

# Coding for Synthesis Defects

Ziyang Lu, Han Mao Kiah, Yiwei Zhang, Robert N. Grass, and Eitan Yaakobi

**Abstract**—Motivated by DNA based data storage system, we investigate the errors that occur when synthesizing DNA strands in parallel, where each strand is appended one nucleotide at a time by the machine according to a template supersequence. If there is a cycle such that the machine fails, then the strands meant to be appended at this cycle will not be appended, and we refer to this as a *synthesis defect*. In this paper, we present two families of codes correcting synthesis defects, which are *t-known-synthesis-defect correcting codes* and *t-synthesis-defect correcting codes*. For the first one, it is assumed that the defective cycles are known, and each of the codeword is a quaternary sequence. We provide constructions for this family of codes for  $t = 1, 2$ , with redundancy  $\log 4$  and  $\log n + 18 \log 3$ , respectively. For the second one, the codeword is a set of  $M$  ordered sequences, and we give constructions for  $t = 1, 2$  to show a strategy for constructing this family of codes. Finally, we derive a lower bound on the redundancy for single-known-synthesis-defect correcting codes, which assures that our construction is almost optimal.

## I. INTRODUCTION

Storing digital information on synthetic DNA strands has attracted significant interest due to its potential for information storage, particularly its durability and high storage density (see [1], [2] and references therein). The storage process involves converting binary digital information into quaternary strings (nucleotide bases) and writing these onto DNA strands using a synthesis machine.

While numerous experiments demonstrated the feasibility of DNA storage (see Table 1.1 in [2] for a recent survey), practical implementation for large-scale data remains challenging primarily due to its high cost. Specifically, DNA synthesis stands out as the most costly component in the storage model (see [3] for a discussion on synthesis procedures). Therefore, understanding the synthesis process is essential to enhance efficiency and reduce costs.

To minimize errors during the synthesis process, DNA strands typically contain no more than 250 nucleotides. Consequently, the user split the encoded quaternary strings into multiple short sequences for synthesis, storing them in an unordered manner. Synthesis of these multiple strands is typically array-based or performed in parallel (see for example, [3, Fig. 6]).

Ziyang Lu and Yiwei Zhang are with Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, 266237, China (e-mail: [zylu@mail.sdu.edu.cn](mailto:zylu@mail.sdu.edu.cn), [ywzhang@sdu.edu.cn](mailto:ywzhang@sdu.edu.cn)).

Han Mao Kiah is with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore (e-mail: [hmkiah@ntu.edu.sg](mailto:hmkiah@ntu.edu.sg)).

Robert N. Grass is with the Department of Chemistry and Applied Biosciences, ETH Zürich, Vladimir-Prelog-Weg 1-5, 8093 Zürich, Switzerland (e-mail: [robert.grass@chem.ethz.ch](mailto:robert.grass@chem.ethz.ch)).

Eitan Yaakobi is with the Department of Computer Science, Technion - Israel Institute of Technology, Haifa 3200003, Israel (e-mail: [yaakobi@cs.technion.ac.il](mailto:yaakobi@cs.technion.ac.il)).

In this process, the synthesis machine scans a fixed template DNA sequence and appends one nucleotide at a time according to this template supersequence. Each synthesis cycle involves appending a nucleotide to a subset of DNA strands requiring that specific nucleotide. Therefore, all synthesized DNA strands must be subsequences of the fixed template supersequence. The length of the fixed supersequence dictates the number of cycles needed for strand generation, thereby influencing the total synthesis time. Minimizing the total synthesis time is essential for reducing the overall cost of DNA synthesis, as each cycle requires reagents and chemicals [4]. This results in an interesting constrained coding problem that was initiated by Lenz *et al.* [5] and subsequently studied in [6]–[10].

In this paper, we introduce a new error model that arises during this synthesis process [11].

Specifically, if the synthesis machine fails to append a nucleotide at a certain cycle, then we refer to this error event as a *synthesis defect* (see Definition 1 for a formal definition). While synthesis defects share certain similarities to deletions, our primary contribution in this study is to demonstrate that by leveraging certain side information, we can dramatically reduce the redundancy of the coding scheme. Our results are distinguished between two scenarios: one where the defective cycles are known, and another where the defective cycles are unknown. In the first case, we construct codes that correct one and two synthesis defects using  $\log 4$  and  $\log n + O(1)$  redundant bits, respectively, where  $n$  denotes the codeword length. In contrast, if we employ a single and a two-deletion-correcting code, we require at least  $\log n$  and  $2 \log n$  redundant bits, respectively. Also, in this case where the defects are known and  $t = 1$ , we provide a matching lower bound for the redundancy. For the second case, where the defective cycles are not known, we consider the task of synthesizing  $M$  length- $n$  words. Here, we construct codes that correct one and two synthesis defects using roughly  $\lambda_1 (\log n)^2 + M \log \log n$  and  $\lambda_2 (\log n)^2 + 2M \log n$  redundant bits, respectively, for some constants  $\lambda_1$  and  $\lambda_2$ .

## II. PROBLEM FORMULATION

Throughout this paper, we use  $\Sigma = \{1, 2, 3, 4\}$  to denote DNA alphabet of size four and consider the shifted modulo operator so that  $(a \bmod 4)$  always belongs to  $\Sigma$ . For an integer  $T$ , we use  $[T]$  to denote the set  $\{1, 2, \dots, T\}$ . For a length- $n$  sequence  $\mathbf{x}$ , its  $i$ th symbol is denoted as  $x_i$ , and we use  $x_1 x_2 \dots x_n$  or  $(x_1, x_2, \dots, x_n)$  to represent it. Denote  $\mathbf{x}_I = x_{i_1} x_{i_2} \dots x_{i_k}$  as a subsequence of  $\mathbf{x}$  indexed by  $I$ , where  $I = \{i_1, i_2, \dots, i_k\} \subseteq [n]$  is a set of indices of size  $k$ .

Consider synthesizing a quaternary sequence  $\mathbf{x} = x_1 x_2 \dots x_n \in \Sigma^n$ . In this paper, we consider the template

sequence 12341234... and appends one nucleotide according to this sequence in each cycle. Thus, in almost all cases, more than  $n$  synthesis cycles are required as we need to wait for the synthesis machine to append the corresponding nucleotide.

Now, to determine the number of cycles, we perform the following computation. First, we define the *difference sequence* of  $\mathbf{x}$  to be  $\text{Diff}(\mathbf{x}) = (x_1, x_2 - x_1 \bmod 4, \dots, x_n - x_{n-1} \bmod 4)$ . Note that the map  $\text{Diff} : \Sigma^n \rightarrow \Sigma^n$  is a one-to-one mapping and hence it is invertible.

Next, we use  $\text{cycle}(\mathbf{x}) \in [4n]^n$  to denote the *synthesis cycle sequence* of  $\mathbf{x}$ . Here, for  $1 \leq i \leq n$ , the  $i$ th symbol of  $\mathbf{x}$  will be synthesized in the cycle given by  $\text{cycle}(\mathbf{x})_i = \sum_{j=1}^i \text{Diff}(\mathbf{x})_j$ . For example, if  $\mathbf{x} = 1241321$ , then  $\text{Diff}(\mathbf{x}) = 1121233$  and  $\text{cycle}(\mathbf{x}) = (1, 2, 4, 5, 7, 10, 13)$ . Observe that the elements of  $\text{cycle}(\mathbf{x})$  are necessarily monotonically increasing. Hence, by a slight abuse of notation, we treat  $\text{cycle}(\mathbf{x})$  both as a set and as a sequence. Specifically, sometimes, we write  $\Delta \cap \text{cycle}(\mathbf{x})$  to mean  $\Delta \cap \{\text{cycle}(\mathbf{x})_i\}_{i=1}^n$ .

Finally, for a subset of synthesis cycles  $\Delta \subseteq [4n]$ , we let  $I(\mathbf{x}, \Delta) = \{i \in [n] : \text{cycle}(\mathbf{x})_i \in \Delta\}$  represent the indices of  $\mathbf{x}$  whose synthesis cycles belong to  $\Delta$ . Then, we define  $\text{SynDef}_\Delta(\mathbf{x}) \triangleq \mathbf{x}_{[n] \setminus I(\mathbf{x}, \Delta)}$  to be the sequence obtained by deleting the symbols whose synthesis cycles belong to  $\Delta$ . Let us continue our example with  $\mathbf{x} = 1241321$ . When  $\Delta = \{12, 13\}$ , then  $I(\mathbf{x}, \Delta) = \{7\}$  and  $\text{SynDef}_\Delta(\mathbf{x}) = 124132$ .

#### A. Problem Formulation

In this paper, we consider the scenario where a set of DNA strands is synthesized in parallel. Similar to before, in every cycle, the machine appends the corresponding nucleotide to the strands that need it. However, if the machine does not append a nucleotide at cycle  $i$ , then we say that a *synthesis defect occurs at cycle  $i$* . Formally, we have the following definition.

**Definition 1.** Suppose that  $\mathbf{C} = (c_1, c_2, \dots, c_M) \in (\Sigma^n)^M$ . For  $\Delta \subseteq [4n]$ , we say that  $\mathbf{C}$  suffers from *synthesis defects* at the cycles in  $\Delta$  if for  $i \in [M]$ , the symbols of  $c_i$  synthesized at cycles  $\Delta$  are deleted and results in  $\text{SynDef}_\Delta(c_i)$ . Therefore, we define  $\text{SynDef}_\Delta(\mathbf{C}) \triangleq (\text{SynDef}_\Delta(c_1), \dots, \text{SynDef}_\Delta(c_M))$ .

For simplicity, we assume the strands are *ordered* and so, the identities of the strands are known to both the sender and receiver. In practice, this means that the strands are assigned unique indices and we assume that the indices are received error-free.

**Example 1.** Let  $M = 3$  and  $n = 5$ . Consider the set of strands  $\mathbf{C} = (c_1, c_2, c_3)$ , where

$$\begin{aligned} c_1 &= 31411 && (\text{cycle}(c_1) = (3, 5, 8, 9, 13)), \\ c_2 &= 12213 && (\text{cycle}(c_2) = (1, 2, 6, 9, 11)), \\ c_3 &= 14131 && (\text{cycle}(c_3) = (1, 4, 5, 7, 9)). \end{aligned}$$

When  $\Delta = \{1\}$ , we have that

$$\text{SynDef}_\Delta(\mathbf{C}) = (31411, 2213, 4131).$$

On the other hand, when  $\Delta = \{10\}$  or  $\Delta = \{12\}$ , we have

$$\text{SynDef}_\Delta(\mathbf{C}) = (31411, 12213, 14131).$$

Observe that when synthesis defects occur in a set of strands, not all strands are erroneous. However, those strands with symbols synthesized during corresponding cycles will be deleted at those cycles. Furthermore, as illustrated in this example, it is possible that  $\text{SynDef}_\Delta(\mathbf{C}) = \mathbf{C}$  and this poses an interesting coding challenge. ■

Our goal in this paper is to study codes correcting synthesis defects. We distinguish between two cases: one where we know the defective cycles; another where the locations of defects are unknown.

*Case I.* In the first case, even if we know the set of defective cycles, that is  $\Delta$ , it is possible that we cannot infer the erroneous positions from  $\text{SynDef}_\Delta(\mathbf{C})$ .

Let us explain using the following example.

**Example 2.** Let  $M = 3$  and  $n = 5$ . Consider the set of strands

$$\mathbf{C} = (12341, 12134, 21231).$$

When  $\Delta = \{5\}$ , we have that

$$\text{SynDef}_{\{5\}}(\mathbf{C}) = (1234, 1234, 2231).$$

Now we want to recover  $\mathbf{C}$  from  $\text{SynDef}_{\{5\}}(\mathbf{C})$  knowing that  $\Delta = \{5\}$ . For the strand 2231, we insert a 1 at the fifth cycle, and we get 21231. This turns out to be the only option. But for the strand 1234, there are four possible positions to insert 1 at the fifth cycle, which result in 11234, 12134, 12314, 12341. Therefore we cannot uniquely recover  $\mathbf{C}$  even if we know the locations of the defects. Furthermore, the erroneous positions corresponding to 1234 are 2, 3, 4, and 5. Hence, coding for erasures is insufficient. ■

The above example gives rise to the following problem: how to recover  $\mathbf{C}$  with the knowledge of locations of defects? In our study of this problem, it is enough to investigate the case where  $M = 1$ . In other words, we design codes for a single-strand. This is because in solving the single-strand case, we obtain sufficient information, including the approximate deleted locations and the values of the deleted symbols, and other strands in the set cannot provide more information about the errors.

Hence, a *t-known-synthesis-defect correcting code* allows one to uniquely recover a synthesized word, in the presence of  $t$  synthesis defects when the locations of defects are known.

**Definition 2.** We say a code  $\mathcal{C} \subseteq \Sigma^n$  is a *t-known-synthesis-defect correcting code (t-KDCC)* if for every pair of distinct codewords,  $c_1, c_2 \in \mathcal{C}$ , and for any  $\Delta \subseteq [4n]$  with  $|\Delta| = t$ , we have that

$$\text{SynDef}_\Delta(c_1) \neq \text{SynDef}_\Delta(c_2).$$

*Case II.* On the other hand, when we do not know the locations of the defects, coding for a single strand is almost equivalent to coding for deletions. Hence, we consider  $M > 1$ . When the locations of the defects are unknown, we can simply employ a deletion-correcting code for each strand. However, this incurs a high redundancy and hence, our goal is to reduce this redundancy.

Consider Example 1 and the case where  $\Delta = \{1\}$ . Suppose  $\mathbf{c}_2 = 12213$  is obtained from a single-deletion-correcting code. Then when we receive 2213, we are able to recover  $\mathbf{c}_2$  and also determine  $\Delta = \{1\}$ . Then we are able to use this information on  $\Delta$  to correct the other strings. Therefore, when the location of defects are unknown, our strategy is to divide the  $M > 1$  strands into two parts and employ two different coding schemes. The first coding scheme not only allows us to correct the defects, but also provide approximate locations of the defects. This then allows us to employ a second coding scheme that incurs less redundancy and we describe this in detail in Section IV. Here, we formally define a *t-synthesis-defect correcting code*.

**Definition 3.** For a set of sequences  $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_M) \in (\Sigma^n)^M$ , and  $d \in [n]$ , we define the *synthesis defect ball* of radius  $d$  of  $\mathbf{C}$  to be the set

$$\text{Ball}_d^{\text{SynDef}}(\mathbf{C}) = \{\text{SynDef}_\Delta(\mathbf{C}) : \Delta \subseteq [4n], |\Delta| \leq d\}.$$

**Definition 4.** A code  $\mathcal{C} \subseteq (\Sigma^n)^M$  is a *t-synthesis-defect correcting code (t-SDCC)* if for every pair of distinct  $\mathbf{C}_1, \mathbf{C}_2 \in \mathcal{C}$ , we have

$$\text{Ball}_t^{\text{SynDef}}(\mathbf{C}_1) \cap \text{Ball}_t^{\text{SynDef}}(\mathbf{C}_2) = \emptyset.$$

In this paper, we provide designs for these two classes of codes. We use redundancy to evaluate a code. Specifically, we define the *redundancy* of a code  $\mathcal{C} \subseteq (\Sigma^n)^M$  to be  $2Mn - \log|\mathcal{C}|$ .

## B. Organization

In Section III, we construct *t*-KDCCs for  $t = 1, 2$  with redundancy  $\log 4$  and  $\log n + 18 \log 3$  respectively. Specifically, we achieve this by constructing binary code capable of correcting *t* deletions where each deletion is within a small window. When the window is large, this will be an important material for constructing SDCC. In Section IV, we first show that using  $O(\log n)$  strands is enough to cover all the  $4n$  cycles. Then, by giving single-SDCC and 2-SDCC we show our idea for constructing *t*-SDCCs. In Section V, we provide the lower bounds for redundancy on 1-KDCCs, which is at least  $\log 4 - o(1)$  bits of redundancy. Last, we conclude and present some future work in Section VI.

## III. CONSTRUCTIONS OF KNOWN-SYNTHESIS-DEFECT CORRECTING CODES

In this section, we provide constructions of *t*-KDCCs for  $t = 1, 2$ . We show that knowing the locations of the defects helps us narrow each of the location of deletion into a small interval of constant length. Then, we can correct these deletions by using binary bounded-deletion-correcting codes [12].

**Definition 5.** For  $\mathbf{P} = (P_1, P_2, \dots, P_t) \in \mathbf{Z}^t$  we call a code *P-bounded t-deletion correcting code* if it can correct *t* deletions where the *t* deletions are located at *t* given intervals of length  $P_1, P_2, \dots, P_t$ , respectively. Furthermore, we let  $P \triangleq \max\{P_1, \dots, P_t\}$  denote the maximum  $P_i$  for  $1 \leq i \leq t$ .

Our main contribution in this section is to construct *P*-bounded *t*-deletion correcting codes with different ranges of *P* for  $t = 1, 2$ . Before that, let us introduce a serial of useful lemmas to show that the knowledge of defective cycles narrows down the locations of resulting deletions.

**Definition 6.** For  $\Delta \subseteq [4n]$  we define

$$B^\Delta(\mathbf{x}) = \{\mathbf{y} \in \Sigma^n : \text{SynDef}_\Delta(\mathbf{x}) = \text{SynDef}_\Delta(\mathbf{y})\}$$

as all the length-*n* words that will result in the same word after deleting the symbols located at these cycles in  $\Delta$ . Besides, if  $|\Delta| = 1$  and has only one element  $\delta \in [4n]$ , we also use the notation  $B^\delta(\mathbf{x})$  to represent the confusable ball of  $\mathbf{x}$  in case of the defect of cycle  $\delta$ .

For radius-1 confusable ball of  $\mathbf{x}$ , we have the following lemmas.

**Lemma 1.** For  $\mathbf{x} \in \Sigma^n$ , and  $\delta \in \text{cycle}(\mathbf{x})$ , we have

$$|B^\delta(\mathbf{x})| = |\{\delta - 4, \delta - 3, \delta - 2, \delta - 1\} \cap (\text{cycle}(\text{SynDef}_{\{\delta\}}(\mathbf{x})) \cup \{0\})|.$$

*Proof:* For every  $\mathbf{y} \in B^\delta(\mathbf{x})$ , it has  $\text{SynDef}_{\{\delta\}}(\mathbf{y}) = \text{SynDef}_{\{\delta\}}(\mathbf{x})$  by the definition of  $B^\delta(\mathbf{x})$ . Thus, we can generate  $B^\delta(\mathbf{x})$  by inserting the symbol  $(\delta \bmod 4)$  in  $\text{SynDef}_{\{\delta\}}(\mathbf{x})$  and make sure it is at the  $\delta$  cycle.

Due to the property of the cycle sequence, any two consecutive elements in a cycle sequence have difference at most 4. Consequently, if we want to insert a symbol making sure it is at the  $\delta$  cycle, then it must be inserted after the symbol synthesized at cycle  $\delta - 4$  or  $\delta - 3$  or  $\delta - 2$  or  $\delta - 1$ . So, the size of  $B^\delta(\mathbf{x})$  depends on  $\{\delta - 4, \delta - 3, \delta - 2, \delta - 1\}$  and  $\text{SynDef}_{\{\delta\}}(\mathbf{x})$ . Specially, if we insert  $(\delta \bmod 4)$  at the beginning of  $\text{SynDef}_{\{\delta\}}(\mathbf{x})$ , then we actually insert it after the cycle 0. Hence, we conclude that  $|B^\delta(\mathbf{x})| = |\{\delta - 4, \delta - 3, \delta - 2, \delta - 1\} \cap (\text{cycle}(\text{SynDef}_{\{\delta\}}(\mathbf{x})) \cup \{0\})|$ . ■

This leads to the following lemma, demonstrating the relationship between the indices of deletions in  $\mathbf{x}$  and those in its confusable sequence.

**Lemma 2.** If  $\mathbf{x} \in \Sigma^n$  suffers from *t* synthesis defects at cycles in  $\Delta = \{\delta_1, \dots, \delta_t\} \subseteq \text{cycle}(\mathbf{x})$  resulting in  $\text{SynDef}_\Delta(\mathbf{x})$ , then for any  $\mathbf{y} \in B^\Delta(\mathbf{x})$  with  $\text{cycle}(\mathbf{x})_{\{i_1, i_2, \dots, i_t\}} = \text{cycle}(\mathbf{y})_{\{j_1, j_2, \dots, j_t\}} = \Delta$ , we have  $|i_k - j_k| \leq 4k - 1$  for  $1 \leq k \leq t$ .

*Proof:* For any  $\mathbf{y} \in B^\Delta(\mathbf{x})$ ,  $\mathbf{y}$  can be obtained by inserting *t* symbols successively into  $\text{SynDef}_\Delta(\mathbf{x})$  in the order  $x_{i_1}, x_{i_2}, \dots, x_{i_t}$  and at positions  $j_1, j_2, \dots, j_t$ . By Lemma 1, there are at most 4 positions to insert  $x_{i_1}$  such that it is inserted at the cycle of  $\text{cycle}(\mathbf{x})_{i_1}$ , where the index is in the range  $[i_1 - 3, i_1 + 3]$ , so  $|i_1 - j_1| \leq 3$ . After inserting  $x_{i_1}$  into  $\text{SynDef}_\Delta(\mathbf{x})$ , each cycle of  $\text{SynDef}_\Delta(\mathbf{x})_i$  will increase by 4 or keep unchanged for  $i > i_1$ . Again there are at most 4 positions to insert  $x_{i_2}$  at the cycle of  $\text{cycle}(\mathbf{x})_{i_2}$ . Since the cycles may have increased by 4,  $x_{i_2}$  is possible to be inserted near the symbol synthesized at  $\text{cycle}(\mathbf{x})_{i_2} - 4$ , so  $|i_2 - j_2| \leq 7$ . Repeating this process, we conclude that  $|i_k - j_k| \leq 4k - 1$  for  $1 \leq k \leq t$ . ■

**Example 3.** Let  $x$  and  $\text{cycle}(x)$  be the following, and let  $\Delta = \{5, 17\}$ .

$$\begin{array}{cccccccccccccccccccc} \text{cycle}(x) & = & 1 & 2 & 3 & 4 & \underline{5} & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & \underline{17} & 18 \\ x & = & 1 & 2 & 3 & 4 & \underline{1} & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & \underline{1} & 2 \end{array}$$

Then, we have  $\text{SynDef}_\Delta(x) \triangleq x'$  as follows:

$$\begin{array}{cccccccccccccccccccc} \text{cycle}(x') & = & 1 & 2 & 3 & 4 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 18 \\ x' & = & 1 & 2 & 3 & 4 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 2 \end{array}$$

Now we want to insert two 1s into  $\text{SynDef}_\Delta(x)$  such that these two 1s are at 5th and 17th cycles. We first insert a 1 at the second position of  $\text{SynDef}_\Delta(x)$ , then we get  $x''$  as follows:

$$\begin{array}{cccccccccccccccccccc} \text{cycle}(x'') & = & 1 & \underline{5} & 6 & 7 & 8 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 22 \\ x'' & = & 1 & \underline{1} & 2 & 3 & 4 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 2 \end{array}$$

Next, to get a sequence in  $B^\Delta(x)$  we need to insert a 1 into  $x''$  such it is at 17th cycle. We insert it at the 10th position of  $x''$ , and we get  $y \in B^\Delta(x)$ :

$$\begin{array}{cccccccccccccccccccc} \text{cycle}(y) & = & 1 & \underline{5} & 6 & 7 & 8 & 10 & 11 & 12 & 13 & \underline{17} & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 26 \\ y & = & 1 & \underline{1} & 2 & 3 & 4 & 2 & 3 & 4 & 1 & \underline{1} & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 2 \end{array}$$

Compare  $y$  with  $x$ , and we have  $\text{cycle}(x)_{\{5,17\}} = \text{cycle}(y)_{\{2,10\}} = \{5, 17\} = \Delta$ , which satisfies the conclusion of Lemma 2 that  $|i_k - j_k| \leq 4k - 1$  for  $k = 1, 2$ . ■

#### A. Reduction to binary codes

In this subsection, we show that the constructions of  $t$ -KDCCs can be reduced to constructions of binary codes.

**Definition 7** ([13]). Let  $x \in \Sigma^n$  be a quaternary word of length  $n$ . Define the *signature* of  $x$  to be the binary sequence  $\tilde{x}$  of length  $n - 1$  such that

$$\tilde{x}_i = \begin{cases} 1 & \text{if } x_{i+1} \geq x_i, \\ 0 & \text{if } x_{i+1} < x_i, \end{cases}$$

for all  $1 \leq i \leq n - 1$ .

**Lemma 3.** If  $x \in \Sigma^n$  suffers from  $t$  synthesis defects at cycle  $\Delta = \{\delta_1, \dots, \delta_t\} \subseteq \text{cycle}(x)$  resulting in  $\text{SynDef}_\Delta(x)$ , then with the knowledge of  $\Delta$  and  $\tilde{x}$ , we can recover  $x$  from  $\text{SynDef}_\Delta(x)$ .

*Proof:* We prove this by giving an algorithm, where the inputs are  $\text{SynDef}_\Delta(x)$ ,  $\Delta = \{\delta_1, \dots, \delta_t\}$  and  $\tilde{x}$ , and output is  $x$ . First we recover the symbol synthesized at  $\delta_1$  cycle. By Lemma 1 and 2, there are at most 4 positions to insert the symbol  $(\delta_1 \bmod 4)$ . Specifically, it can only be inserted after symbols synthesized at cycles in  $\{\delta_1 - 4, \delta_1 - 3, \delta_1 - 2, \delta_1 - 1\}$ . Let  $I = \{i + 1 : \text{cycle}(x)_i \in \{\delta_1 - 4, \delta_1 - 3, \delta_1 - 2, \delta_1 - 1\}\}$ , then  $I$  is the set of indices suitable to insert  $(\delta_1 \bmod 4)$ . Furthermore,  $|I| \leq 4$  and the elements in  $I$  are consecutive. Since we can know the monotonicity of  $x_I$  from  $\tilde{x}$ , there is only one choice in  $I$  to insert  $(\delta_1 \bmod 4)$  keeping the monotonicity. Denote the sequence obtained by inserting  $(\delta_1 \bmod 4)$  into  $\text{SynDef}_\Delta(x)$  as  $x'$ , and renew the inputs as  $x'$ ,  $\Delta = \{\delta_2, \dots, \delta_t\}$  and  $\tilde{x}$ . After  $t$  rounds of algorithm, we can get the correct sequence  $x$ . ■

The following is an algorithm we mentioned above to recover  $x$  with the knowledge of the defective cycles and the signature.

**Algorithm 1:** Recover  $x$  with the knowledge of defective cycles and its signature

---

**Input:**  $\text{SynDef}_\Delta(x)$ ,  $\Delta = \{\delta_1, \dots, \delta_t\}$  and  $\tilde{x}$   
**Output:**  $x$   
Set  $j = 1$ ,  $x' = \text{SynDef}_\Delta(x)$ ;  
**while**  $\Delta \neq \emptyset$  **do**  
     $I \leftarrow \{i + 1 : \text{cycle}(x)_i \in \{\delta_j - 4, \delta_j - 3, \delta_j - 2, \delta_j - 1\}\}$ ;  
    **foreach**  $i \in I$  **do**  
         $x^i \leftarrow \text{insert}(\delta_j \bmod 4)$  at position  $i$  in  $x'$ ;  
        **if**  $\exists i$ , such that  $\tilde{x}^i_I = \tilde{x}_I$  **then**  
             $x' \leftarrow x^i$ ;  
            Remove  $\delta_j$  from  $\Delta$ ;  
             $j = j + 1$ ;  
        **end**  
    **end**  
**end**  
 $x \leftarrow x'$ ;  
**return**  $x$ ;

---

With Lemma 3 and the above algorithm, the problem of correcting  $t$  synthesis defects with the knowledge of defective cycles reduces to recovering the signature with the information of defective cycles.

By the definition of signature, we have the following observation.

**Observation 1** ([13]). For  $x \in \Sigma^n$ , if a deletion occurs at  $x_i$ , then  $\tilde{x}_i$  or  $\tilde{x}_{i-1}$  will be deleted. Conversely, if the index of deletion in  $\tilde{x}$  is  $i$ , then the index of deletion in  $x$  is  $i$  or  $i + 1$ .

Specifically, if  $x_{i-1} \leq x_i \leq x_{i+1}$  or  $x_{i-1} > x_i > x_{i+1}$ , then the deletion of  $x_i$  will cause a deletion of  $\tilde{x}_i$ . If  $x_{i-1} \leq x_i > x_{i+1}$  and  $x_{i-1} > x_{i+1}$ , then the deletion of  $x_i$  will cause a deletion of  $\tilde{x}_{i-1}$ . If  $x_{i-1} \leq x_i > x_{i+1}$  and  $x_{i-1} \leq x_{i+1}$ , then the deletion of  $x_i$  will cause a deletion of  $\tilde{x}_i$ . The case of  $x_{i-1} > x_i \leq x_{i+1}$  is similar. So we have the following corollary of Lemma 2.

**Corollary 1.** Let  $x \in \Sigma^n$ , and  $\Delta \subseteq \text{cycle}(x)$  is of size  $t$ . For any  $y \in B^\Delta(x)$  with  $\tilde{x}_{n \setminus \{i_1, i_2, \dots, i_t\}} = \tilde{y}_{n \setminus \{j_1, j_2, \dots, j_t\}}$ , we have  $|i_k - j_k| \leq 4k$  for  $1 \leq k \leq t$ .

With this corollary, if we want to construct a  $t$ -KDCC, it suffices to construct a binary  $P$ -bounded  $t$ -deletion correcting code for  $P = (5, 9, \dots, 4t + 1)$ .

#### B. Constructions for $t = 1$

We first construct a single-KDCC by using binary  $P$ -bounded 1-deletion correcting code with redundancy  $\log 12$ . Then, we show that this can be improved by giving another construction with redundancy  $\log 4$ .

**Definition 8.** For  $x \in \mathbb{Z}^n$ , the *VT-syndrome* of  $x$  is defined as  $\text{VT}(x) \triangleq \sum_{i=1}^n ix_i$ . Define  $\text{Sum}(x) \triangleq \sum_{i=1}^n x_i$ .



**Construction 1** ([12, Construction 1]). For  $a \in \mathbb{Z}_P, b \in \mathbb{Z}_2$ , let the shifted Varshamov-Tenengolts code be:

$$\text{SVT}_{a,b}(n, P) = \left\{ \mathbf{x} \in \Sigma^n : \text{VT}(\mathbf{x}) = a \bmod P, \right. \\ \left. \text{Sum}(\mathbf{x}) = b \bmod 2 \right\}$$

**Lemma 4** ([12, Lemma 4, 5]). For  $a \in \mathbb{Z}_P, b \in \mathbb{Z}_2$ , the shifted VT code  $\text{SVT}_{a,b}(n, P)$  is a  $P$ -bounded single-deletion correcting code, and there exist  $a$  and  $b$  such that the redundancy is at most  $\log P + 1$ .

**Construction 2.** For  $a \in \mathbb{Z}_5, b \in \mathbb{Z}_2$  define the code

$$\mathcal{C}_1^{KDCC}(n; a, b) = \left\{ \mathbf{x} \in \Sigma^n : \tilde{\mathbf{x}} \in \text{SVT}_{a,b}(n, 5) \right\}.$$

**Theorem 1.** For  $a \in \mathbb{Z}_5, b \in \mathbb{Z}_2$  the code  $\mathcal{C}_1^{KDCC}(n; a, b)$  is a 1-KDCC, and there exist  $a$  and  $b$  such that the redundancy is at most  $\log 10$ .

*Proof:* Suppose  $\mathcal{C}_1^{KDCC}(n; a, b)$  is not a 1-KDCC, then there exists a  $\mathbf{y} \in \mathcal{C}_1^{KDCC}(n; a, b)$ , such that  $\mathbf{y} \in B^\delta(\mathbf{x})$  for some  $\delta \in \text{cycle}(\mathbf{x})$ . Let  $i$  and  $j$  be the indices such that  $\mathbf{x}_{[n] \setminus \{i\}} = \mathbf{y}_{[n] \setminus \{j\}}$ , then we have  $i' \in \{i-1, i\}, j' \in \{j-1, j\}$  where  $\tilde{\mathbf{x}}_{[n] \setminus \{i'\}} = \tilde{\mathbf{y}}_{[n] \setminus \{j'\}}$ . By Lemma 2 and Corollary 1, it has to be  $|i-j| \leq 3$  and  $|i'-j'| \leq 4$  which contradict that  $\tilde{\mathbf{x}}, \tilde{\mathbf{y}} \in \text{SVT}_{a,b}(n, 5)$ .

Set  $P = 5$ , we get that there exist  $a$  and  $b$  such that the redundancy of  $\mathcal{C}_1^{KDCC}(n; a, b)$  is at most  $\log 5 + 1 = \log 10$  by Lemma 4. ■

The above shows an example of constructing 1-KDCC from binary  $P$ -bounded single-deletion correcting code. In the remaining of this subsection, we construct 1-KDCC without using the signature.

**Construction 3.** For  $a \in \mathbb{Z}_4$ , define the code

$$\mathcal{C}_1^{KDCC}(n; a) = \left\{ \mathbf{x} \in \Sigma^n : \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_{2i} = a \bmod 4 \right\}.$$

**Theorem 2.** For  $a \in \mathbb{Z}_4$  the code  $\mathcal{C}_1^{KDCC}(n; a)$  is a 1-KDCC and there exists an  $a$  such that the redundancy is at most  $\log 4$  for some  $a \in \mathbb{Z}_4$ .

*Proof:* Suppose  $\mathcal{C}_1^{KDCC}(n; a)$  is not a 1-KDCC, then there exists a  $\mathbf{y} \in \mathcal{C}_1^{KDCC}(n; a)$ , such that  $\mathbf{y} \in B^\delta(\mathbf{x})$  for some  $\delta \in \text{cycle}(\mathbf{x})$ . Again we let  $i < j$  be the indices such that  $\mathbf{x}_{[n] \setminus \{i\}} = \mathbf{y}_{[n] \setminus \{j\}}$ . By the proof of Lemma 1, if we concentrate on the different parts of  $\mathbf{x}$  and  $\mathbf{y}$  which are  $\mathbf{x}_{[i,j]}$  and  $\mathbf{y}_{[i,j]}$ , then  $x_i = y_j$  and  $x_k = y_{k-1}$  for  $i+1 \leq k \leq j$ . Besides,  $x_i, \dots, x_j$  are all different since they are synthesized within 4 cycles. Now we discuss in three cases,  $j-i = 3, 2, 1$ .

- If  $j-i = 1$ , then we have

$$\left| \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_{2i} - \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_{2i} \right| = |x_i - x_j| \in \{1, 2, 3\},$$

a contradiction.

- If  $j-i = 2$ , then we have

$$\left| \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_{2i} - \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_{2i} \right| = \begin{cases} |x_i + x_j - x_{i+1} - x_i|, & \text{if } i \text{ is even,} \\ |x_{i+1} - x_j|, & \text{if } i \text{ is odd.} \end{cases}$$

Either  $i$  is even or odd, we can get  $\left| \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_{2i} - \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_{2i} \right| = |x_{i+1} - x_j| \in \{1, 2, 3\}$ , a contradiction.

- If  $j-i = 3$ , then

$$\left| \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_{2i} - \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_{2i} \right| = |x_i + x_{i+2} - x_{i+1} - x_j|.$$

In this case,  $\mathbf{x}_{[i:j]} = x_i x_{i+1} x_{i+2} x_{i+3} \in \{1234, 2341, 3412, 4123\}$ , so  $|x_i + x_{i+2} - x_{i+1} - x_j| = 2$ , again a contradiction.

In all cases,  $1 \leq \left| \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_{2i} - \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_{2i} \right| \leq 3$ , which contradicts  $\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_{2i} = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_{2i} \bmod 4$ .

By the pigeonhole principle, there exists  $a \in \mathbb{Z}_4$ , such the code  $\mathcal{C}_1^{KDCC}(n; a)$  has size at least  $\frac{4^n}{4}$ . So the redundancy is at most  $\log 4$ . ■

### C. Construction of $t = 2$

In this subsection, we provide a 2-KDCC by constructing a binary  $P$ -bounded 2-deletion correcting code for  $P = (P_1, P_2)$ .

For  $\mathbf{x} \in \{0, 1\}^n$ , we use  $A(\mathbf{x})$  to represent the  $P \times \frac{n}{P}$  array form of  $\mathbf{x}$ . That is,

$$A(\mathbf{x}) = \begin{bmatrix} x_1 & x_{P+1} & \cdots & x_{n-P+1} \\ x_2 & x_{P+2} & \cdots & x_{n-P+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_P & x_{2P} & \cdots & x_n \end{bmatrix}.$$

Let  $A(\mathbf{x})_i$  be the  $i$ th row of  $A(\mathbf{x})$ , where  $1 \leq i \leq P$ .

**Construction 4.** For  $\mathbf{a} \in \mathbb{Z}_3^P$  and  $b \in \mathbb{Z}_{3^P n}$ , we denote

$$\mathcal{C}_2^P(n; \mathbf{a}, b) = \left\{ \mathbf{x} \in \{0, 1\}^n : \sum_{i=1}^P 3^{i-1} \text{VT}(A(\mathbf{x})_i) = b \bmod 3^P n, \right.$$

$$\left. \text{Sum}(A(\mathbf{x})_i) = a_i \bmod 3, \text{ for } 1 \leq i \leq P \right\}.$$

**Lemma 5.** For  $\mathbf{a} \in \mathbb{Z}_3^P$  and  $b \in \mathbb{Z}_{3^P n}$ , the code  $\mathcal{C}_2^P(n; \mathbf{a}, b)$  can correct two bursts of erasures of length at most  $P$ .

*Proof:* It is easy to see that a burst of erasures of length  $P$  will cause an erasure in every row of  $A(\mathbf{x})$ . If the coordinates of the two bursts of erasures overlap, or in other words, the error type is a burst of erasure of length at most  $2P-1$ , then we can erase more bits such that the error is two bursts of erasures of length  $P$ . Thus, every row of  $A(\mathbf{x})$  will have two erasures.

Suppose the erroneous word is  $\mathbf{x}'$ . We can get the two missing bits in every row by calculating  $\text{Sum}(A(\mathbf{x})_i) - \text{Sum}(A(\mathbf{x}')_i)$  for  $1 \leq i \leq P$ . If the difference is 0, then we know the two missing bits are both 0. If the difference is 2, then we know the two missing bits are both 1. In these two cases, the row will be recovered correctly. If the difference is 1, then we know the two missing bits are 1 and 0, but we do not know their order. In the following, we consider this case.

Let the indices of the two erasures in  $i$ th row are  $k_i$  and  $\ell_i$ , where  $1 \leq k_i < \ell_i \leq \frac{n}{P}$  for  $1 \leq i \leq P$ . We have  $k_i \in \{k_1, k_1 - 1\}$  and  $\ell_i \in \{\ell_1, \ell_1 - 1\}$  for  $2 \leq i \leq P$ . Suppose

we have another codeword  $\mathbf{y} \in \mathcal{C}_2^P(n; \mathbf{a}, b)$ , which results in the same word after erasing the same coordinates. Now we consider the difference of the first constraint between  $\mathbf{x}$  and  $\mathbf{y}$ . Denote their difference as

$$D \triangleq \sum_{i=1}^P 3^{i-1} \text{VT}(A(\mathbf{x})_i) - \sum_{i=1}^P 3^{i-1} \text{VT}(A(\mathbf{y})_i).$$

Since  $\mathbf{x} \neq \mathbf{y}$ , there must exist some  $\{i_1, \dots, i_p\} \subseteq [P]$ , such that  $A(\mathbf{x})_{i_j} \neq A(\mathbf{y})_{i_j}$ . As  $\text{Sum}(A(\mathbf{x})_{i_j}) = \text{Sum}(A(\mathbf{y})_{i_j}) \pmod{3}$ , we have  $s_{i_j} = A(\mathbf{x})_{i_j, k_{i_j}} = 1 - A(\mathbf{y})_{i_j, k_{i_j}} = 1 - A(\mathbf{x})_{i_j, \ell_{i_j}} = A(\mathbf{y})_{i_j, \ell_{i_j}}$ , where  $s_{i_j} \in \{0, 1\}$  for  $1 \leq j \leq p$ . Therefore,

$$\begin{aligned} D &= \sum_{i=1}^P 3^{i-1} \text{VT}(A(\mathbf{x})_i) - \sum_{i=1}^P 3^{i-1} \text{VT}(A(\mathbf{y})_i) \\ &= \sum_{j=1}^p 3^{i_j-1} \text{VT}(A(\mathbf{x})_{i_j}) - \sum_{j=1}^p 3^{i_j-1} \text{VT}(A(\mathbf{y})_{i_j}) \\ &= \sum_{j=1}^p 3^{i_j-1} (\text{VT}(A(\mathbf{x})_{i_j}) - \text{VT}(A(\mathbf{y})_{i_j})) \\ &= \sum_{j=1}^p 3^{i_j-1} (1 - 2s_{i_j})(\ell_{i_j} - k_{i_j}). \end{aligned}$$

Since  $1 \leq \ell_{i_j} - k_{i_j} \leq \frac{n}{3^P}$ , we can bound the range of  $D$ :

$$|D| \leq \sum_{j=1}^p 3^{i_j-1} \frac{n}{3^P} \leq \frac{3^P - 1}{2^P} n,$$

which is less than  $3^P n$ .

Now we show  $D$  cannot be 0. As  $k_i \in \{k_1, k_1 - 1\}$  and  $\ell_i \in \{\ell_1, \ell_1 - 1\}$  for  $2 \leq i \leq P$ , we have  $\ell_{i_j} - k_{i_j} \in \{d-1, d, d+1\}$  for  $1 \leq j \leq p$ , where  $d \triangleq \ell_{i_1} - k_{i_1}$ . We claim that there exists no  $1 \leq u < v \leq p$  such that  $\{\ell_{i_u} - k_{i_u}, \ell_{i_v} - k_{i_v}\} = \{d-1, d+1\}$ . If  $\ell_{i_u} - k_{i_u} = d-1$ , this means  $\ell_{i_u} = \ell_1 - 1, k_{i_u} = k_1$ . Since  $u < v$ , by the array representation it has  $\ell_{i_v} \leq \ell_{i_u}$  and  $k_{i_v} \leq k_{i_u}$ . The possibilities of  $(\ell_{i_v}, k_{i_v})$  are  $(\ell_1 - 1, k_1)$  and  $(\ell_1 - 1, k_1 - 1)$ . Therefore  $\ell_{i_v} - k_{i_v} \in \{d-1, d\}$ . If  $\ell_{i_u} - k_{i_u} = d+1$ , this means  $\ell_{i_u} = \ell_1, k_{i_u} = k_1 - 1$ . Similarly we can get  $\ell_{i_v} - k_{i_v} \in \{d, d+1\}$ . The above claim shows that if there exist a least index  $1 \leq j \leq p$  such that  $\ell_{i_j} - k_{i_j} = d' \neq d$ , then for all  $j < j' \leq p$ ,  $\ell_{i_{j'}} - k_{i_{j'}} \in \{d', d\}$ . In other words,  $\ell_{i_j} - k_{i_j} \in \{d-1, d\}$  or  $\ell_{i_j} - k_{i_j} \in \{d, d+1\}$  for  $1 \leq j \leq p$ .

- If  $s_{i_p} = 0$ , then

$$\begin{aligned} D &= \sum_{j=1}^p 3^{i_j-1} (1 - 2s_{i_j})(\ell_{i_j} - k_{i_j}) \\ &\geq 3^{i_p-1} (\ell_{i_p} - k_{i_p}) - \sum_{j=1}^{p-1} 3^{i_j-1} (\ell_{i_j} - k_{i_j}) \\ &\geq 3^{i_p-1} (\ell_{i_p} - k_{i_p}) - \sum_{j=1}^{p-1} 3^{i_j-1} (\ell_{i_p} - k_{i_p} + 1) \\ &\geq 3^{i_p-1} (\ell_{i_p} - k_{i_p}) - \frac{3^{i_p-1} - 1}{2} (\ell_{i_p} - k_{i_p} + 1) \\ &= \frac{3^{i_p-1} + 1}{2} (\ell_{i_p} - k_{i_p}) - \frac{3^{i_p-1} - 1}{2} \end{aligned}$$

$$\begin{aligned} &\geq \frac{3^{i_p-1} + 1}{2} - \frac{3^{i_p-1} - 1}{2} \\ &= 1. \end{aligned}$$

- If  $s_{i_p} = 1$ , then

$$\begin{aligned} D &= \sum_{j=1}^p 3^{i_j-1} (1 - 2s_{i_j})(\ell_{i_j} - k_{i_j}) \\ &\leq \sum_{j=1}^{p-1} 3^{i_j-1} (\ell_{i_j} - k_{i_j}) - 3^{i_p-1} (\ell_{i_p} - k_{i_p}) \\ &\leq \sum_{j=1}^{p-1} 3^{i_j-1} (\ell_{i_p} - k_{i_p} + 1) - 3^{i_p-1} (\ell_{i_p} - k_{i_p}) \\ &\leq \frac{3^{i_p-1} - 1}{2} (\ell_{i_p} - k_{i_p} + 1) - 3^{i_p-1} (\ell_{i_p} - k_{i_p}) \\ &= -\frac{3^{i_p-1} + 1}{2} (\ell_{i_p} - k_{i_p}) + \frac{3^{i_p-1} - 1}{2} \\ &\leq -\frac{3^{i_p-1} + 1}{2} + \frac{3^{i_p-1} - 1}{2} \\ &= -1. \end{aligned}$$

In either case,  $D$  cannot be 0, which contradicts the fact that  $\mathbf{x}, \mathbf{y} \in \mathcal{C}_2^P(n; \mathbf{a}, b)$ .  $\blacksquare$

**Theorem 3.** For  $\mathbf{P} = (P_1, P_2)$ ,  $\mathbf{a} \in \mathbb{Z}_3^P$  and  $b \in \mathbb{Z}_{3^P n}$ , the code  $\mathcal{C}_2^P(n; \mathbf{a}, b)$  is a binary  $\mathbf{P}$ -bounded two-deletion correcting code, and there exist  $\mathbf{a}$  and  $b$  such that it has redundancy at most  $\log n + 2P \log 3$ .

*Proof:* If we know the knowledge of the two intervals containing deletions, then the bits outside these two intervals are error-free and can be put in their correct positions. For example, if  $\mathbf{x}$  deletes two bits in two intervals  $[i, i + P_1 - 1]$  and  $[j, j + P_2 - 1]$  respectively, where  $1 \leq i \leq j \leq n - P + 1$ . Let  $\mathbf{x}'$  denote the erroneous word of  $\mathbf{x}$ , then by setting

$$\begin{aligned} \mathbf{x}'' &= x'_1, \dots, x'_{i-1}, \underbrace{?, \dots, ?}_{P_1}, x'_{i+P_1-1}, \dots, \\ &\quad x'_{j-2}, \underbrace{?, \dots, ?}_{P_2}, x'_{j+P_2-2}, \dots, x'_{n-2}, \end{aligned}$$

we get a word resulted from  $\mathbf{x}$  by two bursts of erasures of length  $P_1, P_2$ . By Lemma 5, we can recover  $\mathbf{x}$  from  $\mathbf{x}''$ . So the code  $\mathcal{C}_2^P(n; \mathbf{a}, b)$  can always correct two deletions if we have the information of two intervals containing deletions.

By the pigeonhole principle, there exists  $\mathbf{a} \in \mathbb{Z}_3^P, b \in \mathbb{Z}_{3^P n}$ , such the code  $\mathcal{C}_2^P(n; \mathbf{a}, b)$  has size at least

$$\frac{2^n}{3^P n \cdot 3^P}.$$

So the redundancy is at most  $\log n + 2P \log 3$ .  $\blacksquare$

Now we are ready to give the construction of 2-KDCC. By Corollary 1, we can set  $\mathbf{P} = (5, 9)$ , thus  $P = \max\{5, 9\} = 9$ .

**Construction 5.** For  $\mathbf{a} \in \mathbb{Z}_3^9, b \in \mathbb{Z}_{3^9 n}$  and  $P = 9$ , we denote

$$\mathcal{C}_2^{KDCC}(n; \mathbf{a}, b) = \left\{ \mathbf{x} \in \Sigma^n : \tilde{\mathbf{x}} \in \mathcal{C}_2^P(n; \mathbf{a}, b) \right\}.$$

**Theorem 4.** For  $\mathbf{a} \in \mathbb{Z}_3^9$  and  $b \in \mathbb{Z}_{3^9 n}$ , the code  $\mathcal{C}_2^{KDCC}(n; \mathbf{a}, b)$  is a 2-KDCC, and there exists parameters such that the redundancy is at most  $\log n + 18 \log 3$ .

*Proof:* By Lemma 3,  $\mathcal{C}_2^{KDDC}(n; \mathbf{a}, b)$  is a 2-KDDC, as we can uniquely recovery each signature of the codeword which is in a  $(5, 9)$ -bounded 2-deletion correcting code. Furthermore, there exists choices of  $\mathbf{a} \in \mathbb{Z}_3^9$  and  $b \in \mathbb{Z}_{3^9 n}$ , such that the redundancy of  $\mathcal{C}_2^{KDDC}(n; \mathbf{a}, b)$  is at most  $2n - \log \frac{4^n}{3^{18n}} = \log n + 18 \log 3$ . ■

We have constructed a binary  $P$ -bounded 2-deletion correcting code in this subsection. But if we let  $P = c \log n$  for some  $c > 1$ , then the redundancy of  $\mathcal{C}_2^P(n; \mathbf{a}, b)$  will be at least  $(2 \log 3 + 1) \log n$ . To decrease the redundancy when  $P$  is large, we further construct another binary  $P$ -bounded 2-deletion correcting code with redundancy  $2 \log n + o(\log n)$  even when  $P = O(\log n)$ . Although we do not require  $P$  to be large in this section, our  $P$ -bounded 2-deletion correcting code for  $P = O(\log n)$  will be useful in our construction of 2-SDCC.

The following lemma will lead to a  $P$ -bounded 2-deletion correcting code, and the proof can be found in Appendix A.

**Lemma 6.** For any integers  $P_1, P_2 \geq 2$  and  $n \geq 3$ , there exists a function  $\mathcal{E} : \{0, 1\}^n \rightarrow \{0, 1\}^{n+r(n, P_1, P_2)}$ , computable in linear time, such that for any  $\mathbf{x} \in \{0, 1\}^n$ , if  $\mathcal{E}(\mathbf{x})$  suffers two deletions at two intervals of lengths  $P_1, P_2$  resulting in  $\mathcal{E}'(\mathbf{x})$ , then given the information of these two intervals and  $\mathcal{E}'(\mathbf{x})$  we can uniquely recover  $\mathbf{x}$ , where  $i, j \in [n+r(n, P_1, P_2)]$  are the indices of deletions, and  $r(n, P_1, P_2) = 2 \log n + 7 \log \log n + 14 \log(P_1 + P_2) + \log P + O(\log \log(P_1 + P_2))$ .

Let  $\mathcal{C}_2^P = \{\mathbf{x} \in \{0, 1\}^n : \mathbf{x} = \mathcal{E}(\mathbf{z})_{[k]} 01 \mathcal{E}(\mathbf{z})_{[k+1, n-2]}, \mathbf{z} \in \{0, 1\}^k\}$ , where  $k + r(k, P_1, P_2) = n - 2$ . We have the following theorem.

**Theorem 5.**  $\mathcal{C}_2^P$  is a binary  $P$ -bounded two-deletion correcting code of redundancy  $2 \log n + 7 \log \log n + 14 \log(P_1 + P_2) + \log P + O(\log \log(P_1 + P_2))$ .

#### IV. CONSTRUCTIONS OF SYNTHESIS-DEFECT CORRECTING CODES

In this section, we will provide  $t$ -SDCC for  $t = 1, 2$ . First we show that we can localize each defect to a window of length  $O(\log n)$  by letting the signatures of  $O(\log n)$  strands belong to binary  $t$ -deletion correcting codes and with some period constraints. Then, it suffices to employ another coding scheme which incurs less redundancy so that we can correct the remaining strands.

##### A. Defect-Locating Strands

In this subsection, we code for a set of ordered sequences. We show that using  $O(\log n)$  strands is enough to cover all the  $4n$  cycles, so we do not need to let all  $M$  strands belong to deletion-correcting code. By coding these  $O(\log n)$  strands, we can localize each defect to a window. With this, we are prepared to provide the construction of  $t$ -SDCCs for  $t = 1, 2$ .

**Definition 9.** Define a *shift* operation on  $\mathbf{x} \in \Sigma^n$  with parameter  $1 - \text{cycle}(\mathbf{x})_1 \leq a \leq 4n - \text{cycle}(\mathbf{x})_n$  as  $S_a(\mathbf{x}) = (S_a(\mathbf{x})_1, \dots, S_a(\mathbf{x})_n)$ , where  $\text{cycle}(S_a(\mathbf{x})) = (\text{cycle}(\mathbf{x})_1 + a, \dots, \text{cycle}(\mathbf{x})_n + a)$ .

In other word, the shift operation on  $\mathbf{x}$  increases each cycle of  $\mathbf{x}$  by  $a$ . Here, we suppose the sequence  $S_a(\mathbf{x})$  can be synthesized from the  $\text{cycle}(\mathbf{x})_1 + a$  cycle and each sequence has a parameter  $a$  that will be known only by the synthesis machine.

**Example 4.** Let  $\mathbf{x} = (1, 3, 2, 1, 1, 4, 3, 2, 3, 4)$ , then we know that  $\text{cycle}(\mathbf{x}) = (1, 3, 6, 9, 13, 16, 19, 22, 23, 24)$ . Let  $a = 1$ , then  $S_1(\mathbf{x}) = (2, 4, 3, 2, 2, 1, 4, 3, 4, 1)$ , and  $\text{cycle}(S_1(\mathbf{x})) = (2, 4, 7, 10, 14, 17, 20, 23, 24, 25)$ . Let  $a = 6$ , then  $S_1(\mathbf{x}) = (3, 1, 4, 3, 3, 2, 1, 4, 1, 2)$ , and  $\text{cycle}(S_1(\mathbf{x})) = (7, 9, 12, 15, 19, 22, 25, 28, 29, 30)$ .

**Lemma 7.** For  $\mathbf{x} \in \Sigma^n$ , we have  $\text{cycle}(S_t(\mathbf{x})) \cup \text{cycle}(S_{t+1}(\mathbf{x})) \cup \text{cycle}(S_{t+2}(\mathbf{x})) \cup \text{cycle}(S_{t+3}(\mathbf{x})) = \{\text{cycle}(\mathbf{x})_1 + t, \dots, \text{cycle}(\mathbf{x})_n + t + 3\}$ , for  $1 - \text{cycle}(\mathbf{x})_1 \leq t \leq 4n - \text{cycle}(\mathbf{x})_n - 3$ .

*Proof:* By the definition of shift,  $\text{cycle}(S_t(\mathbf{x})) = (\text{cycle}(\mathbf{x})_1 + t, \dots, \text{cycle}(\mathbf{x})_n + t)$ . Thus we have

$$\cup_{a=t}^{t+3} \text{cycle}(S_a(\mathbf{x}))_i = \{\text{cycle}(\mathbf{x})_i + t, \dots, \text{cycle}(\mathbf{x})_i + t + 3\},$$

for  $i \in [n]$ . As the two elements in a cycle sequence have distance at most 4, we have  $\text{cycle}(S_{t+3}(\mathbf{x}))_i = \text{cycle}(\mathbf{x})_i + t + 3 \geq \text{cycle}(\mathbf{x})_{i+1} + t - 1 = \text{cycle}(S_t(\mathbf{x}))_{i+1} - 1$ , which means  $(\cup_{a=t}^{t+3} \text{cycle}(S_a(\mathbf{x}))_i) \cup (\cup_{a=t}^{t+3} \text{cycle}(S_a(\mathbf{x}))_{i+1}) = \{\text{cycle}(\mathbf{x})_i + t, \dots, \text{cycle}(\mathbf{x})_{i+1} + t + 3\}$ . Therefore, we have

$$\bigcup_{i=1}^n \cup_{a=t}^{t+3} \text{cycle}(S_a(\mathbf{x}))_i = \{\text{cycle}(\mathbf{x})_1 + t, \dots, \text{cycle}(\mathbf{x})_n + t + 3\},$$

which complete the proof. ■

**Lemma 8.** For  $c_1, \dots, c_{M_1} \in \mathbb{C}$  and  $t \in [1, 3n + 1]$ , there always exist  $a_1, \dots, a_{M_1}$ , such that

$$[t, n + t - 1] \subseteq \bigcup_{i=1}^{M_1} \text{cycle}(S_{a_i}(c_i)),$$

where  $M_1 = \frac{1}{2 - \log 3} \log n + 1$ ,  $a_i \in [-3, t + 2] \cap [1 - \text{cycle}(c_i)_1, 4n - \text{cycle}(c_i)_n]$ , and  $1 \leq t \leq 3n + 1$ .

*Proof:* First of all, we choose  $a_1 = t - \text{cycle}(c_1)_1$ , so  $\text{cycle}(S_{a_1}(c_1)) = \{t, t + \text{cycle}(c_1)_2 - \text{cycle}(c_1)_1, \dots, t + \text{cycle}(c_1)_n - \text{cycle}(c_1)_1\}$ . Denote  $T_1 = [t, n + t - 1] \setminus \text{cycle}(S_{a_1}(c_1))$ . Since every two consecutive elements in  $\text{cycle}(S_{a_1}(c_1))$  have difference at most 4, we have that

$$|T_1| \leq n - \frac{1}{4}n = \frac{3}{4}n.$$

Then, we choose  $a_2$ . By Lemma 7,  $\text{cycle}(S_{t - \text{cycle}(c_2)_1}(c_2)) \cup \text{cycle}(S_{t - \text{cycle}(c_2)_1 + 1}(c_2)) \cup \text{cycle}(S_{t - \text{cycle}(c_2)_1 + 2}(c_2)) \cup \text{cycle}(S_{t - \text{cycle}(c_2)_1 + 3}(c_2)) = \{t, \dots, \text{cycle}(c_2)_n - \text{cycle}(c_2)_1 + t + 3\} \supset [t, n + t - 1]$ . We claim that for  $a_2 \in \{t - \text{cycle}(c_2)_1, t - \text{cycle}(c_2)_1 + 1, t - \text{cycle}(c_2)_1 + 2, t - \text{cycle}(c_2)_1 + 3\}$ , there always exists a choice, such that  $|T_1 \cap \text{cycle}(S_{a_2}(c_2))| \geq \frac{1}{4}|T_1|$ . Otherwise,  $|T_1 \cap \text{cycle}(S_{a_2}(c_2))| < \frac{1}{4}|T_1|$  for any  $a_2 \in \{t - \text{cycle}(c_2)_1, t - \text{cycle}(c_2)_1 + 1, t - \text{cycle}(c_2)_1 + 2, t - \text{cycle}(c_2)_1 + 3\}$ . Then, we have  $|T_1| = |T_1 \cap [t, n + t - 1]| = |T_1 \cap \bigcup_{i=0}^3 (\text{cycle}(S_{t - \text{cycle}(c_2)_1 + i}(c_2)))| = |\bigcup_{i=0}^3 (T_1 \cap \text{cycle}(S_{t - \text{cycle}(c_2)_1 + i}(c_2)))| \leq \sum_{i=0}^3 |T_1 \cap \text{cycle}(S_{t - \text{cycle}(c_2)_1 + i}(c_2))| < |T_1|$ , which is a contradiction.

Let  $a_2$  be the one such that  $|T_1 \cap \text{cycle}(S_{a_2}(c_2))| \geq \frac{1}{4}|T_1|$ . Denote  $T_2 = [t, n+t-1] \setminus (\text{cycle}(S_{a_1}(c_1)) \cup \text{cycle}(S_{a_2}(c_1)))$ , then we have

$$|T_2| = |T_1 \setminus (T_1 \cap \text{cycle}(S_{a_2}(c_2)))| \leq \frac{3}{4}|T_1| \leq \frac{9}{16}n.$$

Repeating this, we get  $|T_{M_1-1}| \leq (\frac{3}{4})^{M_1-1}n = 1$ . Same as previous, we choose  $a_{M_1}$ , such that  $|T_{M_1-1} \cap \text{cycle}(S_{a_{M_1}}(c_{M_1}))| \geq 1$ . So when we finish choosing  $a_1, \dots, a_{M_1}$ , we can make sure  $[t, n+t-1]$  is covered by  $\text{cycle}(S_{a_i}(c_i))$  for  $i \in [M_1]$ . ■

**Corollary 2.** For  $c_1, \dots, c_{4M_1} \in \mathbb{C}$ , there always exist  $a_1, \dots, a_{4M_1} \in [-3, 3n+1]$ , such that  $[1, 4n] \subseteq \bigcup_{i=1}^{4M_1} \text{cycle}(S_{a_i}(c_i))$ , where  $M_1 = \frac{1}{2-\log 3} \log n + 1$ .

Lemma 8 and Corollary 2 show that we can use  $O(\log n)$  strands to cover all synthesis cycles. So, we can know the approximate locations of all the defects if these  $O(\log n)$  strands are corrected. To correct the quaternary strands, our solution in this paper is to correct their signatures first. Then, the locations of deletions in signatures will provide us the information of defects in quaternary strands.

**Lemma 9.** If  $\mathbf{x} \in \Sigma^n$  suffers from a synthesis defect at cycle  $\delta_1 \in [4n]$  and result in  $\mathbf{x}'$ , then for any  $\mathbf{y} \in \Sigma^n$  with  $\mathbf{y}_{[n] \setminus \{j\}} = \mathbf{x}'$  and  $\tilde{\mathbf{y}} = \tilde{\mathbf{x}}$ , we have  $|\delta_2 - \delta_1| \leq 4P + 4$ , where  $P$  is the length of the longest run in  $\tilde{\mathbf{x}}$  and  $j \in [n]$  is the index satisfying  $\text{cycle}(\mathbf{y})_j = \delta_2$ .

*Proof:* Suppose the synthesis defect causes a deletion at  $i$ th position of  $\mathbf{x}$ , then by Observation 1 the location of deletion in  $\tilde{\mathbf{x}}$  is  $i' \in \{i-1, i\}$ . Since  $\mathbf{y}_{[n] \setminus \{j\}} = \mathbf{x}_{[n] \setminus \{i\}}$ ,  $\mathbf{y}$  can be obtained by inserting  $y_j$  in  $\mathbf{x}'$ . Besides, due to  $\tilde{\mathbf{y}} = \tilde{\mathbf{x}}$ , the insertion of  $y_j$  must lead to an insertion at the same run with  $\tilde{x}_{i'}$ . So, if the insertion of  $y_j$  causes an insertion at  $j'$ th position of  $\mathbf{x}'$ , then we have  $|j' - i'| \leq P - 1$ . As  $j' \in \{j-1, j\}$ , we have  $|j - i| \leq P$ . Without loss of generality, we assume  $j > i$ . Furthermore,  $\text{cycle}(\mathbf{x})_{i+1} \in [\delta_1 + 1, \delta_1 + 4]$ , so we have  $\text{cycle}(\mathbf{y})_{j-1} = \text{cycle}(\mathbf{x}')_{j-1} \leq \text{cycle}(\mathbf{x})_j \leq \delta_1 + 4(j - i)$ , and thus  $\text{cycle}(\mathbf{y})_j \leq \text{cycle}(\mathbf{y})_{j-1} + 4 \leq \delta_1 + 4(j - i) + 4 \leq \delta_1 + 4P + 4$ . Therefore, we have  $|\delta_2 - \delta_1| \leq 4P + 4$ . ■

Notice that, even if the signature of a strand is recovered from  $t$  deletions, the locations of errors may not be determined exactly. For example, if a strand  $\mathbf{x} \in \Sigma^n$  suffers from  $t$  deletions, which causes  $t$  consecutive deletions in  $\tilde{\mathbf{x}}$ , where  $\tilde{x}_i = \tilde{x}_{i+t}$  for  $1 \leq i \leq n - t - 1$ , then we cannot know the locations of deletions since deleting any  $t$  consecutive bits from  $\tilde{\mathbf{x}}$  will result in the same one. Intuitively, if we want to locate each of the location of error in a signature uniquely, then we need to make sure  $\tilde{\mathbf{x}}$  cannot have period  $p$  for all  $1 \leq p \leq t$ , which is a strict condition. In the following, we use the regular sequence from [14, Definition 3] to handle the cases of  $t = 1, 2$ .

**Definition 10** ([14, Definition 3]). A sequence  $\mathbf{x} \in \{0, 1\}^n$  is *regular* if each consecutive substring of  $\mathbf{x}$  of length  $d \log n$  contains both 00 and 11.

**Lemma 10** ([14, Lemma 11]). Denote  $\mathcal{R}(n)$  the set of all regular binary sequences of length  $n$  with  $d = 7$ , then  $|\mathcal{R}(n)| \geq 2^{n-1}$ .

## B. 1-Synthesis-Defect Correcting Code

Next we are prepared to provide a construction of 1-SDCC, which shows our general strategy to construct synthesis-defect correcting code. Before that, let us introduce the well-known *Varshamov-Tenengolts (VT) codes*.

**Lemma 11** ([15]). The Varshamov-Tenengolts (VT) code  $\text{VT}(n; a) = \{\mathbf{x} \in \{0, 1\}^n : \text{VT}(\mathbf{x}) = a \bmod (n+1)\}$  is a single-deletion correcting code, and there exists  $a \in \mathbb{Z}_{n+1}$  such that the redundancy of  $\text{VT}(n; a)$  is at most  $\log(n+1)$ .

**Construction 6.** For  $\mathbf{s} \in \mathbb{Z}_4^{4M_1}$ ,  $\mathbf{b} \in \mathbb{Z}_{n+1}^{4M_1}$ ,  $\mathbf{d} \in \mathbb{Z}_{P+1}^{M-4M_1}$  and  $\mathbf{e} \in \mathbb{Z}_2^{M-4M_1}$ , where  $P = 28 \log n + 5$  and  $M_1 = \frac{1}{2-\log 3} \log n + 1$ , we denote

$$\mathcal{C}_1^{SD} = \left\{ \mathbf{C} \in (\Sigma^n)^M : \begin{aligned} & \mathbf{c}_i = S_{a_i}(\mathbf{x}_i), \text{Sum}(\mathbf{x}_i) = s_i \bmod 4, \\ & \text{and } \tilde{\mathbf{x}}_i \in \text{VT}(n; b_i) \cap \mathcal{R}(n), \text{ for } i \in [4M_1], \\ & \tilde{\mathbf{c}}_i \in \text{SVT}_{d_i-4M_1, e_i-4M_1}(n, P+1) \text{ for } i \in [4M_1+1, M] \end{aligned} \right\},$$

where  $a_i$  is chosen based on Corollary 2 for  $1 \leq i \leq 4M_1$ .

**Theorem 6.** For  $\mathbf{s} \in \mathbb{Z}_4^{4M_1}$ ,  $\mathbf{b} \in \mathbb{Z}_{n+1}^{4M_1}$ ,  $\mathbf{d} \in \mathbb{Z}_{P+1}^{M-4M_1}$  and  $\mathbf{e} \in \mathbb{Z}_2^{M-4M_1}$ , where  $P = 28 \log n + 5$ , the code  $\mathcal{C}_1^{SD}$  is a 1-SDCC, and there exist choices of parameters, such that the redundancy of  $\mathcal{C}_1^{SD}$  is at most  $\frac{4}{2-\log 3}(\log n)^2 + M(\log \log n + 6) - \Omega(\log n \log \log n)$ .

*Proof:* By Corollary 2, we have  $\bigcup_{i=1}^{4M_1} \text{cycle}(c_i) = [4n]$ . So if there is a defective cycle, then there must be a  $c_i$  suffering from a deletion for  $i \in [4M_1]$ . As  $\tilde{\mathbf{x}}_i$  belongs to a VT code, we can recover  $\tilde{\mathbf{x}}_i$  from a deletion uniquely. Besides, we know  $\text{Sum}(\mathbf{x}_i) \bmod 4$  and  $\text{Sum}(\mathbf{x}'_i) \bmod 4$  for the erroneous sequence  $\mathbf{x}'$ , so we can calculate the value of deleted symbol. With the symbol and the signature, we can recover  $\mathbf{x}$  in the same way as non-binary VT code [13].

Due to  $\tilde{\mathbf{x}}_i \in \mathcal{R}(n)$ , the length of a run in  $\tilde{\mathbf{x}}_i$  is at most  $7 \log n$ . By Lemma 9, we know the defective cycle is within an interval of length  $P = 28 \log n + 5$ .

Now we show how to recover  $c_i$  for  $i \in [4M_1+1, M]$ . In the previous step, we have already located the defective cycle at a length- $P$  interval, so the deletions in the remaining  $\tilde{\mathbf{c}}_i$ s are within an interval of length at most  $P+1$ . Since  $\tilde{\mathbf{c}}_i$  is in a shifted VT code  $\text{SVT}_{d_i-4M_1, e_i-4M_1}(n, P+1)$ , we can recover it. As we have already known the value of defective cycle by recovering  $c_i$  for  $i \in [4M_1]$ , we can recover  $c_i$  from  $\tilde{\mathbf{c}}_i$  for  $i \in [4M_1+1, M]$ .

Now we calculate the redundancy of  $\mathcal{C}_1^{SD}$ . There are  $4M_1$  regular sequences belonging to VT code, and the sum of symbols in each sequence is equal to some value modular 4. So there exists  $\mathbf{s} \in \mathbb{Z}_4^{4M_1}$ ,  $\mathbf{b} \in \mathbb{Z}_{n+1}^{4M_1}$ , such the redundancy for this part is at most  $4M_1 \log(n+1) + 12M_1$ . For the remaining sequences, the redundancy for their part is at most  $(M - 4M_1) \log 2(P+1)$ . So the total redundancy is at most

$$\frac{4}{2-\log 3}(\log n)^2 + M(\log \log n + 6) - \Omega(\log n \log \log n).$$

The above 1-SDCC  $\mathcal{C}_1^{SD}$  shows a high-level idea to construct  $t$ -SDCC: use  $O(\log n)$  strands cover all cycles and let



them belong to a  $t$ -deletion correcting code with some period constraints, then let the remaining sequences belong to some bounded deletion-correcting codes.

### C. 2-Synthesis-Defect Correcting Code

In the case of  $t = 1$ , each strand either suffers from 1 deletion or keeps unchanged. However, if  $t \geq 2$ , each strand will suffer  $t'$  deletions for  $t' \in [0, t]$ . Even when we locate these  $t$  deletions to  $t$  interval, we do not know each of the remaining  $M - 4M_1$  strands suffers from which of these  $t$  deletions. So we cannot locate the approximate locations of  $t'$  deletions for the remaining  $M - 4M_1$  strands. Nevertheless, we can solve the case for  $t = 2$ , which is more complicated than the case  $t = 1$ .

We first present a quaternary two-deletion correcting code, which is constructed by using the signature.

**Lemma 12** ([14, Theorem 7 and 8]). There is a binary code  $\mathcal{C}_2 \subseteq \{0, 1\}^n$  with each codeword is regular and capable of correcting two deletions. The redundancy of  $\mathcal{C}_2$  is at most  $4 \log n + 10 \log \log n + O(1)$ .

**Definition 11.** Define the run sequence of  $\mathbf{x} \in \{0, 1\}^n$  as  $R(\mathbf{x})$ , where  $R(\mathbf{x})_1 = 1$ , and for  $1 \leq i \leq n - 1$ ,

$$R(\mathbf{x})_{i+1} = \begin{cases} R(\mathbf{x})_i + 1, & \text{if } x_{i+1} \neq x_i, \\ R(\mathbf{x})_i, & \text{if } x_{i+1} = x_i. \end{cases}$$

Furthermore, let  $r(\mathbf{x}) = R(\mathbf{x})_n$  be the number of run in  $\mathbf{x}$ .

For example, if  $\mathbf{x} = 10010111$ , then  $R(\mathbf{x}) = 12234555$ . The following lemma comes from [16], but complements their flaw.

**Lemma 13** ([16, Lemma III.1]). For any  $\mathbf{x} \in \Sigma^n$ , given the values  $\sum_{i=1}^{r(\tilde{\mathbf{x}})} x_i \cdot R(\tilde{\mathbf{x}})_i \pmod{4n}$  and  $\tilde{\mathbf{x}}$ , if  $\mathbf{x}$  suffers from two deletions resulting in  $\mathbf{x}'$ , and the corresponding two deleted bits in  $\tilde{\mathbf{x}}$  are not consecutive and in a period-2 alternate substring of length greater than 2, then we can uniquely recovery  $\mathbf{x}$  with the two values of deleted symbol.

With this lemma, it suffices to consider the case where two deletions in  $\tilde{\mathbf{x}}$  are consecutive and in a period-2 alternate substring. In this case, deleting any two consecutive bits in a period-2 alternate substring will result in the same string, so we cannot obtain the runs where the two deletions are deleted from.

**Lemma 14.** For  $\mathbf{x} \in \Sigma^n$ , if two deletions of  $\mathbf{x}$  cause two consecutive deletions in an alternate substring of  $\tilde{\mathbf{x}}$ , then the two deletions in  $\mathbf{x}$  are consecutive or separated by one position.

*Proof:* By Observation 1, a deletion indexed at  $i$  in  $\tilde{\mathbf{x}}$  implies the deletion in  $\mathbf{x}$  is indexed at  $i$  or  $i + 1$ . Suppose two consecutive deletions in  $\tilde{\mathbf{x}}$  caused by the two deletions in  $\mathbf{x}$  have indices  $i$  and  $i + 1$  for  $1 \leq i \leq n - 1$ , then it follows that the two deletions in  $\mathbf{x}$  have indices  $i, i + 1$  or  $i, i + 2$  or  $i + 1, i + 2$ , which completes the proof. ■

**Definition 12.** For  $\mathbf{x} \in \Sigma^n$  and  $\sigma \in \Sigma$ , define  $N_\sigma(\mathbf{x})$  the number of  $\sigma$  in  $\mathbf{x}$ . Furthermore, let  $\mathbf{x}^\sigma \in [n]^{N_\sigma(\mathbf{x})}$  denote the sequence consists of the indices of  $\sigma$ .

For example, if  $\mathbf{x} = 122124123$ , then  $N_1(\mathbf{x}) = 3, N_2(\mathbf{x}) = 4, N_3(\mathbf{x}) = 1, N_4(\mathbf{x}) = 1$  and  $\mathbf{x}^1 = 147, \mathbf{x}^2 = 2358, \mathbf{x}^3 = 9, \mathbf{x}^4 = 6$ . Now we present a lemma which shows that the quaternary sequence is unique if it consists of two symbols of given numbers, and has an alternate signature.

**Lemma 15.** If  $\mathbf{x}, \mathbf{y} \in \Sigma^n$  satisfying that  $N_{\sigma_1}(\mathbf{x}) = N_{\sigma_1}(\mathbf{y}) > 0, N_{\sigma_2}(\mathbf{x}) = N_{\sigma_2}(\mathbf{y}) > 0, N_{\sigma_3}(\mathbf{x}) = N_{\sigma_3}(\mathbf{y}) = N_{\sigma_4}(\mathbf{x}) = N_{\sigma_4}(\mathbf{y}) = 0$  for  $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ , and  $\tilde{\mathbf{x}} = \tilde{\mathbf{y}} = 0101\dots$  or  $1010\dots$ , then we have  $\mathbf{x} = \mathbf{y}$ .

*Proof:* Without loss of generality, we suppose  $\sigma_1 < \sigma_2$ . We prove the case  $\tilde{\mathbf{x}} = \tilde{\mathbf{y}} = 0101\dots$ , the other can be proved similarly. For every  $\tilde{x}_i \tilde{x}_{i+1} = 01$ , it must have  $x_i x_{i+1} = \sigma_2 \sigma_1$  due to  $\tilde{x}_i = 0$ . Thus, if  $n$  is even, then we have  $\mathbf{x} = \mathbf{y} = (\sigma_2 \sigma_1)^{n/2}$ . If  $n$  is odd, we can determine all symbols except the last one. As  $\tilde{x}_{n-1} = 1$  and  $x_{n-1} = \sigma_1$ , the last symbol  $x_n$  could be either  $\sigma_1$  or  $\sigma_2$ . For the reason that  $N_{\sigma_1}(\mathbf{x}) = N_{\sigma_1}(\mathbf{y})$  and  $N_{\sigma_2}(\mathbf{x}) = N_{\sigma_2}(\mathbf{y})$ , we must have  $\mathbf{x} = \mathbf{y}$ , which completes the proof. ■

**Construction 7.** For  $\mathbf{a} \in \mathbb{Z}_3^4, \mathbf{b} \in \mathbb{Z}_{14 \log n}^4$  and  $c \in \mathbb{Z}_{nq}$ , denote the code

$$\mathcal{C}_2^D = \left\{ \mathbf{x} \in \Sigma^n : \tilde{\mathbf{x}} \in \mathcal{C}_2, \sum_{i=1}^{r(\tilde{\mathbf{x}})} x_i \cdot R(\tilde{\mathbf{x}})_i = c \pmod{4n}, \text{ and} \right. \\ \left. N_\sigma(\mathbf{x}) = a_\sigma \pmod{3}, \text{Sum}(\mathbf{x}^\sigma) = b_\sigma \pmod{14 \log n} \text{ for } \sigma \in \Sigma \right\}.$$

**Theorem 7.** For  $\mathbf{a} \in \mathbb{Z}_3^4, \mathbf{b} \in \mathbb{Z}_{14 \log n}^4$  and  $c \in \mathbb{Z}_{nq}$ , the code  $\mathcal{C}_2^D$  is a quaternary two-deletion correcting code, and there are parameters such the redundancy of  $\mathcal{C}_2^D$  is at most  $5 \log n + 14 \log \log n + O(1)$ .

*Proof:* For  $\mathbf{x} \in \mathcal{C}_2^D$ , denote the erroneous sequence of  $\mathbf{x}$  as  $\mathbf{x}'$ . Since the signature  $\tilde{\mathbf{x}}$  belongs to a binary two-deletion correcting code  $\mathcal{C}_2$ , so we can recovery  $\tilde{\mathbf{x}}$  from  $\tilde{\mathbf{x}}'$ . If the two deletions in  $\tilde{\mathbf{x}}$  are not consecutive or are consecutive 00 or 11, then by Lemma 13 we can recovery  $\mathbf{x}$  uniquely.

On the other hand, if the two deletions in  $\tilde{\mathbf{x}}$  are consecutive 01 or 10 in an alternate substring of length greater than 2, then we can make sure the locations of these two deletions are in an interval of length  $7 \log n$  for the reason that  $\tilde{\mathbf{x}}$  is regular. Therefore, the two deletions in  $\mathbf{x}$  is within a window of length at most  $7 \log n + 1$ .

For  $\sigma \in \Sigma$ , by calculating  $N_\sigma(\mathbf{x}) - N_\sigma(\mathbf{x}') \pmod{3}$ , the values of two deleted symbols can be obtained. If there is a  $\sigma \in \Sigma$ , such  $N_\sigma(\mathbf{x}) - N_\sigma(\mathbf{x}') \pmod{3} = t$ , then the number of  $\sigma$  deleted from  $\mathbf{x}$  is  $t$ . We consider the two cases whether the two deleted symbols are the same.

- The two deleted symbols are  $\sigma_1 \neq \sigma_2$ :

We prove that inserting  $\sigma_1$  and  $\sigma_2$  into  $\mathbf{x}'$  can only get  $\mathbf{x}$  if we want to keep  $\text{Sum}(\mathbf{x}^\sigma) = b_\sigma \pmod{14 \log n}$  for all  $\sigma \in \Sigma$ . Otherwise, suppose there is another way to insert  $\sigma_1$  and  $\sigma_2$  satisfying the constraints, and the resultant sequence is denoted as  $\mathbf{y}$ . Let  $i_1$  and  $i_2$  denote the indices of the deleted  $\sigma_1$  and  $\sigma_2$  in  $\mathbf{x}$ , and  $\{j_1, j_2\}$  denote the set of indices of the inserted  $\sigma_1$  and  $\sigma_2$  in  $\mathbf{y}$ . Without loss of generality, we assume  $i_1 \leq j_1, i_1 < i_2$ , and  $j_1 < j_2$ . By Lemma 14, we have  $i_2 = i_1 + \epsilon_1$  and  $j_2 = j_1 + \epsilon_2$ ,

where  $\epsilon_1, \epsilon_2 \in \{1, 2\}$ . We claim that  $j_2 > i_2$ . Otherwise if  $j_2 \leq i_2$ , then to make sure  $\mathbf{x} \neq \mathbf{y}$  we must have  $\epsilon_1 = 2$ , and  $j_2 = i_2 = j_1 + 1$  or  $j_1 = i_1 = j_2 - 1$ . In the case of  $j_2 = i_2 = j_1 + 1$ , we have  $\mathbf{x}_{[i_1, i_2]} = \sigma_1 x_{i+1} \sigma_2$ , and  $\mathbf{y}_{[i_1, i_2]} = x_{i+1} \sigma_1 \sigma_2$  or  $x_{i+1} \sigma_2 \sigma_1$ . If  $x_{i+1} \leq \sigma_2$ , then  $\sigma_1 > x_{i+1}$  by the condition that  $\tilde{\mathbf{x}}_{[i_1, i_2]}$  is alternate. So  $\mathbf{y}_{[i_1, i_2]}$  cannot be  $x_{i+1} \sigma_1 \sigma_2$  or  $x_{i+1} \sigma_2 \sigma_1$ , where both are contradictory to  $\tilde{\mathbf{x}}_{[i_1, i_2]} = \tilde{\mathbf{y}}_{[i_1, i_2]}$ . If  $x_{i+1} > \sigma_2$ , then  $\sigma_1 \leq x_{i+1}$ , and due to  $\tilde{\mathbf{y}}_{i_1} = \tilde{\mathbf{x}}_{i_1} = 1$  the only possible of  $\mathbf{y}_{[i_1, i_2]}$  will be  $x_{i+1} \sigma_1 \sigma_2$ , where  $x_{i+1} = \sigma_1$ . This leads to  $\mathbf{x} = \mathbf{y}$ , again a contradiction. The case  $j_1 = i_1 = j_2 - 1$  can be proved similarly, so we have  $j_2 > i_2$ . If  $\mathbf{x}_{[i_1, j_2]}$  consists of  $\sigma_1$  and  $\sigma_2$  only, then to keep  $\tilde{\mathbf{x}}_{[i_1, j_2-1]}$  is an alternate sequence it has to be  $\mathbf{x} = \mathbf{y}$  by Lemma 15. Thus, there must exist a  $\sigma_3 \in \Sigma \setminus \{\sigma_1, \sigma_2\}$  in  $\mathbf{x}_{[i_1, j_2]}$ . Due to  $\mathbf{x}_{[i_1, j_2] \setminus \{i_1, i_2\}} = \mathbf{y}_{[i_1, j_2] \setminus \{j_1, j_2\}}$  and  $i_1 < i_2 < j_2$ , we have

$$N_{\sigma_3}(\mathbf{x}_{[i_1, j_2]}) \leq \text{Sum}(\mathbf{x}^{\sigma_3}) - \text{Sum}(\mathbf{y}^{\sigma_3}) \leq 2N_{\sigma_3}(\mathbf{x}_{[i_1, j_2]}),$$

which is a contradiction to  $\text{Sum}(\mathbf{x}^{\sigma_3}) - \text{Sum}(\mathbf{y}^{\sigma_3}) = 0 \pmod{14 \log n}$ .

- The two deleted symbols are both equal to  $\sigma_1$ :

Like the previous case, we prove that inserting these two  $\sigma_1$  into  $\mathbf{x}'$  can only get  $\mathbf{x}$  if we want to keep  $\text{Sum}(\mathbf{x}^\sigma) = b_\sigma \pmod{14 \log n}$  for all  $\sigma \in \Sigma$ . By Lemma 14, the two  $\sigma_1$  inserted in  $\mathbf{x}'$  are consecutive or separated by one position. Suppose there is another way to insert two  $\sigma_1$  into  $\mathbf{x}'$ , and the resultant sequence is  $\mathbf{y} \in \mathcal{C}_2^D$ . We again assume the indices of these two  $\delta_1$  in  $\mathbf{x}$  and  $\mathbf{y}$  are  $i_1 < i_2$  and  $j_1 < j_2$ , respectively. Without loss of generality, let  $i_1 < j_1$ . Since  $\mathbf{x} \neq \mathbf{y}$ , there must be a  $\sigma_2 \in \Sigma \setminus \{\delta_1\}$  in  $\mathbf{x}_{[i_1, j_2]}$ . We can calculate that

$$N_{\sigma_2}(\mathbf{x}_{[i_1, j_2]}) \leq \text{Sum}(\mathbf{x}^{\sigma_2}) - \text{Sum}(\mathbf{y}^{\sigma_2}) \leq 2N_{\sigma_2}(\mathbf{x}_{[i_1, j_2]}),$$

which contradicts that  $\text{Sum}(\mathbf{x}^{\sigma_2}) \equiv \text{Sum}(\mathbf{y}^{\sigma_2}) \pmod{14 \log n}$ .

Now we have proved that we can uniquely recover  $\mathbf{x}$  from two deletions if the two deletions in  $\tilde{\mathbf{x}}$  are consecutive 01 or 10 in an alternate substring. Combining with Lemma 13, the code  $\mathcal{C}_2^D$  is indeed a quaternary two-deletion correcting code.

For the redundancy, there exist parameters, such that the size of  $\mathcal{C}_2^D$  is at least

$$\frac{4^n}{c \cdot n^4 \cdot (\log n)^{10} \cdot 4n \cdot 3^4 \cdot (14 \log n)^4},$$

where  $c$  is a constant and from  $\mathcal{C}_2$ . Therefore, the redundancy is at most  $5 \log n + 14 \log \log n + O(1)$ . ■

**Construction 8.** For  $c \in \mathbb{Z}_{14 \log n}^{(M-4M_1) \times 4}$ ,  $\mathbf{P} = (P_1, P_2)$ , where  $P_1, P_2 \leq 28 \log n + 5$ , we denote

$$\mathcal{C}_2^{SD} = \left\{ \mathbf{C} \in (\Sigma^n)^M : \mathbf{c}_i = S_{a_i}(\mathbf{x}_i) \text{ for } i \in [4M_1] \text{ where } \mathbf{x}_i \in \mathcal{C}_2^D, \right. \\ \left. \tilde{\mathbf{c}}_i \in \mathcal{C}_2^P, \text{ and } \text{Sum}(\tilde{\mathbf{c}}_i') = b_{i,j} \pmod{14 \log n} \right. \\ \left. \text{for } i \in [4M_1 + 1, M], j \in \Sigma \right\},$$

where for  $1 \leq i \leq 4M_1$ ,  $a_i$  is chosen based on Corollary 2.

**Lemma 16.**  $\mathcal{C}_2^P$  is not only a binary  $\mathbf{P}$ -bounded two-deletion correcting code, but also a single-deletion correcting code.

*Proof:* From the construction of  $\mathcal{C}_2^P$ , every codeword in  $\mathcal{C}_2^P$  has the form  $(\mathbf{z}, 0, 1, \mathcal{E}_1(\mathbf{z}), \mathcal{E}_2(\mathbf{z}), \xi(\mathcal{E}_1(\mathbf{z}), \mathcal{E}_2(\mathbf{z})))$ , where  $\mathbf{z} \in \{0, 1\}^k$  and  $\mathcal{E}_1, \mathcal{E}_2, \xi$  are labeling function we will describe in Appendix A. As we will show,  $\xi(\mathbf{z})$  allows us recover  $\mathbf{z}$  from two deletions from the erroneous sequence, and  $\mathcal{E}_2$  contains the value  $\text{VT}(\mathbf{z}) \pmod{k+1}$ .

When a deletion occurs in a sequence with the form  $(\mathbf{z}, 0, 1, \mathcal{E}_1(\mathbf{z}), \mathcal{E}_2(\mathbf{z}), \xi(\mathcal{E}_1(\mathbf{z}), \mathcal{E}_2(\mathbf{z})))$  and result in  $\mathbf{x} \in \{0, 1\}^{n-1}$ , we first look at the  $k+1$  position of  $\mathbf{x}$ . If  $x_{k+1}=0$ , then there is no error in  $\mathbf{z}$ , so we can calculate the values  $\mathcal{E}_1(\mathbf{x}_{[k]}), \mathcal{E}_2(\mathbf{x}_{[k]}), \xi(\mathcal{E}_1(\mathbf{x}_{[k]}), \mathcal{E}_2(\mathbf{x}_{[k]}))$ , and return  $(\mathbf{x}_{[k]}, 0, 1, \mathcal{E}_1(\mathbf{x}_{[k]}), \mathcal{E}_2(\mathbf{x}_{[k]}), \xi(\mathcal{E}_1(\mathbf{x}_{[k]}), \mathcal{E}_2(\mathbf{x}_{[k]})))$  as the correct sequence. If  $x_{k+1} = 1$ , then the deletion occurs at  $\mathbf{z}$ . We use  $\mathbf{x}_{[k-1]}$  and the value  $\text{VT}(\mathbf{z}) \pmod{k+1}$  in  $\mathcal{E}_2(\mathbf{z})$  to recover  $\mathbf{z}$ , then we return  $(\mathbf{z}, \mathbf{x}_{[k,n]})$  as the correct sequence. ■

**Theorem 8.**  $\mathcal{C}_2^{SD}$  is a 2-SDCC, and there exist parameters such that the redundancy is at most  $\frac{12}{2-\log 3}(\log n)^2 + M(2 \log n + 26 \log \log n + o(\log \log n)) - \Omega(\log n \log \log n)$ .

*Proof:* The proof is similar to that of Theorem 6. ■

## V. A LOWER BOUND ON THE REDUNDANCY OF KNOWN-SYNTHESIS-DEFECT CORRECTING CODES

In this section, we give the lower bound for redundancy on 1-KDCC using some tools from graph theory. Specifically, we use the same method as [17, Section IV] to derive the lower bound for redundancy. First, let us introduce some definitions.

**Definition 13.** A collection  $\mathcal{Q}$  of cliques is a clique cover of a graph  $\mathcal{G}$  if every vertex in  $\mathcal{G}$  belongs to some clique in  $\mathcal{Q}$ .

**Lemma 17** ([18]). If  $\mathcal{Q}$  is a clique cover of  $\mathcal{G}$ , then the size of any independent set of  $\mathcal{G}$  is at most  $|\mathcal{Q}|$ .

Let the vertices of  $\mathcal{G}$  to be all quaternary sequences of length  $n$ , and two vertices  $\mathbf{u}, \mathbf{v}$  are connected if there exists a set  $\Delta = \{\delta\}$ , such that  $\text{SynDef}_\Delta(\mathbf{u}) = \text{SynDef}_\Delta(\mathbf{v})$ , where  $\delta \in [4n]$ . Obviously, an independent set of  $\mathcal{G}$  is a 1-KDCC. By Lemma 17, the upper bound of the size of 1-KDCC is upper bounded by the size of any one of clique cover of  $\mathcal{G}$ . Now we give a construction of a clique cover of  $\mathcal{G}$ .

**Lemma 18.** There exists a clique cover  $\mathcal{Q}$  for  $\mathcal{G}$  and is of size  $4^{n-1}(1 + 3(\frac{4^5-16}{4^5})^{\lfloor \frac{n}{5} \rfloor})$ .

*Proof:* Let

$$\begin{aligned} A &= \{11234, 12134, 12314, 12341\} \\ B &= \{22341, 23241, 23421, 23412\} \\ C &= \{33412, 34312, 34132, 34123\} \\ D &= \{44123, 41423, 41243, 41234\}, \end{aligned}$$

and  $E = \Sigma^5 \setminus (A \cup B \cup C \cup D)$ . Furthermore, define

$$Z = \left\{ (p, \mathbf{r}, i) : p \in E^i, \mathbf{r} \in \Sigma^{n-5i-5}, i \in \{0, \dots, \lfloor \frac{n}{5} \rfloor - 1\} \right\}.$$

For every  $z = (\mathbf{p}, \mathbf{r}, i) \in Z$ , we define

$$\begin{aligned} Q_z^{(A)} &= \{\mathbf{pqr} : \mathbf{q} \in A\}, \\ Q_z^{(B)} &= \{\mathbf{pqr} : \mathbf{q} \in B\}, \\ Q_z^{(C)} &= \{\mathbf{pqr} : \mathbf{q} \in C\}, \\ Q_z^{(D)} &= \{\mathbf{pqr} : \mathbf{q} \in D\} \end{aligned}$$

to be the cliques of size 4. For the remaining vertices, we define

$$S_x = \{\mathbf{x}\}, \text{ where } \mathbf{x} \in E^{\lfloor \frac{n}{5} \rfloor} \times \Sigma^{n-5\lfloor \frac{n}{5} \rfloor}$$

to be the singletons.

Define

$$\begin{aligned} \mathcal{Q} &= \left\{ Q_z^{(A)}, Q_z^{(B)}, Q_z^{(C)}, Q_z^{(D)}, z \in Z \right\} \cup \\ &\quad \left\{ Q_x : \mathbf{x} \in E^{\lfloor \frac{n}{5} \rfloor} \times \Sigma^{n-5\lfloor \frac{n}{5} \rfloor} \right\}. \end{aligned}$$

To prove that  $\mathcal{Q}$  is a clique cover, we need to show every set in  $\mathcal{Q}$  is a clique and all vertices in these sets cover the whole space  $\Sigma^n$ .

First, for every  $\mathbf{x} \in E^{\lfloor \frac{n}{5} \rfloor} \times \Sigma^{n-5\lfloor \frac{n}{5} \rfloor}$ , the set  $Q_x$  has only one element and thus is a clique.

Then, for  $z \in Z$ , the set  $Q_z^{(A)}$  has 4 elements, which are  $\mathbf{p11234r}, \mathbf{p12134r}, \mathbf{p12314r}, \mathbf{p12341r}$ . Suppose  $\text{cycle}(\mathbf{p11234r})_{|\mathbf{p}|+2} = \delta$ , then  $\text{SynDef}_{\{\delta\}}(\mathbf{p11234r}) = \text{SynDef}_{\{\delta\}}(\mathbf{p12134r}) = \text{SynDef}_{\{\delta\}}(\mathbf{p12314r}) = \text{SynDef}_{\{\delta\}}(\mathbf{p12341r})$ , so they are connected to each other and thus  $Q_z^{(A)}$  is a clique. Similarly,  $Q_z^{(B)}, Q_z^{(C)}, Q_z^{(D)}$  are all cliques.

For a vertex  $\mathbf{v} \in \Sigma^n$ , if there exists a  $i \in \{0, \dots, \lfloor \frac{n}{5} \rfloor - 1\}$ , such that  $v_{[5i+1, 5i+5]} \in A \cup B \cup C \cup D$ , then  $\mathbf{v} \in Q_z^{(A)}$  or  $Q_z^{(B)}$  or  $Q_z^{(C)}$  or  $Q_z^{(D)}$  for some  $z = (\mathbf{p}, \mathbf{r}, i)$ . If no such  $i$  exists, then  $\mathbf{v} \in S_v$ . Therefore, we conclude that  $\mathcal{Q}$  is a clique cover.

For the size of  $\mathcal{Q}$ , we calculate as follows. For every  $z \in Z$ , there are 4 cliques of  $Q_z^{(A)}, Q_z^{(B)}, Q_z^{(C)}$  and  $Q_z^{(D)}$ . The size of  $Z$  is

$$\sum_{i=0}^{\lfloor \frac{n}{5} \rfloor - 1} (4^5 - 16)^i 4^{n-5i-5}.$$

So there are total  $4|Z|$  cliques of size 4. On the other side, the size of singleton is

$$(4^5 - 16)^{\lfloor \frac{n}{5} \rfloor} 4^{n-5\lfloor \frac{n}{5} \rfloor}.$$

Therefore,

$$\begin{aligned} |\mathcal{Q}| &= 4 \sum_{i=0}^{\lfloor \frac{n}{5} \rfloor - 1} (4^5 - 16)^i 4^{n-5i-5} + (4^5 - 16)^{\lfloor \frac{n}{5} \rfloor} 4^{n-5\lfloor \frac{n}{5} \rfloor} \\ &= 4^{n-1} \left( 1 + 3 \left( \frac{4^5 - 16}{4^5} \right)^{\lfloor \frac{n}{5} \rfloor} \right). \end{aligned}$$

**Theorem 9.** Let  $\mathcal{C} \subseteq \Sigma^n$  be a 1-KDCC, then the redundancy of  $\mathcal{C}$  is at least  $\log 4 - o(1)$ .

*Proof:* An independent set of  $\mathcal{G}$  has the property that for any  $\mathbf{u}, \mathbf{v} \in \mathcal{G}$ ,  $\text{SynDef}_{\{\delta\}}(\mathbf{u}) \cap \text{SynDef}_{\{\delta\}}(\mathbf{v}) = \emptyset$  for

$\delta \in [4n]$ . This means an independent set of  $\mathcal{G}$  is a 1-KDCC. By Lemma 17, the size of any independent set is upper bounded by the size of a clique cover of  $\mathcal{G}$ , and we can construct a clique cover of size  $4^{n-1} (1 + 3 \left( \frac{4^5 - 16}{4^5} \right)^{\lfloor \frac{n}{5} \rfloor})$  by Lemma 18. So the size of a 1-KDCC is upper bounded by  $4^{n-1} (1 + 3 \left( \frac{4^5 - 16}{4^5} \right)^{\lfloor \frac{n}{5} \rfloor})$ , and the redundancy of it is at least  $n - \log(4^{n-1} (1 + 3 \left( \frac{4^5 - 16}{4^5} \right)^{\lfloor \frac{n}{5} \rfloor})) = 1 - o(1)$ . ■

With this theorem, it follows that the 1-KDCC  $\mathcal{C}_1^{KDCC}(n; a)$  we construct is almost optimal.

## VI. CONCLUSION AND FUTURE WORK

We investigate codes correcting synthesis defects and provide two type of codes to correct this error. In the first one, we assume we have already known the information of defective cycles. Although this seems to provide sufficient information about the errors, it is not enough to recover the strand. We construct  $t$ -KDCCs for  $t = 1, 2$ , and this shows the strategy to construct  $t$ -KDCC for general  $t$ . Besides, we derive a bound on the size of 1-KDCC, which shows our construction for 1-KDCC is almost optimal. In the second type of codes, each codeword is a set of  $M$  sequence, and they share the same synthesis defects. We utilize this feature, and provide  $t$ -SDCCs for  $t = 1, 2$ , where each codeword is divided into two parts with different coding schemes. The redundancy of our  $t$ -SDCCs is roughly  $\lambda_1 (\log n)^2 + M \log \log n$  and  $\lambda_2 (\log n)^2 + 2M \log n$  for  $t = 1, 2$ , respectively, for some constants  $\lambda_1$  and  $\lambda_2$ . In contrast, if we employ the best existing single deletions and two-deletion correcting codes for each sequence, then the redundancy for 1 and 2-SDCC will be at least  $M \log n$  and  $4M \log n$ .

As explained in Section III-A, if we want to construct a  $t$ -KDCC, then it suffices to construct a binary  $\mathbf{P}$ -bounded  $t$ -deletion correcting code. We only provide this code for  $t = 2$  in this paper, and the cases for  $t > 2$  are left to the future.

For the  $t$ -SDCC, we only provide the constructions for  $t \leq 2$ . The case  $t \geq 3$  is far more complicated. Even if we locate each defect in a small interval, we cannot tell which cycles are defective in a sequence. For example, three cycles  $\delta_1, \delta_2, \delta_3$  are defective, but if there is a sequence suffers from 2 deletions, then we do not know these two deletions are caused by  $\delta_1, \delta_2$  or  $\delta_2, \delta_3$  or  $\delta_1, \delta_3$ . So, it is not enough to let it belongs to only a  $(P_1, P_2, P_3)$ -bounded 3-deletion correcting code. Even if it also belongs to a  $(P_1, P_2)$ -bounded 2-deletion correcting code, we cannot recover it because we do not know the two interval. The problem of constructing  $t$ -SDCCs for  $t \geq 3$  is left for future work.

## REFERENCES

- [1] 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.
- [2] I. Shomorony and R. Heckel, "Information-theoretic foundations of dna data storage," *Foundations and Trends® in Communications and Information Theory*, vol. 19, no. 1, pp. 1–106, 2022.
- [3] M. Yu, X. Tang, Z. Li, W. Wang, S. Wang, M. Li, Q. Yu, S. Xie, X. Zuo, and C. Chen, "High-throughput dna synthesis for data storage," *Chemical Society Reviews*, 2024.
- [4] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using dna," *Nature Reviews Genetics*, vol. 20, no. 8, pp. 456–466, 2019.

- [5] A. Lenz, Y. Liu, C. Rashtchian, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding for efficient dna synthesis," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 2885–2890.
- [6] A. Lenz, S. Melcher, C. Rashtchian, and P. H. Siegel, "Multivariate analytic combinatorics for cost constrained channels and subsequence enumeration," *arXiv preprint arXiv:2111.06105*, 2021.
- [7] K. Makarychev, M. Z. Racz, C. Rashtchian, and S. Yekhanin, "Batch optimization for dna synthesis," *IEEE Transactions on Information Theory*, vol. 68, no. 11, pp. 7454–7470, 2022.
- [8] O. Elishco and W. Huleihel, "Optimal reference for dna synthesis," *IEEE Transactions on Information Theory*, 2023.
- [9] M. Abu-Sini, A. Lenz, and E. Yaakobi, "Dna synthesis using shortmers," in *2023 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2023, pp. 585–590.
- [10] J. Chrisnata and H. M. Kiah, "Deletion correcting codes for efficient dna synthesis," in *2023 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2023, pp. 352–357.
- [11] J. Lietard, A. Leger, Y. Erlich, N. Sadowski, W. Timp, and M. M. Somoza, "Chemical and photochemical error rates in light-directed synthesis of complex dna libraries," *Nucleic acids research*, vol. 49, no. 12, pp. 6687–6701, 2021.
- [12] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 1971–1985, 2017.
- [13] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion (corresp.)," *IEEE Transactions on Information Theory*, vol. 30, no. 5, pp. 766–769, 1984.
- [14] V. Guruswami and J. Hastad, "Explicit two-deletion codes with redundancy matching the existential bound," *IEEE Transactions on Information Theory*, vol. 67, no. 10, pp. 6384–6394, 2021.
- [15] R. R. Varshamov and G. Tenenholz, "A code for correcting a single asymmetric error," *Automatica i Telemekhanika*, vol. 26, no. 2, pp. 288–292, 1965.
- [16] S. Liu, I. Tjuawinata, and C. Xing, "Explicit construction of q-ary 2-deletion correcting codes with low redundancy," *IEEE Transactions on Information Theory*, 2024.
- [17] J. Chrisnata, H. M. Kiah, and E. Yaakobi, "Correcting deletions with multiple reads," *IEEE Transactions on Information Theory*, vol. 68, no. 11, pp. 7141–7158, 2022.
- [18] D. E. Knuth, "The sandwich theorem," *The Electronic Journal of Combinatorics*, vol. 1, no. 1, 1994.
- [19] J. Sima, N. Raviv, and J. Bruck, "Two deletion correcting codes from indicator vectors," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2375–2391, 2019.

## APPENDIX

To correct two deletions where each deletion is within an interval of length  $P_i$ , we distinguish if the two intervals are adjacent or not.

1) When the two intervals are adjacent or intersect

Denote  $\rho = P_1 + P_2$  and

$$I_i = \begin{cases} [(i-1)\rho + 1, (i+1)\rho], & \text{for } i \in \{1, \dots, \lceil n/\rho \rceil - 2\}, \\ [(i-1)\rho + 1, n], & \text{for } i = \lceil n/\rho \rceil - 1. \end{cases}$$

**Lemma 19** ([19, Theorem 1]). For any integer  $n \geq 3$ , there exist a sketch function  $\xi : \{0, 1\}^n \rightarrow \{0, 1\}^{r_0(n)}$ , computable in linear time, such that for any  $\mathbf{x} \in \{0, 1\}^n$ , given  $\xi(\mathbf{x})$  and  $\mathbf{x}_{[n] \setminus \{i, j\}}$ , one can uniquely recover  $\mathbf{x}$ , where  $i, j \in [n]$ , where  $r_0(n) = 7 \log n + O(\log \log n)$ .

**Construction 9.** Define  $\mathcal{E}_1 : \{0, 1\}^n \rightarrow \{0, 1\}^{r_1(P_1, P_2)}$  as follows:

for  $\mathbf{x} \in \{0, 1\}^n$ ,

$$\mathcal{E}_1(\mathbf{x}) = \left( \sum_{j=1}^{\lceil \frac{\lceil n/\rho \rceil - 1}{2} \rceil} \xi(\mathbf{x}_{I_{2j-1}}) \bmod 2^{7 \log \rho + O(\log \log \rho)}, \sum_{j=1}^{\lfloor \frac{\lceil n/\rho \rceil - 1}{2} \rfloor} \xi(\mathbf{x}_{I_{2j}}) \bmod 2^{7 \log \rho + O(\log \log \rho)} \right),$$

where  $r_1(P_1, P_2) = 14 \log \rho + O(\log \log \rho)$

**Lemma 20.** For any integers  $P_1, P_2 \geq 2$  and  $n \geq 3$ , there exists a sketch function  $\mathcal{E}_1 : \{0, 1\}^n \rightarrow \{0, 1\}^{r_1(P_1, P_2)}$ , computable in linear time, such that for any  $\mathbf{x} \in \{0, 1\}^n$ , if  $\mathbf{x}'$  is obtained by deleting two bits at two adjacent or overlapping intervals of lengths  $P_1, P_2$ , then given  $\mathcal{E}_1(\mathbf{x})$  and  $\mathbf{x}'$  together with the locations of this two intervals, we can uniquely recover  $\mathbf{x}$ , where  $r_1(P_1, P_2) = 14 \log(P_1 + P_2) + O(\log \log(P_1 + P_2))$

*Proof:* As  $\mathbf{x}'$  is obtained by deleting two bits at two adjacent or overlapping intervals of lengths  $P_1, P_2$ , the locations of two deletions are within an interval  $I$  of length  $\rho = P_1 + P_2$ . Furthermore, there exists a  $j \in [\lceil \frac{\rho}{2} \rceil - 1]$ , such that  $I \subseteq I_j$ . Since  $I_j = [(j-1)\rho + 1, (j+1)\rho]$ , we know  $\mathbf{x}'_{[(j-1)\rho+1, (j+1)\rho-2]}$  is a subsequence of  $\mathbf{x}_{[(j-1)\rho+1, (j+1)\rho]}$ , and  $\mathbf{x}_{[n] \setminus [(j-1)\rho+1, (j+1)\rho]} = \mathbf{x}'_{[n-2] \setminus [(j-1)\rho+1, (j+1)\rho-2]}$ . So we can get  $\xi(\mathbf{x}_{I_j}) \bmod 2^{7 \log \rho + O(\log \log \rho)}$  from  $\mathcal{E}_1(\mathbf{x})$ . By Lemma 19, given  $\xi(\mathbf{x}_{I_j})$ ,  $\mathbf{x}'_{[(j-1)\rho+1, (j+1)\rho-2]}$ , we can uniquely recover  $\mathbf{x}_{I_j}$ , so  $\mathbf{x}$  is recovered. ■

2) When the two intervals are separated by at least one symbol

We use the same analytical approach as in [14]. First, we use  $f_0(\mathbf{x}) = \sum_{i=1}^n x_i \bmod 3$  to identify the following 3 cases:

- i) the two deleted symbols are both 0;
- ii) the two deleted symbols are both 1;
- iii) the two deleted symbols are 1 and 0.

Besides, we need two more functions:

$$f_1(\mathbf{x}) = \sum_{i=1}^n i x_i,$$

and

$$f_2(\mathbf{x}) = \sum_{i=1}^n \binom{i}{2} x_i.$$

**Lemma 21** ([14, Lemma 5]). If we have deleted two 1's or two 0's in forming the subsequence  $\mathbf{x}'$  from  $\mathbf{x}$ , then  $f_1(\mathbf{x})$  and  $f_2(\mathbf{x})$  together with  $\mathbf{x}'$  identify  $\mathbf{x}$  uniquely.

**Lemma 22** ([14, Observation 2]). If we have deleted a 0 and a 1 from two nonadjacent intervals respectively in forming the subsequence  $\mathbf{x}'$  from  $\mathbf{x}$ , then  $f_1(\mathbf{x})$  and  $f_2(\mathbf{x})$  together with  $\mathbf{x}'$  identify  $\mathbf{x}$  uniquely.

*Proof:* As in [14], we insert a 0 and a 1 as far right as possible in the two intervals of  $\mathbf{x}'$  to make the value of  $f_1(\mathbf{x})$  correct.

When we move the 0 left passing a 1, the value of  $f_1(\mathbf{x})$  increases 1, so we need to move the 1 in the other interval left passing a 0 to keep the value of  $f_1(\mathbf{x})$  constant. During this



process, the order of the 0 and 1 will keep, since the moving bits in the left interval will never move to the right interval. So by Lemma 2 and Observation 2 of [14], there will never happen an overtake event, so  $f_2(\mathbf{x})$  will be monotonically increasing or decreasing in the process. ■

Combining Lemma 21 and 22, we can prove that if we have  $\mathbf{x}'$  together with  $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x})$ , then  $\mathbf{x}$  can be uniquely recover knowing the two nonadjacent intervals of two deletions located. Now we need to determine how many values do we need for  $f_1(\mathbf{x})$  and  $f_2(\mathbf{x})$ .

Since the two deleted bits can only move within their intervals, the bits outside these two intervals will keep their contributions to  $f_1(\mathbf{x})$  and  $f_2(\mathbf{x})$ . So it suffices to consider the bits in these two intervals.

First we consider  $f_1(\mathbf{x})$ . We analyze case by case:

- i) the two deleted symbols are both 0: For the  $i$ th interval, its contribution to  $f_1(\mathbf{x})$  will increase at most  $P_i - 1$  during the process of 0 moving left. So  $f_1(\mathbf{x})$  will increase at most  $P_1 + P_2 - 2$ .
- ii) the two deleted symbols are both 1: For the  $i$ th interval, its contribution to  $f_1(\mathbf{x})$  will increase at most  $P_i - 1$  during the process of 1 moving right. So  $f_1(\mathbf{x})$  will increase at most  $P_1 + P_2 - 2$ .
- iii) the two deleted symbols are 1 and 0: The value of  $f_1(\mathbf{x})$  is least when 1 is inserted in the front of the first interval and 0 is inserted in the end of the second interval, and is largest when 0 is inserted in the front of the first interval and 1 is inserted in the end of the second interval. The difference of them is  $j_2 + P_2 - 1 - j_1 < n$ , where  $j_i$  is the first index of  $i$ th interval.

Therefore, it has at most  $n$  values for  $f_1(\mathbf{x})$ , so it suffices to apply modular  $n$ .

Now we consider  $f_2(\mathbf{x})$ . Since we must keep  $f_1(\mathbf{x})$  constant when we insert the two bits in  $\mathbf{x}'$ , we do not have to consider all the possibilities of insertions. There are at most  $\min\{P_1, P_2\}$  possibilities of insertions to keep  $f_1(\mathbf{x})$  constant. By [14, Lemma 2], we can easily identify the extremal values of  $f_2(\mathbf{x})$ .

- i) the two deleted symbols are both 0: The value of  $f_2(\mathbf{x})$  will increase by  $\binom{j+1}{2} - \binom{j}{2} = j$  if the moving 0 moves left passing a 1 located at  $j$ , and will decrease by  $\binom{i}{2} - \binom{i-1}{2} = i - 1$  if the moving 0 moves right passing a 1 located at  $i$ . First we insert the two 0's as far away from each other as possible to keep  $f_1(\mathbf{x})$  correct. Then, every time we move the left 0 right passing a 1 located at  $i$ , the right 0 need to be moved left passing a 1 located at  $j$ . In this process,  $f_2(\mathbf{x})$  will increase  $j - i + 1$ . Suppose the indices of 1's in the left interval are  $\{i_1, \dots, i_p\}$ , and in the right interval are  $\{j_1, \dots, j_q\}$ , where  $p < P_1$  and  $q < P_2$ . When the two moving 0's are as far close to each other as possible, the value of  $f_2(\mathbf{x})$  grows to maximum, and will increase at most  $\sum_{k=1}^{\min\{p,q\}} (j_k - i_k + 1) < \min\{P_1, P_2\}n$  compared to the beginning.
- ii) the two deleted symbols are both 1: The value of  $f_2(\mathbf{x})$  will decrease by  $\binom{i+1}{2} - \binom{i}{2} = i$  if the moving 1 moves left passing a 0 located at  $i$ , and will increase by  $\binom{j}{2} - \binom{j-1}{2} = j - 1$  if the moving 1 moves right passing a 0

located at  $j$ . First we insert the two 1's as far close to each other as possible to keep  $f_1(\mathbf{x})$  correct. Then, every time we move the left 1 left passing a 0 located at  $i$ , the right 1 need to be moved right passing a 0 located at  $j$ . In this process,  $f_2(\mathbf{x})$  will increase  $j - i - 1$ . Suppose the indices of 0's in the left interval are  $\{i_1, \dots, i_p\}$ , and in the right interval are  $\{j_1, \dots, j_q\}$ , where  $p < P_1$  and  $q < P_2$ . When the two moving 1's are as far away from each other as possible, the value of  $f_2(\mathbf{x})$  grows to maximum, and will increase at most  $\sum_{k=1}^{\min\{p,q\}} (j_k - i_k - 1) < \min\{P_1, P_2\}n$  compared to the beginning.

- iii) the two deleted symbols are 1 and 0: By Lemma 22, The moving bits can never change their intervals. So we can consider the two case separately.

- 0 is inserted at the first interval, and 1 the second. Similar to the above two cases, we insert 0 and 1 as left as possible to keep  $f_1(\mathbf{x})$  correct. Once we move 0 right passing a 1 located at  $i$ , 1 need to be moved right passing a 0 located at  $j$ .  $f_2(\mathbf{x})$  will increase by  $j - i$  in this process. When the two moving 0 and 1 are as far right as possible, the value of  $f_2(\mathbf{x})$  will increase at most  $\min\{P_1, P_2\}n - 1$ .
- 1 is inserted at the first interval, and 0 the second. Similar to the previous,  $f_2(\mathbf{x})$  will increase at most  $\min\{P_1, P_2\}n - 1$ .

Therefore, it has at most  $\min\{P_1, P_2\}n$  values for  $f_2(\mathbf{x})$ , so it suffices to apply modular  $\min\{P_1, P_2\}n$  for  $f_2(\mathbf{x})$ .

**Construction 10.** Define  $\mathcal{E}_2 : \{0, 1\}^n \rightarrow \{0, 1\}^{r_2(n, P_1, P_2)}$  as follows:

for  $\mathbf{x} \in \{0, 1\}^n$ ,

$$\mathcal{E}_2(\mathbf{x}) = \left( f_0(\mathbf{x}) \bmod 3, f_1(\mathbf{x}) \bmod n + 1, f_2(\mathbf{x}) \bmod Pn \right),$$

where  $r_2(n, P_1, P_2) = 2 \log n + \log P + O(1)$ .

By Lemma 21 and 22, and the above analysis for ranges of  $f_1(\mathbf{x})$  and  $f_2(\mathbf{x})$ , we have the following lemma.

**Lemma 23.** For any integers  $P_1, P_2 \geq 2$  and  $n \geq 3$ , there exists a sketch function  $\mathcal{E}_2 : \{0, 1\}^n \rightarrow \{0, 1\}^{r_2(n, P_1, P_2)}$ , computable in linear time, such that for any  $\mathbf{x} \in \{0, 1\}^n$ , if  $\mathbf{x}'$  is obtained by deleting two bits at two non-overlapping intervals of lengths  $P_1, P_2$ , then given  $\mathcal{E}_2(\mathbf{x})$  and  $\mathbf{x}'$  together with the locations of this two intervals, we can uniquely recover  $\mathbf{x}$ , where  $r_2(n, P_1, P_2) = 2 \log n + \log P + O(1)$ .

**Construction 11.** Define  $\mathcal{E} : \{0, 1\}^n \rightarrow \{0, 1\}^{n+r(n, P_1, P_2)}$  as follows:

for  $\mathbf{x} \in \{0, 1\}^n$ ,

$$\mathcal{E}(\mathbf{x}) = \left( \mathbf{x}, \mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x}), \xi(\mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x})) \right),$$

where  $r(n, P_1, P_2) = r_1(P_1, P_2) + r_2(n, P_1, P_2) + r_0(r_1(P_1, P_2) + r_2(n, P_1, P_2)) = 2 \log n + 7 \log \log n + 14 \log(P_1 + P_2) + \log P + O(\log \log(P_1 + P_2))$ .

*Proof of Lemma 6:* To be convenient, we omit the parameters of  $r_0, r_1, r_2$  and  $r$ , and use them to denote the

lengths of  $\xi(\mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x}))$ ,  $\mathcal{E}_1(\mathbf{x})$ ,  $\mathcal{E}_2(\mathbf{x})$  and  $\mathcal{E}(\mathbf{x})$ , respectively. Since we know the information of the two intervals, we can distinguish if the two intervals are separated.

- If the intervals do not intersect with  $[n]$ : we know the part of  $\mathbf{x}$  is correct, so just set  $\mathbf{x} = \mathcal{E}'(\mathbf{x})_{[n]}$ .
- If the intervals intersect with  $[n]$ :
  - if the two intervals are not separated, we first recover  $\mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x})$  from  $\mathcal{E}(\mathbf{x})_{[n+r_1+r_2+1, n+r-2]}$  and  $\mathcal{E}(\mathbf{x})_{[n+1, n+r_1+r_2-2]}$ . It is obvious that  $\mathcal{E}(\mathbf{x})_{[n+r_1+r_2+1, n+r-2]}$  is a subsequence of  $\xi(\mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x}))$  of length  $r_0 - 2$ , and  $\mathcal{E}(\mathbf{x})_{[n+1, n+r_1+r_2-2]}$  is a subsequence of  $\mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x})$  of length  $r_1 + r_2 - 2$ . By Lemma 19, we can recover  $\mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x})$ . Then, we can recovery  $\mathbf{x}$  by using  $\mathcal{E}(\mathbf{x})_{[1, n-2]}$  and  $\mathcal{E}_1(\mathbf{x})$  by Theorem 20.
  - if the two intervals are separated, similar to above, we can recover  $\mathcal{E}_1(\mathbf{x}), \mathcal{E}_2(\mathbf{x})$ , and we use  $\mathcal{E}(\mathbf{x})_{[1, n-2]}$  and  $\mathcal{E}_2(\mathbf{x})$  to recover  $\mathbf{x}$  by Theorem 23.

■