the decoded vector, where S'_r is the decoded version of the message group S_r, determined as follows.

For $r = 1, \ldots, \hat{N}$:

1) If there exists a T-group $\mathcal{T}$ appearing after b message blocks such that $b \in [(r-1)T-2p, (r-1)T+p-1]$, then let $S'_r = \mathcal{T}$.
2) If such a T-group does not exist, let $S'_r = \Lambda$, denoting an erasure.

We note that for each r, at most one T-group may satisfy the condition in 1). If two such T-groups exist appearing after b and b' message blocks, we must have $|b-b'| \geq T$ and $b, b' \in [(r-1)T-2p, (r-1)T+p-1]$, implying $3p-1 \geq T$, which contradicts the assumption on T.

If a message group S_r is not subject to a substring edit, then by Lemma 30, we have $S'_r = S_r$. Otherwise, we may have a substitution of that message group, i.e., $S'_r \neq S_r$, or an erasure, $S'_r = \Lambda$. Since each substring edit may affect at most 2 message groups, the total number of substitutions and erasures is no more than $2p$. ∎

We now construct an MDS code that can correct the output of the decoder of Theorem 31.

Construction E. *Let $\mathcal{C}_{(\boldsymbol{\sigma},T)}$ be the code in Construction D with parameters $l, m, T, \hat{N}$ satisfying $m > \mathcal{L}, m > l \geq 5, T \geq 3p$, and $\hat{N} \geq 4p+1$. Furthermore, assume $|\mathcal{B}^m_{\boldsymbol{\sigma}}(j)| \geq \hat{N}+1$ for $j \in [T]$. Finally, let γ be a positive integer such that $2^\gamma \leq \hat{N}+1$ and $\zeta_j : \mathbb{F}_{2^\gamma} \to \mathcal{B}^m_{\boldsymbol{\sigma}}(j)$ be an injective mapping*

for $j \in [T]$. We define $\mathcal{C}_E$ as

$$\begin{aligned}\mathcal{C}_E = \{&\zeta_1(c_1^1)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_j(c_1^j)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_T(c_1^T)\boldsymbol{\sigma}\\ &\zeta_1(c_2^1)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_j(c_2^j)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_T(c_2^T)\boldsymbol{\sigma}\cdots\\ &\zeta_1(c_{\hat{N}}^1)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_j(c_{\hat{N}}^j)\boldsymbol{\sigma}\cdots\boldsymbol{\sigma}\zeta_T(c_{\hat{N}}^T):\\ &\{\boldsymbol{c}^j, j\in[T]\}\subseteq \mathrm{MDS}(\hat{N},\hat{N}-4p,4p+1)\},\end{aligned}$$

where $\mathrm{MDS}(\hat{N},\hat{N}-4p,4p+1)$ denotes an MDS code over $\mathbb{F}_{2^\gamma}$ of length $\hat{N}=2^\gamma-1$, dimension $\hat{N}-4p$, and minimum Hamming distance $d_H=4p+1$, and $\boldsymbol{c}^j=(c_1^j,c_2^j,\ldots,c_{\hat{N}}^j)$ is a codeword of the MDS code.

For each j, we also define an inverse ζ_j^{-1} for ζ_j. For $B\in\mathcal{B}_{\boldsymbol{\sigma}}^m(j)$, if $\beta\in\mathbb{F}_{2^\gamma}$ such that $\zeta_j(\beta)=B$ exists, then let $\zeta_j^{-1}(B)=\beta$. Otherwise, let $\zeta_j^{-1}(B)=0$.

Theorem 32. *The error-correcting codes $\mathcal{C}_E$ in Construction E can correct any number of short duplications and at most p symbol substitutions.*

Proof: Given a codeword $\boldsymbol{x}\in\mathcal{C}_E$, let $\boldsymbol{x}''\in\mathcal{D}^{\le p}(\boldsymbol{x})$ and let $\boldsymbol{y}=R(\boldsymbol{x}'')$. Note that by construction, $\boldsymbol{x}$ is irreducible. Thus, by Theorem 13, $\boldsymbol{y}$ can be obtained from $\boldsymbol{x}$ through at most p $\mathcal{L}$-substring edits. As $\mathcal{C}_E\subseteq\mathcal{C}_{(\boldsymbol{\sigma},T)}$, based on Theorem 31, $\boldsymbol{v}=\mathrm{Dec}(\boldsymbol{y})$ suffers at most t substitutions and e erasures of message groups, where $t+e\le 2p$. Hence, for $j\in[T]$, the blocks $(\zeta_j(c_1^j),\zeta_j(c_2^j),\ldots,\zeta_j(c_{\hat{N}}^j))$ suffer at most $2p$ erasures or substitutions. Consequently, if we apply ζ_j^{-1} to the corresponding retrieved blocks in $\boldsymbol{v}$, the codeword $(c_1^j,c_2^j,\ldots,c_{\hat{N}}^j)$ also suffers at most $2p$ substitutions or erasures, which can be corrected using the MDS code. ■

B. Code rate

In this subsection, we present choices for the parameters of Construction E and discuss the rate of the resulting code.

Among the n_E symbols of each codeword in Construction E, $4pTm+(\hat{N}T-1)l$ symbols belong to MDS parities or markers. We choose T and l to be their smallest possible values and set $T=3p$ and $l=5$.

The construction requires that $\|\mathcal{B}_{\boldsymbol{\sigma}}^m(j)\|\ge\hat{N}+1$ for all j. Let $M_{\boldsymbol{\sigma}}^{(m)}=\|\mathcal{B}_{\boldsymbol{\sigma}}^m\|$. Dividing $\mathcal{B}_{\boldsymbol{\sigma}}^m$ into parts of nearly equal sizes, we find that each part $\mathcal{B}_{\boldsymbol{\sigma}}^m(j)$ has size at least $M_{\boldsymbol{\sigma}}^{(m)}/T-1$. We then choose $\hat{N}+1$ as the largest power of two not larger than $M_{\boldsymbol{\sigma}}^{(m)}/T-1$, ensuring that $\hat{N}+1\ge M_{\boldsymbol{\sigma}}^{(m)}/(2T)-(1/2)$. Assume

$$M_{\boldsymbol{\sigma}}^{(m)}\ge 24p^2+15p. \tag{12}$$

Then $\hat{N}+1\ge M_{\boldsymbol{\sigma}}^{(m)}/(2T)-(1/2)\ge 4p+2$.

Furthermore, note that $\hat{N}T(m+5)-5=n_E$ and thus $\hat{N}=\frac{n_E+5}{(m+5)(3p)}$. The size of the code then becomes

$$\|\mathcal{C}_E\|=(\hat{N}+1)^{(\hat{N}-4p)(3p)},$$

and

$$\begin{aligned}\log\|\mathcal{C}_E\| &\ge \left(\frac{n_E}{m+5}-12p^2\right)\log\left(\frac{M_{\boldsymbol{\sigma}}^{(m)}}{6p}-\frac{1}{2}\right)\\ &\ge \left(\frac{n_E}{m+5}-12p^2\right)\left(\log M_{\boldsymbol{\sigma}}^{(m)}+\log\left(\frac{1}{6p}-\frac{1}{2M_{\boldsymbol{\sigma}}^{(m)}}\right)\right)\\ &\ge \left(\frac{n_E}{m+5}-12p^2\right)\left(\log M_{\boldsymbol{\sigma}}^{(m)}-\log(6p+1)\right),\end{aligned} \tag{13}$$

where in the last step we have used the fact that $M_{\boldsymbol{\sigma}}^{(m)}\ge 24p^2+15p$.

It was shown in [32] that $M_{\boldsymbol{\sigma}}^{(m)}\ge(q-2)^{m-c_q}$ for some $\boldsymbol{\sigma}$, where c_q is a constant independent of m. In particular, $c_3\le 13, c_4\le 7, c_5\le 6$, and $c_q\le 5$ for $q\ge 6$. To satisfy (12), we need

$$m\ge\max\{\log_{q-2}(24p^2+15p)+c_q,\mathcal{L}+1\}. \tag{14}$$

From (13), for the rate of $\mathcal{C}_E$,

$$\begin{aligned}\frac{\log\|\mathcal{C}_E\|}{n_E} &\ge \left(\frac{m-c_q}{m+5}-\frac{12p^2m}{n_E}\right)\log(q-2)-\frac{\log(6p+1)}{m+5}\\ &\ge \left(1-\frac{c_q+5}{m+5}-\frac{12p^2m}{n_E}\right)\log(q-2)-\frac{\log(6p+1)}{m+5},\end{aligned}$$

where m satisfies (14). For $\log p=o(\log n_E)$, letting $m=\Theta(\log n_E)$, we find that the rate asymptotically satisfies

$$\frac{\log\|\mathcal{C}_E\|}{n_E}\ge\log(q-2)(1-o(1)),$$

while the redundancy is at least $\Theta(n_E/\log n_E)$. We observe that a low redundancy and an asymptotic rate equal to that of $\mathrm{Irr}_q(n_E)$ is not guaranteed for $\mathcal{C}_E$, unlike $\mathcal{C}^B$, proposed in the previous section. However, $\mathcal{C}^B$ relies on $\mathcal{C}_E$ to protect its syndrome as stated in Lemma 18, whose proof is given in the next subsection.

C. Proof of Lemma 18

To simplify the proof, instead of directly proving Lemma 18, we prove the following lemma, which essentially reverses the sequences in Lemma 18. Since both duplication and deduplication are symmetric operations, the lemmas are equivalent.

Lemma 33. *Let $\boldsymbol{\sigma}=01020$. There exists an encoder $\mathcal{E}_1:\Sigma_2^L\to\mathrm{Irr}_q(L')$ such that i) $\mathcal{E}_1(\boldsymbol{u})\boldsymbol{\sigma}\in\mathrm{Irr}_q(*)$ and ii) for any string $\boldsymbol{x}\in\mathrm{Irr}_q(*)$ with $\mathcal{E}_1(\boldsymbol{u})\boldsymbol{\sigma}\boldsymbol{x}\in\mathrm{Irr}_q(*)$, we can recover $\boldsymbol{u}$ from any $\boldsymbol{w}\in R(\mathcal{D}^{\le p}(\mathcal{E}_1(\boldsymbol{u})\boldsymbol{\sigma}\boldsymbol{x}))$. Asymptotically, $L'\le L/\log(q-2)(1+o(1))$.*

Proof: Let $\boldsymbol{v}=\mathcal{E}_1(\boldsymbol{u})$ and $\boldsymbol{w}\in R(\mathcal{D}^{\le p}(\boldsymbol{v}\boldsymbol{\sigma}\boldsymbol{x}))$. Furthermore, let $\boldsymbol{s}$ be $|\boldsymbol{v}|-p\mathcal{L}$-prefix of $\boldsymbol{w}$. By Lemma 17, we have $\boldsymbol{s}\in\mathcal{D}^{\le p}_{\le 2p\mathcal{L}}(\boldsymbol{v})$. So $\boldsymbol{s}$ can be obtained from $\boldsymbol{v}$ through at most $3p$ $\mathcal{L}$-substring edits. So if we let $\mathcal{E}_1$ be an encoder for $\mathcal{C}_E$ designed to correct $3p$ substitution errors and an infinite number of duplications, we can recover $\boldsymbol{u}$ from $\boldsymbol{s}$. The rate of this encoder is lower bounded by $\log(q-2)(1+o(1))$. ■

D. Time complexity of encoding and decoding

In this subsection, we analyze the time complexities of both the encoding and decoding algorithms for the error-correcting code in Construction E. Recall that we choose T

to be a constant and choose $\hat{N} = \Theta(\|\mathcal{B}^m_{\boldsymbol{\sigma}}\|)$ thus satisfying $\log \hat{N} = \Theta(m)$. Also, note that $n_E = \Theta(\hat{N})$. Furthermore, we choose each part $\mathcal{B}^m_{\boldsymbol{\sigma}}(j)$ in the partition of $\mathcal{B}^m_{\boldsymbol{\sigma}}$ to be a contiguous block in the lexicographically sorted list of the elements of $\mathcal{B}^m_{\boldsymbol{\sigma}}$. So the complexity of computing the mapping ζ_j is polynomial in $\|\mathcal{B}^m_{\boldsymbol{\sigma}}\|$ and thus in $\hat{N}$.

We first discuss the complexity of the encoding. The complexity of producing the MDS codewords used in $\mathcal{C}_E$ is polynomial in $\hat{N}$. Mapping these to sequences in $\mathcal{B}^m_{\boldsymbol{\sigma}}$ is also polynomial in $\hat{N}$ as discussed in the previous paragraph. Hence, the encoding complexity is polynomial in $\hat{N}$ as well as in n_E.

Decoding can be performed as described in the proof of Theorem 32, using the decoder described in Theorem 31 and its proof. As the steps described in the proofs of these theorems are polynomial in the length of the received sequence, so is the time complexity of the decoding.

VI. Conclusion

We introduced codes for correcting any number of duplication and at most p edit errors simultaneously. Recall that the set of irreducible strings is a code capable of correcting short duplication errors. To additionally correct edit errors, we append to each irreducible sequence $\boldsymbol{x}$ of length n a vector generated through syndrome compression that enables us to distinguish confusable inputs. Given that edit and duplication errors manifest as substring edit errors, we designed a buffer and the auxiliary code in a way to enable us to recover the syndrome information from the received string. In each step of the construction, we carefully ensured that the resulting sequence is still irreducible. The additional redundancy compared to the codes correcting duplications only [10] is $8p(\log_q n)(1+o(1))$, with the number of edits p and the alphabet size q being constants and $q \geq 4$. This additional redundancy is at most a factor of 2 away from the lowest-redundancy codes for correcting p edits only [37] and a factor of 4 away from the GV bound given in Theorem 15. The encoding and decoding processes have polynomial time complexities. We focused on $q \geq 4$ as it includes the case with the most practical importance, i.e., $q = 4$. While not all the results of the paper are valid for $q = 3$ (e.g., the bound on L' in Lemma 18), we expect many of the ideas to be applicable to this case.

The codes proposed in this work correct a wide range of errors. However, the number of edit errors is limited to be a constant. An important and interesting open problem is extending the work to correct more edits, e.g., linear in the code length. Additionally, only duplications bounded in length by three can be corrected, due to the fact that such duplications result in a regular language. So a second future direction is extending the work to correct longer duplications.

References

[1] Y. Tang, H. Lou, and F. Farnoud, "Error-correcting codes for short tandem duplications and at most p substitutions," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 1835–1840.

[2] Y. Tang, S. Wang, R. Gabrys, and F. Farnoud, "Correcting multiple short-duplication and substitution errors," *ISIT2022*, vol. 1, pp. 1–6, 2022.

[3] S. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 1, no. 3, pp. 230–248, 2015.

[4] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Noise and uncertainty in string-duplication systems," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 3120–3124.

[5] S. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Scientific reports*, vol. 5, no. 1, pp. 1–10, 2015.

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

[7] M. Blawat, K. Gaedke, I. Huetter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, "Forward error correction for DNA data storage," *Procedia Computer Science*, vol. 80, pp. 1011–1022, 2016.

[8] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen *et al.*, "Random access in large-scale DNA data storage," *Nature biotechnology*, vol. 36, no. 3, pp. 242–248, 2018.

[9] H. H. Lee, R. Kalhor, N. Goela, J. Bolot, and G. M. Church, "Terminator-free template-independent enzymatic DNA synthesis for digital information storage," *Nature communications*, vol. 10, no. 1, pp. 1–12, 2019.

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

[11] S. L. Shipman, J. Nivala, J. D. Macklis, and G. M. Church, "CRISPR–Cas encoding of a digital movie into the genomes of a population of living bacteria," *Nature*, vol. 547, no. 7663, pp. 345–349, Jul. 2017.

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

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

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

[15] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2331–2351, 2020.

[16] K. Cai, Y. M. Chee, R. Gabrys, H. M. Kiah, and T. T. Nguyen, "Optimal codes correcting a single indel/edit for DNA-based data storage," *arXiv preprint arXiv:1910.06501*, 2019.

[17] O. Elishco, R. Gabrys, and E. Yaakobi, "Bounds and constructions of codes over symbol-pair read channels," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1385–1395, 2020.

[18] A. Lenz, Y. Liu, C. Rashtchian, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding for efficient DNA synthesis," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 2885–2890.

[19] R. Gabrys, S. Pattabiraman, and O. Milenkovic, "Mass error-correction codes for polymer-based data storage," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 25–30.

[20] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Coding for optimized writing rate in DNA storage," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 711–716.

[21] H. M. Kiah, T. Thanh Nguyen, and E. Yaakobi, "Coding for sequence reconstruction for single edits," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 676–681.

[22] Y. Yehezkeally and M. Schwartz, "Uncertainty of reconstructing multiple messages from uniform-tandem-duplication noise," in *IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 126–131.

[23] T. T. Nguyen, K. Cai, K. A. S. Immink, and H. M. Kiah, "Constrained coding with error control for DNA-based data storage," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 694–699.

[24] J. Sima, N. Raviv, and J. Bruck, "Robust indexing-optimal codes for DNA storage," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 717–722.

[25] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Efficient encoding/decoding of GC-balanced codes correcting tandem duplications," *IEEE Transactions on Information Theory*, vol. 66, no. 8, pp. 4892–4903, 2020.

[26] Y. Tang and F. Farnoud, "Correcting deletion errors in DNA data storage with enzymatic synthesis," in *2021 IEEE Information Theory Workshop (ITW)*, 2021, pp. 1–6.

[27] ——, "Error-correcting codes for noisy duplication channels," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3452–3463, 2021.

[28] S. Jain, F. Farnoud, and J. Bruck, "Capacity and expressiveness of genomic tandem duplication," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6129–6138, 2017.

[29] M. Kovačević, "On the maximum number of non-confusable strings evolving under short tandem duplications," *Problems of Information Transmission*, vol. 58, no. 2, pp. 111–121, 2022.

[30] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Deciding the confusability of words under tandem repeats in linear time," *ACM Transactions on Algorithms (TALG)*, vol. 15, no. 3, pp. 1–22, 2019.

[31] Y. Tang and F. Farnoud, "Error-correcting codes for short tandem duplication and substitution errors," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 734–739.

[32] ——, "Error-correcting codes for short tandem duplication and edit errors," *IEEE Transactions on Information Theory*, vol. 68, no. 2, pp. 871–880, 2022.

[33] ——, "Error-correcting codes for noisy duplication channels," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 140–146.

[34] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An introduction to coding for constrained systems," *Lecture notes*, 2001. [Online]. Available: http://cmrr-star.ucsd.edu/psiegel/book_draft/

[35] J. Sima, R. Gabrys, and J. Bruck, "Syndrome compression for optimal redundancy codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 751–756.

[36] J. Sima and J. Bruck, "On optimal k-deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3360–3375, 2020.

[37] J. Sima, R. Gabrys, and J. Bruck, "Optimal codes for the q-ary deletion channel," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 740–745.

[38] ——, "Optimal systematic t-deletion correcting codes," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 769–774.

APPENDIX A
PROOF OF LEMMA 9

Lemma 9. *Given $q \geq 3$, we have*

$$\max_{\boldsymbol{t}\in\Sigma_q^5} \|R(\mathcal{D}^1(\boldsymbol{t}))\| \leq \|R(\mathcal{D}^1(01234))\|,$$

where $\mathcal{D}^1(01234) \subseteq \Sigma_{q+4}^*$ (the substituted symbol can be replaced with another symbol from Σ_{q+4}).

To prove Lemma 9, we start with the definition of dominance between two sequences from [32].

Definition 34. *Let $\boldsymbol{s}$ and $\bar{\boldsymbol{s}}$ be strings of length n, and let A be the set of symbols in $\boldsymbol{s}$ and $\bar{A}$ the set of symbols in $\bar{\boldsymbol{s}}$. We say that $\boldsymbol{s}$ dominates $\bar{\boldsymbol{s}}$ if there exists a function $\eta : A \to \bar{A}$ such that $\bar{\boldsymbol{s}} = \eta(\boldsymbol{s})$, where $\eta(\boldsymbol{s}) = \eta(s_1)\cdots\eta(s_n)$. Furthermore, a set U of strings dominates a set T if there is a single mapping η such that for each string $\boldsymbol{t} \in T$ there is a string $\boldsymbol{u} \in U$ such that $\boldsymbol{t} = \eta(\boldsymbol{u})$.*

For example, 0102 dominates 1212 (using the mapping $\eta(0) = 1, \eta(1) = 2, \eta(2) = 2$) but 0102 does not dominate 0010. The string $012\cdots k$ dominates any string of length $k+1$.

We recall an auxiliary lemma showing properties of dominance from [32], along with two other auxiliary lemmas that are used to simplify the proof of Lemma 9.

Lemma 35. *([32, Lemma 1]) Assume there are two strings $\boldsymbol{s}, \bar{\boldsymbol{s}}$ with $\boldsymbol{s}$ dominating $\bar{\boldsymbol{s}}$.*

1) *Suppose we apply the same duplication in both $\boldsymbol{s}$ and $\bar{\boldsymbol{s}}$ (that is, in the same position and with the same length). Let the resulting strings be $\boldsymbol{s}'$ and $\bar{\boldsymbol{s}}'$, respectively. Then $\boldsymbol{s}'$ dominates $\bar{\boldsymbol{s}}'$.*
2) *If a deduplication is possible in $\boldsymbol{s}$, a deduplication in the same position and with the same length is possible in $\bar{\boldsymbol{s}}$. Let the result of applying this deduplication to $\boldsymbol{s}$ and $\bar{\boldsymbol{s}}$ be denoted by $\boldsymbol{s}'$ and $\bar{\boldsymbol{s}}'$, respectively. Then $\boldsymbol{s}'$ dominates $\bar{\boldsymbol{s}}'$.*

Lemma 36. *Let $\bar{\boldsymbol{s}}$ be a string over $\bar{\Sigma}$ and $\boldsymbol{s}$ a string over Σ such that $\boldsymbol{s}$ dominates $\bar{\boldsymbol{s}}$. Let the number of distinct symbols in $\bar{\boldsymbol{s}}$ and $\boldsymbol{s}$ be denoted $\bar{q}_s$ and q_s, respectively, and suppose $\|\Sigma\| \geq \|\bar{\Sigma}\| + (q_s - \bar{q}_s)$. Then $\mathcal{D}^p(\boldsymbol{s}) \subseteq \Sigma^*$ dominates $\mathcal{D}^p(\bar{\boldsymbol{s}}) \subseteq \bar{\Sigma}^*$. In other words, there is a mapping $\eta : \Sigma \to \bar{\Sigma}$ that for any $\bar{\boldsymbol{y}} \in \mathcal{D}^p(\bar{\boldsymbol{s}}) \subseteq \bar{\Sigma}^*$, there exists $\boldsymbol{y} \in \mathcal{D}^p(\boldsymbol{s}) \subseteq \Sigma^*$ such that $\bar{\boldsymbol{y}} = \eta(\boldsymbol{y})$.*

Before proving the lemma, we provide an example with multiple short duplications and a substitution error, where the duplicated substrings are marked with underlines and the substituted symbols are in red.

Let $\Sigma = \{0, 1, 2, 3, 4\}$ and $\bar{\Sigma} = \{0, 1, 2, 3\}$. Suppose $\boldsymbol{s} = 012$ and $\bar{\boldsymbol{s}} = 010$ with $q_s = 3$ and $\bar{q}_s = 2$. The mapping $\eta(0) = 0$, $\eta(1) = 1$, and $\eta(2) = 0$, shows that $\boldsymbol{s}$ dominates $\bar{\boldsymbol{s}}$, i.e., $\boldsymbol{s} = 012 \to \bar{\boldsymbol{s}} = 010$.

Let $\bar{\boldsymbol{y}}_1 = 010\underline{010010} \in \mathcal{D}(\bar{\boldsymbol{s}})$. Then there exists $\boldsymbol{y}_1 = 012\underline{012012} \in \mathcal{D}(\boldsymbol{s})$ dominating $\bar{\boldsymbol{y}}_1$, via the same mapping η.

Next, assume $\bar{\boldsymbol{y}}_2 = 010012010$ is generated from $\bar{\boldsymbol{y}}_1$ by a substitution $0 \to 2$. Then $\boldsymbol{y}_2 = 012013012$, obtained from $\boldsymbol{y}_1$ after a substitution $2 \to 3$ in the same position, dominates $\bar{\boldsymbol{y}}_2$, via the mapping η extended by $\eta(3) = 2$.

Proof of Lemma 36: Without loss of generality, assume that $\bar{\Sigma} = \{0, 1, \ldots, \|\bar{\Sigma}\| - 1\}$ and that the symbols appearing in $\bar{\boldsymbol{s}}$ are $0, 1, \ldots, \bar{q}_s - 1$, where $\bar{q}_s \leq \|\bar{\Sigma}\|$. Similar statements hold for $\Sigma, \boldsymbol{s}, q_s$. By assumption, there exists some mapping $\eta : \{0, \ldots, q_s - 1\} \to \{0, \ldots, \bar{q}_s - 1\}$ showing that $\boldsymbol{s}$ dominates $\bar{\boldsymbol{s}}$. Since $\|\Sigma\| - q_s \geq \|\bar{\Sigma}\| - \bar{q}_s$, we may extend η by mapping symbols in Σ not occurring in $\boldsymbol{s}$ to symbols in $\bar{\Sigma}$ not occurring in $\bar{s}$. Specifically, we assign $\eta(i) = i - (q_s - \bar{q}_s) \in \bar{\Sigma}$ for $i \in \{q_s, q_s + 1, \ldots, \|\Sigma\| - 1\} \subseteq \Sigma$ to construct $\eta : \Sigma \to \bar{\Sigma}$.

Let the sequence of errors transforming $\bar{\boldsymbol{s}}$ to $\bar{\boldsymbol{y}}$ be denoted by $\bar{T}_j, j = 1, \ldots, k$ and let $\bar{\boldsymbol{y}}_j = \bar{T}_j(\bar{\boldsymbol{y}}_{j-1})$ with $\bar{\boldsymbol{y}}_0 = \bar{\boldsymbol{s}}$ and $\bar{\boldsymbol{y}} = \bar{\boldsymbol{y}}_k$. We will find a corresponding sequence (T_j), where each T_j has the same type of error as $\bar{T}_j$, and define $\boldsymbol{y}_j = T_j(\boldsymbol{y}_{j-1})$. We prove that for each j, we have $\bar{\boldsymbol{y}}_j = \eta(\boldsymbol{y}_j)$. The claim holds for $j = 0$ by assumption. Suppose it holds for $j - 1$. We show that it also holds for j. If $\bar{T}_j$ is a duplication, by Lemma 35(1), then we choose T_j to be a duplication of the same length in the same position. If $\bar{T}_j$ substitutes some symbol in $\bar{\boldsymbol{y}}_{j-1}$ with $a \in \bar{\Sigma}$, then T_j substitutes the symbol in the same position in $\boldsymbol{y}_{j-1}$ with a symbol $b \in \Sigma$ such that $\eta(b) = a$. It then follows that $\bar{\boldsymbol{y}}_j = \eta(\boldsymbol{y}_j)$ for each $\bar{\boldsymbol{y}}_j$. Therefore, we have $\mathcal{D}^p(\boldsymbol{s}) \subseteq \Sigma^*$ dominates $\mathcal{D}^p(\bar{\boldsymbol{s}}) \subseteq \bar{\Sigma}^*$. ∎

Lemma 37. *If a set of strings Y dominates a second set $\bar{Y}$, then $||R(\bar{Y})|| \leq ||R(Y)||$.*

Proof: Suppose Y dominates $\bar{Y}$ via a mapping $\eta : \Sigma \to \bar{\Sigma}$. Then, for each $\bar{\boldsymbol{y}} \in \bar{Y}$, there exists some $\boldsymbol{y} \in Y$ such that

$\bar{\boldsymbol{y}} = \eta(\boldsymbol{y})$. For $\bar{\boldsymbol{y}} \in \bar{Y}$, define $\eta^{-1}(\bar{\boldsymbol{y}})$ as the lexicographically-smallest sequence among $\{\boldsymbol{y} \in Y : \eta(\boldsymbol{y}) = \bar{\boldsymbol{y}}\}$. Furthermore, define $Y' = \{\eta^{-1}(\bar{\boldsymbol{y}}) : \bar{\boldsymbol{y}} \in \bar{Y}\}$ and note that $Y' \subseteq Y$. With this definition, Y' dominates $\bar{Y}$ and η is a bijection between the two sets. We have $\|\bar{Y}\| = \|Y'\| \leq \|Y\|$. Also, as $Y' \subseteq Y$, we have $\|R(Y')\| \leq \|R(Y)\|$.

To prove the lemma, we show that $\|R(\bar{Y})\| \leq \|R(Y')\|$. It suffices to prove that if $\bar{\boldsymbol{y}}_1, \bar{\boldsymbol{y}}_2 \in \bar{Y}$ have distinct roots, then $\boldsymbol{y}_1, \boldsymbol{y}_2 \in Y'$, where $\boldsymbol{y}_1 = \eta^{-1}(\bar{\boldsymbol{y}}_1)$ and $\boldsymbol{y}_2 = \eta^{-1}(\bar{\boldsymbol{y}}_2)$, also have distinct roots.

Suppose, on the contrary, that $\boldsymbol{y}_1, \boldsymbol{y}_2$ do not have distinct roots, i.e., $R(\boldsymbol{y}_1) = R(\boldsymbol{y}_2)$. Let T_1 and T_2 represent the sequences of deduplications on $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ that produce their roots, i.e., $R(\boldsymbol{y}_1) = T_1(\boldsymbol{y}_1)$ and $R(\boldsymbol{y}_2) = T_2(\boldsymbol{y}_2)$. Based on the Lemma 35.2) above, there exist two corresponding sequences of deduplications $\bar{T}_1$ and $\bar{T}_2$ such that $\bar{T}_1(\bar{\boldsymbol{y}}_1) = \eta(R(\boldsymbol{y}_1))$ and $\bar{T}_2(\bar{\boldsymbol{y}}_2) = \eta(R(\boldsymbol{y}_2))$. If $R(\boldsymbol{y}_1) = R(\boldsymbol{y}_2)$, then $\bar{T}_1(\bar{\boldsymbol{y}}_1) = \bar{T}_2(\bar{\boldsymbol{y}}_2)$. But by the uniqueness of the root, $R(\bar{\boldsymbol{y}}_1) = R(\bar{T}_1(\bar{\boldsymbol{y}}_1))$ and $R(\bar{\boldsymbol{y}}_2) = R(\bar{T}_2(\bar{\boldsymbol{y}}_2))$. So $R(\bar{\boldsymbol{y}}_1) = R(\bar{\boldsymbol{y}}_2)$. But this contradicts the assumption. Hence, the roots of $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are distinct. ∎

With Lemma 36 and Lemma 37 in hand, we prove Lemma 9 in the following.

Proof of Lemma 9: Let $\boldsymbol{s} = 01234$. If $\boldsymbol{t}$ is the empty string, the claim is trivial. So in the rest of the proof, we assume $\boldsymbol{t}$ is not empty. Based on Definition 34, $\boldsymbol{s}$ dominates $\boldsymbol{t}$ for any $\boldsymbol{t} \in \Sigma_q^5 \setminus \{\Lambda\}$. Let q_t denote the number of distinct symbols in $\boldsymbol{t}$ and note that there are 5 distinct symbols in $\boldsymbol{s}$. By Lemma 36, with $p = 1$, $\mathcal{D}^1(\boldsymbol{s}) \subseteq \Sigma_{q+4}^*$ dominates $\mathcal{D}^1(\boldsymbol{t}) \subseteq \Sigma_q^*$ for any $\boldsymbol{t} \in \Sigma_q^5$ since $q + 4 \geq q + (5 - q_t)$ as $q_t \geq 1$. Applying Lemma 37 to $\mathcal{D}^1(\boldsymbol{s})$ and $\mathcal{D}^1(\boldsymbol{t})$ completes the proof. ∎

Appendix B
Proof of Lemma 10

Lemma 10. *Let U and V be the sets of labels of all paths from Start to any state and from any state to S_9, respectively, in the finite automaton of Figure 3. Then $\|U\| = \|V\|$ and $\|R(U)\| = \|R(V)\|$.*

Proof: Define $h(a) = 4 - a$ for $a \in \Sigma_5$ and $h(\boldsymbol{u}) = h(u_n)h(u_{n-1})\cdots h(u_1)$ for $\boldsymbol{u} \in \Sigma_5^n$. Furthermore, for $S \subseteq \Sigma_5^*$, define $h(S) = \{h(\boldsymbol{u}) : \boldsymbol{u} \in S\}$. Note that h is its own inverse. We claim that h has the following properties, to be proved later:

a) For $\boldsymbol{s}, \boldsymbol{t} \in \Sigma_5^*$, $\boldsymbol{s}$ is a prefix of $\boldsymbol{t}$ if and only if $h(\boldsymbol{s})$ is a suffix of $h(\boldsymbol{t})$.
b) For $\boldsymbol{t} \in \Sigma_5^*$, $\mathcal{D}(h(\boldsymbol{t})) = h(\mathcal{D}(\boldsymbol{t}))$.
c) For $S \subseteq \Sigma_5^*$, $R(h(S)) = h(R(S))$.

By definition, if $\boldsymbol{u} \in U$ then $\boldsymbol{u}$ is a prefix of some $\boldsymbol{x} \in \mathcal{D}(01234)$. Then, by Property a), $h(\boldsymbol{u})$ is a suffix of $h(\boldsymbol{x})$. By setting $\boldsymbol{t} = 01234$, it follows from Property b) that $\mathcal{D}(01234) = h(\mathcal{D}(01234))$, and thus $h(\boldsymbol{x}) \in \mathcal{D}(01234)$. Hence, $h(\boldsymbol{u})$ is in V. Similarly, we can show that if $\boldsymbol{v} \in V$, then $h(\boldsymbol{v}) \in U$. As h is its own inverse, we have $V = h(U)$ and $\|U\| = \|V\|$. Applying Property c) with $S = U$ yields $R(V) = h(R(U))$ and $\|R(V)\| = \|R(U)\|$.

We now prove Properties a)-c). Property a) follows from the definition of h. Property b) follows from the observation that if $\boldsymbol{x}'$ is obtained from $\boldsymbol{x}$ via a duplication, then $h(\boldsymbol{x}')$ can be obtained from $h(\boldsymbol{x})$ via a duplication, i.e., the relationship represented by h is maintained under duplication. To prove Property c), it suffices to show that $R(h(\boldsymbol{t})) = h(R(\boldsymbol{t}))$ for $\boldsymbol{t} \in \Sigma_5^*$, which holds as h is maintained under deduplication. ∎

Appendix C
Proof of Lemma 17

Lemma 17. *Let $\boldsymbol{x}$ be an irreducible string of length n and $\boldsymbol{r}$ any string such that $\boldsymbol{xr}$ is irreducible. Let $\boldsymbol{w} \in R(\mathcal{D}^{\leq p}(\boldsymbol{xr}))$ and $\boldsymbol{s}$ be the prefix of $\boldsymbol{w}$ of length $n - p\mathcal{L}$. Then $\boldsymbol{s} \in \mathcal{D}^{\leq p}_{\leq 2p\mathcal{L}}(\boldsymbol{x})$.*

Proof: Based on Theorem 13, $\boldsymbol{w}$ can be considered as being generated from $\boldsymbol{xr}$ by at most p $\mathcal{L}$-substring edits. Let j be the last symbol of $\boldsymbol{x}$ not affected by a substring edit (i.e., it is not deleted by a substring edit, but it may be shifted). Suppose $t \leq p$ substring edits occur before x_j and at most $p - t$ after x_j. Then, $j \in [n - (p - t)\mathcal{L}, n]$. The symbol x_j appears as the ith symbol of $\boldsymbol{w}$ for some $i \in [j - t\mathcal{L}, j + t\mathcal{L}]$. Then, $\boldsymbol{w}_{[i]} \in R(\mathcal{D}^t(\boldsymbol{x}_{[j]}))$. It follows that $\boldsymbol{v} \in R(\mathcal{D}^t(\boldsymbol{x}))$ for $\boldsymbol{v} = \boldsymbol{w}_{[i]}\boldsymbol{x}_{[j+1,n]}$. As $i \geq j - t\mathcal{L}$ and $j \geq n - (p - t)\mathcal{L}$, we have $n - p\mathcal{L} \leq i$. Hence, $\boldsymbol{s} = \boldsymbol{w}_{[n-p\mathcal{L}]}$ is a prefix of $\boldsymbol{w}_{[i]}$ and thus also a prefix of $\boldsymbol{v}$. Specifically, $\boldsymbol{s}$ can be obtained from $\boldsymbol{v}$ by a suffix deletion of length

$$\begin{aligned} |\boldsymbol{v}| - (n - p\mathcal{L}) &= i + (n - j) - (n - p\mathcal{L}) \\ &\leq n + t\mathcal{L} + (p - t)\mathcal{L} - (n - p\mathcal{L}) \\ &= 2p\mathcal{L}. \end{aligned}$$

As $\boldsymbol{v} \in \mathcal{D}^{\leq p}(\boldsymbol{x})$, we have $\boldsymbol{s} \in \mathcal{D}^{\leq p}_{\leq 2p\mathcal{L}}(\boldsymbol{x})$. ∎

Appendix D
Proof of Lemma 19

Lemma 19. *For $q \geq 3$ and any irreducible string $\boldsymbol{x}$ over Σ_q, there is a string $\boldsymbol{b}_{\boldsymbol{x}}$ of length c_q such that $\boldsymbol{xb}_{\boldsymbol{x}}\boldsymbol{\sigma}$ is irreducible. Furthermore, $c_3 = 13$, $c_4 = 7$, $c_5 = 6$, and $c_q = 5$ for $q \geq 6$.*

Before proving Lemma 19, we recall from [10] that $\mathrm{Irr}_q(*)$ is a regular language whose graph $G_q = (V_q, \xi_q)$ is a subgraph of the De Bruijn graph. The vertex set V_q consists of 5-tuples $a_1a_2a_3a_4a_5 \in \mathrm{Irr}_q(5)$ that do not have any repeats (of length at most 4). There is an edge from $a_1a_2a_3a_4a_5 \to a_2a_3a_4a_5a_6$ if $a_1a_2a_3a_4a_5a_6$ belongs to $\mathrm{Irr}_q(6)$. The label for this edge is a_6. The label for a path is the 5-tuple representing its starting vertex concatenated with the labels of the subsequent edges. The proof below is similar to that of [32, Theorem 15] and is presented here for completeness.

Proof: Given $\boldsymbol{x} \in \mathrm{Irr}_q(n)$ and $q \geq 3$, $\boldsymbol{x}$ can be represented by a path over the graph G_q, ending at the vertex $\boldsymbol{x}_{[n-4:n]}$. Furthermore, $\boldsymbol{\sigma} = 01020$ can be considered as a vertex in G_q since $\boldsymbol{\sigma} \in \mathrm{Irr}_q(5)$. Let us assume for the moment that $q \geq 6$. Based on [32, Lemma 14], each vertex has at least $q - 2$ outgoing edges. So from each vertex, there is at least one outgoing edge whose label is equal to either 3, 4, or 5. So, starting from $\boldsymbol{x}_{[n-4:n]}$, we may arrive at some vertex with label

$\boldsymbol{b_x} \in \{3,4,5\}^5$ in 5 steps. Furthermore, $\boldsymbol{b_x}\boldsymbol{\sigma}$ is irreducible as both $\boldsymbol{b_x}$ and $\boldsymbol{\sigma}$ are irreducible and have no symbols in common. Hence, there is a path of length 5 from $\boldsymbol{b_x}$ to $\boldsymbol{\sigma}$ in G_q. So there is a path in G_q with label $\boldsymbol{x}\boldsymbol{b_x}\boldsymbol{\sigma}$, implying that $\boldsymbol{x}\boldsymbol{b_x}\boldsymbol{\sigma}$ is irreducible. We further have $c_q = |\boldsymbol{b_x}| = 5$. For $q \in \{3,4,5\}$, we have verified computationally that, for any choice of $\boldsymbol{x}_{[n-4:n]}$, there exists a path from $\boldsymbol{x}_{[n-4:n]}$ to $\boldsymbol{\sigma}$ of length $c_q + 5$, with the value of c_q as given in the lemma. Denoting the label of this path as $\boldsymbol{b_x}\boldsymbol{\sigma}$ gives us the sequence $\boldsymbol{b_x}$ of length c_q, with $\boldsymbol{x}\boldsymbol{b_x}\boldsymbol{\sigma}$ being irreducible. ■

Appendix E
Example for Theorem 13

Theorem 13 (c.f. [32, Theorem 5]). *Given strings $\boldsymbol{x} \in \Sigma_q^n$ and $\boldsymbol{v} \in \mathcal{D}^{\leq p}(\boldsymbol{x})$, $R(\boldsymbol{v})$ can be obtained from $R(\boldsymbol{x})$ by at most p $\mathcal{L}$-substring edits, where $\mathcal{L} = 17$.*

The following example illustrates the theorem.

Example 38. *Let the alphabet be $\Sigma_4 = \{0,1,2,3\}$ and $p = 2$. We take the input $\boldsymbol{x}$ to be irreducible, i.e., $R(\boldsymbol{x}) = \boldsymbol{x}$. By passing through the channel, $\boldsymbol{x}$ suffers multiple duplications and 2 symbol substitutions, resulting in $\boldsymbol{y} \in \mathcal{D}^2(\boldsymbol{x})$. We show the difference between $R(\boldsymbol{x})$ and $R(\boldsymbol{y})$ for two possible input-output pairs. Below, substrings added via duplication are marked with underlines, while substituted symbols are red and bold.*

First, we provide an example where $R(\boldsymbol{y})$ can be obtained from $R(\boldsymbol{x})$ via non-overlapping substring edits:

$$\begin{aligned}
\boldsymbol{x} &= 3210313230121321,\\
\boldsymbol{y} &= 3213\underline{2}\mathbf{0}\underline{321}031\underline{31}321\underline{32}3\mathbf{2}121\underline{321}\underline{321},\\
R(\boldsymbol{x}) &= \underbrace{321}_{\boldsymbol{\alpha}_0}\underbrace{}_{\boldsymbol{\beta}_1}\underbrace{031}_{\boldsymbol{\alpha}_1}\underbrace{3230121}_{\boldsymbol{\beta}_2}\underbrace{321}_{\boldsymbol{\alpha}_2},\\
R(\boldsymbol{y}) &= \underbrace{321}_{\boldsymbol{\alpha}_0}\underbrace{32\mathbf{0}321}_{\boldsymbol{\beta}_1'}\underbrace{031}_{\boldsymbol{\alpha}_1}\underbrace{}_{\boldsymbol{\beta}_2'}\underbrace{321}_{\boldsymbol{\alpha}_2},
\end{aligned}$$

where the errors are $\boldsymbol{\beta}_1 = \Lambda \to \boldsymbol{\beta}_1'$ and $\boldsymbol{\beta}_2 \to \boldsymbol{\beta}_2' = \Lambda$.

In the second case, the two edits overlap, leading to a single substring substitution:

$$\begin{aligned}
\boldsymbol{x} &= 132031230,\\
\boldsymbol{y} &= 1323203\mathbf{2}13\underline{2}\mathbf{0}3212\underline{30}\underline{230}230,\\
R(\boldsymbol{x}) &= \underbrace{13203}_{\boldsymbol{\alpha}_0}\underbrace{}_{\boldsymbol{\beta}}\underbrace{1230}_{\boldsymbol{\alpha}_1}\\
R(\boldsymbol{y}) &= \underbrace{13203}_{\boldsymbol{\alpha}_0}\underbrace{\mathbf{2}132\mathbf{0}32}_{\boldsymbol{\beta}'}\underbrace{1230}_{\boldsymbol{\alpha}_1}.
\end{aligned}$$

Appendix F
Proof of Lemma 25

Lemma 25. *For $\boldsymbol{x} \in \mathrm{Irr}_q(n)$ with $q \geq 3$,*

$$\|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| \leq q^{4p\mathcal{L}}(n+p\mathcal{L})^{2p}.$$

Proof: The proof is similar to that of Theorem 14, but also takes into account the effect of the suffix deletions, as shown in Figure 5. We have

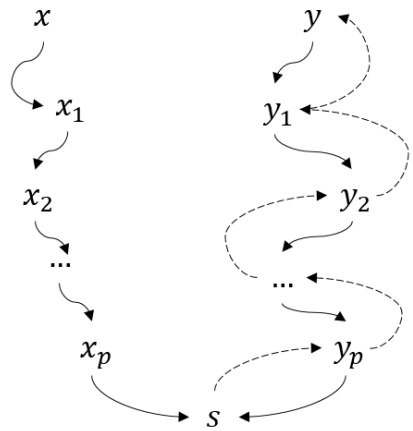


Figure 5: $\boldsymbol{s}$ results from passing $\boldsymbol{x}$ and $\boldsymbol{y}$ through a concatenation of p DSD(1) channels and a channel deleting a suffix of length at most $2p\mathcal{L}$ (c.f. Figure 2).

$$\begin{aligned}
\|B_{\mathrm{Irr}}^{\leq p,\leq 2p\mathcal{L}}(\boldsymbol{x})\| &\leq (968q(n+p\mathcal{L})+1)^{2p}(2p\mathcal{L}+1)(2p\mathcal{L}q^{2p\mathcal{L}}+1)\\
&\leq (2p\mathcal{L}+1)^2 q^{2p\mathcal{L}}(968q+1)^{2p}(n+p\mathcal{L})^{2p}\\
&\leq q^{4p\mathcal{L}}(n+p\mathcal{L})^{2p}.
\end{aligned}$$

In the first line, $(968q(n+p\mathcal{L})+1)^{2p}$ is derived based on Theorem 14; $(2p\mathcal{L}+1)$ bounds the number of ways $\boldsymbol{s}$ can be obtained from $\boldsymbol{x}_p$ through a suffix deletion of length at most $2p\mathcal{L}$; and $(2p\mathcal{L}q^{2p\mathcal{L}}+1)$ bounds the number of ways $\boldsymbol{y}_p$ can be obtained from $\boldsymbol{s}$ by appending a sequence of length at most $2p\mathcal{L}$. The third line is obtained by noting that $(968q+1)^{2p}(2p\mathcal{L}+1)^2 \leq q^{2p\mathcal{L}}$ with $\mathcal{L} = 17$. ■

Yuanyuan Tang (S'19) is a Ph.D. candidate in the Department of Electrical and Computer Engineering at the University of Virginia. His research interests consist of information theory, coding theory, wireless communications, and DNA data storage.

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

Shuche Wang (Graduate Student Member, IEEE) received the B.Eng. and M.Sc. degrees in information and communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2017 and 2020, respectively. He is currently pursuing a Ph.D. degree at the National University of Singapore. His research interests include coding theory in data storage systems and machine learning theory.

Hao Lou (S'18) received the bachelor's degree from Xi'an Jiaotong University, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Virginia. His research interests include data deduplication, stochastic and information-theoretic modeling of DNA mutations, compression of metagenomic sequencing data, computational biology, and machine learning.

Ryan Gabrys (Member, IEEE) received the B.S. degree in mathematics and computer science from the University of Illinois at Urbana-Champaing in 2005, and the Ph.D. degree in electrical engineering from the University of California, Los Angeles in 2014. He is currently a Scientist jointly affiliated with the Naval Information Warfare Center and the California Institute for Telecommunications and Information Technology (Calit2) at the University of California, San Diego. His research interests broadly lie in the areas of theoretical computer science and electrical engineering, including coding theory, combinatorics, and communication theory.

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

He received his MS degree in Electrical and Computer Engineering from the University of Toronto in 2008. From the University of Illinois at Urbana-Champaign, he received his MS degree in mathematics and his Ph.D. in Electrical and Computer Engineering in 2012 and 2013, respectively. His research interests include coding for storage, data compression, probabilistic modeling and analysis, and machine learning. He is the recipient of a 2022 Faculty Early Career Development Award (CAREER) from the National Science Foundation, the 2013 Robert T. Chien Memorial Award from the University of Illinois for demonstrating excellence in research in electrical engineering, and the 2014 IEEE Data Storage Best Student Paper Award.