Efficient and Explicit Balanced Primer Codes

Yeow Meng Chee, Han Mao Kiah, *Member, IEEE*, and Hengjia Wei

***Abstract*— To equip DNA-based data storage with random-access capabilities, Yazdi *et al.* (2018) prepended DNA strands with specially chosen address sequences called primers and provided certain design criteria for these primers. We provide explicit constructions of error-correcting codes that are suitable as primer addresses and equip these constructions with efficient encoding algorithms. Specifically, our constructions take cyclic or linear codes as inputs and produce sets of primers with similar error-correcting capabilities. Using certain classes of BCH codes, we obtain infinite families of primer sets of length n, minimum distance d with $(d+1)\log_4 n + O(1)$ redundant symbols. Our techniques involve reversible cyclic codes (1964), an encoding method of Tavares *et al.* (1971) and Knuth's balancing technique (1986). In our investigation, we also construct efficient and explicit binary balanced error-correcting codes and codes for DNA computing.**

***Index Terms*— Constrained coding, DNA-based data storage systems.**

I. Introduction

ADVANCES in synthesis and sequencing technologies have made DNA macromolecules an attractive medium for digital information storage. Instead of using binary sequences, the DNA-based data storage uses synthetic DNA strands, i.e., sequences comprising four bases `A`, `T`, `G` and `C`, to represent data. Besides being biochemically robust, DNA strands offer ultrahigh storage densities of $10^{15} - 10^{20}$ bytes per gram of DNA, as demonstrated in recent experiments (see [2, Table 1]). Therefore, in recent years, new error models were proposed and novel coding schemes were constructed by various authors (see [3] for a survey).

Unlike traditional medium, when we store information on DNA strands, the strands are *unordered*, i.e., they do not preserve the order in which they are stored. Hence, traditional file access methods are not applicable in such memories. In order to *read a specific block of the information* without retrieving all information blocks [1], Yazdi *et al.* proposed a strategy that allows random-access to encoded DNA strands and also, verified the feasibility of their architecture in a series of experiments [3], [5]. Their broad proposal is to prepend each information block with a specially chosen address sequence called *primer*, and make use of either the complement or the reverse complement of the primer to "fish" out the desired strand via a chemical process called *hybridization*, see Figure 1.

Two kinds of coding problems arise when we implement this proposal for accurate access of DNA strands. The first coding problem is to design the primers so that they are distinguishable in some sense. The other is to encode the data so that the information blocks do not contain any primers. The problem of encoding information blocks is studied in [3], and in this paper, we focus on the problem of primer design. We follow the design considerations provided by Yazdi *et al.* [6]. Specifically, Yazdi *et al.* require the primers to

- have a high minimum Hamming distance;
- be weakly mutually uncorrelated;
- avoid secondary structure; and
- have almost constant GC-content (close to 50%).

In their work, Yazdi *et al.* gave a number of constructions for primers satisfying two or more of the above conditions. In particular, they provided an iterative construction satisfying all the four conditions [6, Construction 11]. However, their iterative construction requires *short* codes satisfying all the above conditions and a collection of disjoint subcodes. Hence, it is unclear whether the codes can be constructed efficiently and whether efficient encoding is possible.

In this paper, we continue this investigation and provide efficient and explicit constructions of error-correcting codes that are suitable as primer addresses. Among others, one of our constructions produces the first infinite family of codes which satisfy all the above conditions with at most $(d+1)\log_4 n + O(1)$ redundant symbols, where n is the block length and d is the minimum distance. Compared with the iterative construction provided by Yazdi *et al.*, our constructions only rely on the existence of some known cyclic codes, rather than codes with special properties.

Our techniques involve *Knuth's balancing technique* [7], *reversible cyclic codes* [8], and an encoding method of Tavares *et al.* [9]. In particular, we propose novel modifications of Knuth's balancing technique to construct binary balanced error-correcting codes from cyclic codes, increasing the redundancy only by $\log_2 n + 1$. Compared with the

Manuscript received June 3, 2019; revised January 5, 2020; accepted February 19, 2020. Date of publication March 3, 2020; date of current version August 18, 2020. The work of Yeow Meng Chee was supported in part by the Singapore Ministry of Education under Grant MOE2017-T3-1-007 and Grant MOE2015-T2-2-086. The work of Han Mao Kiah and Hengjia Wei was supported in part by the Singapore Ministry of Education under Grant MOE2015-T2-2-086. This article was presented in part at the 2019 IEEE International Symposium on Information Theory. *(Corresponding author: Hengjia Wei.)*

Yeow Meng Chee is with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 117576.

Han Mao Kiah is with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371.

Hengjia Wei is with the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beersheba 84105, Israel (e-mail: hjwei05@gmail.com).

Communicated by V. Sidorenko, Associate Editor for Coding Theory.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2020.2977915

[1] For coding strategies when retrieving *all* information blocks, we refer the interested reader to Lenz *et al.* [4] and the references therein.

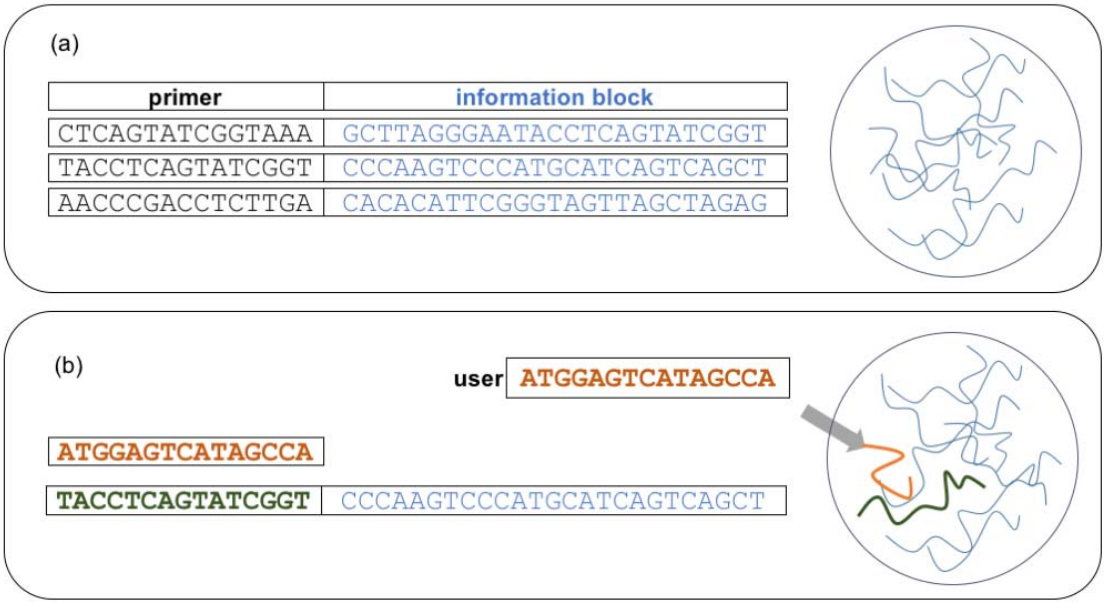

Fig. 1. (a) Suppose that there are three information blocks in the pool. Prepend each of them with a primer from the primer code. (b) In order to retrieve the second information block, the user uses the complement of the corresponding primer as a "bait" and inject it into the pool. Then the target primer of the second information block attaches itself to the "bait" so that the desired strand is retrieved.

TABLE I

SUMMARY OF CONSTRUCTIONS FOR CODES OF LENGTH n AND DISTANCE d

Constr.	Input	Output	Redundancy for Infinite Family	Prior Works
A	binary cyclic code	balanced $(n,d)_2$-code	$(t+1)\log_2 n+1$, where $t=d/2-1$ (*c.f.* Corollary 11)	[10], [11]
B	two binary linear codes	GC-balanced $(n,d)_4$-code	$(2t+1)\log_4 n+2t$, where $t=\lceil (d-1)/2\rceil$ (*c.f.* Corollary 14)	[10]
C	binary linear code, and ℓ-APD-constrained code	GC-balanced $(n,d;\kappa,f)_4$-primer code	$(2t+1)\log_4 n+O(1)$, where $t=\lceil (d-1)/2\rceil$ (no GC-balanced constraint) $(d+1)\log_4 n+O(1)$ (GC-balanced) (*c.f.* Corollary 20)	[6]
D	cyclic code containing 1^n	almost GC-balanced $(n,d)_4$-code that is κ-WMU	N.A.	[6]
E	reversible cyclic code containing 1^n, and rc-generating set of polynomials	primer code with $\kappa=f$	$(d+1)\log_4(n+1)$ (*c.f.* Corollary 30)	N.A.
F	reversible cyclic code containing 1^n, and rc2-generating set of polynomials	GC-balanced (n,d)-DNA computing codes	$(d+1)\log_4(n+1)$ (*c.f.* Corollary 35)	[12]

previous work of Weber *et al.* [10], this method does not require short balanced error-correcting codes as inputs, and can produce codes with less redundant bits. We note that reversible cyclic codes have been studied in another coding application for DNA computing. It turns out our techniques can be modified to improve code constructions in the latter application. We also revisit the work of Tavares *et al.* [9] that efficiently encodes messages into cyclic classes of a cyclic code and adapt their method to give efficient encoding schemes for our codes. We provide a summary of our constructions and the redundancy of the corresponding codes in Table I.

II. PRELIMINARY AND CONTRIBUTIONS

Let $\mathbb{F}_q$ denote the finite field of size q. Two cases of special interest are $q=2$ and $q=4$. In the latter case, we let ω denote a primitive element of $\mathbb{F}_4$ and identify the elements of $\mathbb{F}_4$ with the four DNA bases $\Sigma=\{\mathtt{A},\mathtt{C},\mathtt{T},\mathtt{G}\}$. Specifically,

$$0\leftrightarrow \mathtt{A},\quad 1\leftrightarrow \mathtt{T},\quad \omega\rightarrow \mathtt{C},\quad \omega+1\leftrightarrow \mathtt{G}.$$

For an element $x\in\Sigma$, its *Watson-Crick complement* is

$$\bar{x}=\begin{cases}\mathtt{T}, & \text{if } x=\mathtt{A};\\ \mathtt{A}, & \text{if } x=\mathtt{T};\\ \mathtt{G}, & \text{if } x=\mathtt{C};\\ \mathtt{C}, & \text{if } x=\mathtt{G}.\end{cases}$$

Hence, for an element $x\in\mathbb{F}_4$, its Watson-Crick complement corresponds to $x+1$.

Let n be a positive integer. Let $[n]$ denote the set $\{1,2,\ldots,n\}$, while $[\![n]\!]$ denotes the set $\{0,1,\ldots,n-1\}$. For a word $\boldsymbol{a}=(a_1,\ldots,a_n)\in\mathbb{F}_q^n$, let $\boldsymbol{a}[i]$ denote the ith symbol a_i and $\boldsymbol{a}[i,j]$ denote the subword of $\boldsymbol{a}$ starting at position i

and ending at position j. In other words,

$$\boldsymbol{a}[i,j] = \begin{cases} (a_i, a_{i+1}, \ldots, a_j), & \text{if } i \leq j; \\ (a_i, a_{i-1}, \ldots, a_j), & \text{if } i > j. \end{cases}$$

Moreover, the *reverse* of $\boldsymbol{a}$, denoted as $\boldsymbol{a}^r$, is $(a_n, a_{n-1}, \ldots, a_1)$; the *complement* $\overline{\boldsymbol{a}}$ of $\boldsymbol{a}$ is $(\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_n)$, where $\overline{x} = x + 1$ for $x \in \mathbb{F}_2$ or $x \in \mathbb{F}_4$; and the *reverse-complement* $\boldsymbol{a}^{rc}$ of $\boldsymbol{a}$ is $\overline{\boldsymbol{a}^r}$.

For two words $\boldsymbol{a}$ and $\boldsymbol{b}$, we use $\boldsymbol{ab}$ to denote the concatenation of $\boldsymbol{a}$ and $\boldsymbol{b}$, and $\boldsymbol{a}^\ell$ to denote the sequence of length ℓn comprising ℓ copies of $\boldsymbol{a}$.

A *q-ary code* $\mathcal{C}$ of length n is a collection of words from $\mathbb{F}_q^n$. For two words $\boldsymbol{a}$ and $\boldsymbol{b}$ of the same length, we use $d(\boldsymbol{a}, \boldsymbol{b})$ to denote the Hamming distance between them. A code $\mathcal{C}$ has *minimum Hamming distance* d if any two distinct codewords in $\mathcal{C}$ are at least distance d apart. Such a code is denoted as an $(n, d)_q$-code. Its *size* is given by $|\mathcal{C}|$, while its *redundancy* is given by $n - \log_q |\mathcal{C}|$. An *$[n, k, d]_q$-linear code* is an $(n, d)_q$-code that is also a k-dimension vector subspace of $\mathbb{F}_q^n$. Hence, an $[n, k, d]_q$-linear code has redundancy $n - k$.

A. Cyclic and Reversible Codes

For a vector $\boldsymbol{a} \in \mathbb{F}_q^n$, let $\boldsymbol{\sigma}^i(\boldsymbol{a})$ be the vector obtained by cyclically shifting the components of $\boldsymbol{a}$ to the right i times. So, $\boldsymbol{\sigma}^1(\boldsymbol{a}) = (a_n, a_1, a_2, \ldots, a_{n-1})$. An *$[n, k, d]_q$-cyclic code* $\mathcal{C}$ is an $[n, k, d]_q$-linear code that is closed under cyclic shifts. In other words, $\boldsymbol{a} \in \mathcal{C}$ implies $\boldsymbol{\sigma}^1(\boldsymbol{a}) \in \mathcal{C}$.

Cyclic codes are well-studied because of their rich algebraic structure. In the theory of cyclic codes (see for example, MacWilliams and Sloane [13, Chapter 7]), we identify a word $\boldsymbol{c} = (c_i)_{i \in [\![n]\!]}$ of length n with the polynomial $\sum_{i=0}^{n-1} c_i X^i$ in the quotient ring $\mathbb{F}_q[X]/\langle X^n - 1 \rangle$. Given a cyclic code $\mathcal{C}$ of length n and dimension k, there exists a unique monic polynomial $g(X)$ of degree $n - k$ such that $\mathcal{C}$ is given by the set $\{m(X)g(X) : \deg m < k\}$. The polynomial $g(X)$ is referred to as the *generator polynomial* of $\mathcal{C}$ and we write $\mathcal{C} = \langle g(X) \rangle$. We continue this discussion on this algebraic structure in Section VI, where we exploit certain polynomial properties for efficient encoding.

When d is fixed, there exists a class of Bose-Chaudhuri-Hocquenghem (BCH) codes that are cyclic codes whose redundancy is asymptotically optimal.

Theorem 1 (Primitive Narrow-Sense BCH Codes [14, Theorem 10]): Fix $m \geq 1$ and $2 \leq d \leq 2^{\lceil m/2 \rceil} - 1$. Set $n = 2^m - 1$ and $t = \lceil (d-1)/2 \rceil$. There exists an $[n, k, d]_2$-cyclic code $\mathcal{C}$ with $k \geq n - tm$. In other words, $\mathcal{C}$ has redundancy at most $t \log_2(n+1)$.

A cyclic code $\mathcal{C}$ is called *reversible* if $\boldsymbol{a} \in \mathcal{C}$ implies $\boldsymbol{a}^r \in \mathcal{C}$. A reversible cyclic code is also known as a cyclic *LCD code* (*linear code with complementary dual*) and has been studied extensively [8], [15]–[17]. In this paper, reversible cyclic codes containing the all-one vector 1^n are of particular interest. Suppose that $\mathcal{C}$ is one such code. Then for any codeword $\boldsymbol{a} \in \mathcal{C}$, both its complement $\overline{\boldsymbol{a}} = \boldsymbol{a} + 1^n$ and its reverse-complement $\boldsymbol{a}^{rc} = \boldsymbol{a}^r + 1^n$ belong to $\mathcal{C}$.

Recently, Li *et al.* [16] explored two other classes of BCH codes and determined their minimum distances and dimensions. These codes are reversible cyclic and contain the all-one vector.

Theorem 2 (Li et al. *[16]):* Let $m \geq 2$, $m \neq 3$ and $1 \leq \tau \leq \lceil m/2 \rceil$. Let q be even and set $n = q^m - 1$ and $d = q^\tau - 1$. There exists an $[n, k, d]_q$-reversible cyclic code that contains 1^n and has dimension

$$k = \begin{cases} n - (d - q + 1)m, & \text{if } m \geq 5 \text{ is odd and } \tau = \frac{m+1}{2}; \\ n - (d-1)m, & \text{otherwise.} \end{cases}$$

In other words, $\mathcal{C}$ has redundancy at most $(d-1)\log_q(n+1)$.

B. Balanced Codes

A binary word of length n is *balanced* if $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$ bits are zero, while a quaternary word of length n is GC-*balanced* if $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$ symbols are either G or C. A binary (or quaternary) code is *balanced* (resp. GC-*balanced*) if all its codewords are balanced (resp. GC-balanced).

Motivated by applications in laser disks, Knuth [7] studied balanced binary codes and proposed an efficient method to encode an arbitrary binary message to a binary balanced codeword by introducing $\log_2 n$ redundant bits. In this paper, we consider balanced codes with error-correcting capabilities. Previous constructions of balanced error-correcting codes can be found in [11], [18]–[20]. Van Tilborg and Blaum [18] proposed a concatenation method to obtain codes of moderate rates. Al-Bassam and Bose [11] constructed several classes of balanced codes which can correct up to t errors, for $1 \leq t \leq 4$. Fu and Wei [19] studied a class of balanced error-correcting codes in which the complement of each codeword is also contained in the codebook. They derived an upper bound for such codes and obtained some optimal codes achieving this upper bound by using tools from combinatorial design theory. Recently, Weber *et al.* [10] extended Knuth's scheme to include error-correcting capabilities. Specifically, their construction takes two input codes of distance d: a linear code of length n and a short *balanced* code $\mathcal{C}_p$; and outputs a long balanced code of distance d. Even though the balanced code $\mathcal{C}_p$ is only required to be of size n, it is unclear how to find one efficiently, especially when d grows with n.

On the other hand, GC-balanced codes have been extensively studied in the context of DNA computing and DNA-based storage (see [3], [21], [22] for a survey). However, most constructions are based on search heuristics or apply to a restricted set of parameters. Recently, Yazdi *et al.* [6] introduced the coupling construction (Lemma 6) that takes two binary error-correcting codes, one of which is balanced, as inputs and outputs a GC-balanced error-correcting code. As with the construction of Weber *et al.* [10], it is unclear how to find the balanced binary error-correcting code efficiently.

In this work, we *avoid these requirements of additional balanced codes*. Specifically, we provide construction that takes a binary cyclic code (or two binary linear codes) and outputs a binary balanced code (resp. a GC-balanced code) with error-correcting capabilities.

C. Primer Codes

In order to introduce random access to DNA-based data storage systems, Yazdi *et al.* [6] proposed the following criteria for the design of primer addresses.

Definition 3: A code $\mathcal{C}$ of length n is *κ-weakly mutually uncorrelated (κ-WMU)* if for all $\ell \geq \kappa$, no proper prefix of length ℓ of a codeword appears as a suffix of another codeword (including itself). In other words, for any two codewords $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$, not necessarily distinct, and $\kappa \leq \ell < n$,

$$\boldsymbol{a}[1, \ell] \neq \boldsymbol{b}[n - \ell + 1, n].$$

When $\mathcal{C}$ is 1-WMU, we say that $\mathcal{C}$ is mutually uncorrelated (MU).

During the amplification process, a primer dimer byproduct is formed when two primers used for selection attach to each other or hybridize. To avoid these undesired hybridization, we have the following definition.

Definition 4: A code $\mathcal{C}$ of length n is said to *avoid primer dimer byproducts of effective length f* (f-APD) if the reverse complement and the complement of any substring of length f in a codeword does not appear in as a substring of another codeword (including itself). In other words, for any two codewords $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$, not necessarily distinct, and $1 \leq i, j \leq n + 1 - f$, we have

$$\overline{\boldsymbol{a}}[i, i + f - 1] \notin \{\boldsymbol{b}[j, j + f - 1], \boldsymbol{b}[j + f - 1, j]\}.$$

For primer design in DNA-based storage, WMU codes are desired to be GC-balanced, have large Hamming distance and avoid primer dimer byproducts.

Definition 5: A code $\mathcal{C} \in \mathbb{F}_q^n$ is an *$(n, d; \kappa, f)_q$-primer code* if the following are satisfied:

(P1) $\mathcal{C}$ is an $(n, d)_q$-code;
(P2) $\mathcal{C}$ is κ-WMU;
(P3) $\mathcal{C}$ is an f-APD code.

Furthermore, if $\mathcal{C}$ is balanced or GC-balanced, then $\mathcal{C}$ is an *$(n, d; \kappa, f)_q$-balanced primer code*.

Yazdi *et al.* [6] provided a number of constructions for WMU codes which satisfy some combinations of the constraints (P1), (P2) and (P3). In particular, Yazdi *et al.* provided the following coupling construction.

Lemma 6 (Coupling Construction - Yazdi et al. *[6]):* For $i \in [2]$, let $\mathcal{C}_i$ be an $(n, d_i)_2$-code of size M_i. Define the map $\Psi : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \Sigma^n$ such that $\Psi(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{c}$, where for $i \in [n]$,

$$c_i = \begin{cases} \mathtt{A}, & \text{if } a_i b_i = 00; \\ \mathtt{T}, & \text{if } a_i b_i = 01; \end{cases} \quad c_i = \begin{cases} \mathtt{C}, & \text{if } a_i b_i = 10; \\ \mathtt{G}, & \text{if } a_i b_i = 11. \end{cases}$$

Then the code $\mathcal{C} \triangleq \{\Psi(\boldsymbol{a}, \boldsymbol{b}) : \boldsymbol{a} \in \mathcal{C}_1, \boldsymbol{b} \in \mathcal{C}_2\}$ is an $(n, d)_4$-code of size $M_1 M_2$, where $d = \min\{d_1, d_2\}$. Furthermore,

(i) if $\mathcal{C}_1$ is balanced, $\mathcal{C}$ is GC-balanced;
(ii) if $\mathcal{C}_2$ is κ-WMU, then $\mathcal{C}$ is also κ-WMU;
(iii) if $\mathcal{C}_2$ is an f-APD code, then $\mathcal{C}$ is also an f-APD code.

Yazdi *et al.* also provided an iterative construction for primer codes satisfying all the constraints, i.e., balanced primer codes. However, the construction requires a short balanced primer code and a collection of subcodes, some of which are disjoint. Hence, it is unclear whether the code can be constructed efficiently and whether efficient encoding is possible.

In this work, we provide constructions that take cyclic, reversible cyclic or linear codes as inputs and produce primer or balanced primer codes as outputs. Using known families of cyclic codes given by Theorems 1 and 2, we obtain infinite families of primer codes and provide explicit upper bounds on the redundancy. We discuss lengths of runs, i.e., DNA homopolymers, in the proposed primer codes. We also describe methods that efficiently encode into these codewords.

D. Our Contributions

In this paper, we study balanced codes, primer codes and other related coding problems. Our contributions are as follow:

(i) In Section III, we propose efficient methods to construct both balanced and GC-balanced error-correcting codes. In comparison with the method of Weber *et al.* [10] which requires short balanced error-correcting codes, our method uses only cyclic and linear codes as inputs, and always produces codes with fewer redundant symbols. Compared with the work of Al-Bassam and Bose [11] that constructed balanced codes correcting up to four errors, our method works for more general parameters. For some specific parameters our codes have less redundancy.
(ii) In Section IV, we provide three explicit constructions of primer codes, all of which only take some known codes as inputs. The first one is based on the coupling construction (Lemma 6). Using binary error-correcting codes constructed in Section III as inputs, it produces the first infinite class of $(n, d; \kappa, f)_4$-balanced primer codes whose redundancy is $(d+1)\log_4 n + O(1)$. The other two constructions rely on cyclic codes and use less redundancy albeit for a specific set of parameters. In particular, we have a class of $(n, d; \kappa, \kappa)_4$-primer codes with redundancy $(d + 1)\log_4(n + 1)$.
(iii) In Section V, we construct codes for DNA computing. We provide a class of GC-balanced $(n, d)_4$-DNA computing codes with minimum distance $d \approx \sqrt{n}$ and redundancy $(d + 1)\log_4(n + 1)$. In contrast, the GC-balanced $(n, d)_4$-DNA computing codes from [12] have minimum distance nearly $n/2$ but much smaller size.
(iv) In Section VI, we adapt the technique of Tavares *et al.* to efficiently encode messages into codes constructed in this paper.

III. Balanced Error-Correcting Codes

The celebrated Knuth's balancing technique [7] is a linear-time algorithm that maps a binary message of length m to a balanced word of length approximately $m + \log m$. The technique first finds an index z such that flipping the first z bits yields a balanced word $\boldsymbol{c}$. Then Knuth appends a short balanced word $\boldsymbol{p}$ that represents the index z. Hence, $\boldsymbol{cp}$ is the resulting codeword and the redundancy of the code is equal

to the length of $\boldsymbol{p}$, which is approximately $\log m$. The crucial observation demonstrated by Knuth is that such an index z always exists and z is commonly referred to as the *balancing index*.

Recently, Weber *et al.* [10] modified Knuth's balancing technique to endow the code with error-correcting capabilities. Their method requires two error-correcting codes as inputs: an $(m,d)_2$ code $\mathcal{C}_m$ and a short $(p,d)_2$ balanced code $\mathcal{C}_p$, where $|\mathcal{C}_p| \geq m$. Given a message, they first encode it into a codeword $\boldsymbol{m} \in \mathcal{C}_m$. Then they find the balancing index z of $\boldsymbol{m}$ and flip the first z bits to obtain a balanced $\boldsymbol{c}$. Using $\mathcal{C}_p$, they encode z into a balanced word $\boldsymbol{p}$ and the resulting codeword is $\boldsymbol{cp}$. Since both $\mathcal{C}_m$ and $\mathcal{C}_p$ have distance d, the resulting code has minimum distance d.

Now, this method introduces p *additional* redundant bits and since p is necessarily at least d, the method introduces more than $\log_2\ n$ bits of redundancy when d is big. Furthermore, the method requires the existence of a short balanced code $\mathcal{C}_p$. We overcome this obstacle in our next two constructions. Specifically, Construction A and B require only a cyclic code and a linear code, respectively. Both constructions do *not* require short balanced codes. In particular, Construction A introduces only $\log_2\ n+1$ *additional* bits of redundancy, regardless the value of d.

A. Binary Balanced Error-Correcting Codes

Our first construction modifies Knuth's balancing technique and uses a binary cyclic code as input. Since the lengths of the cyclic codes from Theorem 1 and Theorem 2 are odd, in this subsection we assume that n is odd so that the proof is easier to write. We note that our method also works for even n and state the result for even n without proof in the end of this subsection.

Let n be odd. In contrast with Knuth's balancing technique, we *always flip the first* $(n+1)/2$ *bits* of a word $\boldsymbol{a}$. However, this does not guarantee a balanced word. Nevertheless, if we consider all cyclic shifts of $\boldsymbol{a}$, i.e., $\boldsymbol{\sigma}^i(\boldsymbol{a})$ for $i \in [\![n]\!]$, then flipping the first $(n+1)/2$ bits of one of these shifts must yield a balanced word.

Formally, let $\phi : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be the map where $\phi(\boldsymbol{a}) = \boldsymbol{a} + 1^{(n+1)/2}0^{(n-1/2)}$. In other words, the map ϕ flips the first $(n+1)/2$ bits of $\boldsymbol{a}$. For $\boldsymbol{a} \in \mathbb{F}_2^n$, denote its Hamming weight as $\mathrm{wt}(\boldsymbol{a})$. Let $\mathrm{wt}_1(\boldsymbol{a})$ be the Hamming weight of the first $(n+1)/2$ bits and $\mathrm{wt}_2(\boldsymbol{a})$ be the Hamming weight of the last $(n-1)/2$ bits. So, we have $\mathrm{wt}(\boldsymbol{a}) = \mathrm{wt}_1(\boldsymbol{a}) + \mathrm{wt}_2(\boldsymbol{a})$. We have the following crucial lemma.

Lemma 7: Let n be odd. For $\boldsymbol{a} \in \mathbb{F}_2^n$, we can find $i \in [\![n]\!]$ such that $\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))$ has weight either $(n-1)/2$ or $(n+1)/2$.

Proof: Let $\boldsymbol{a}' = \boldsymbol{\sigma}^{(n+1)/2}(\boldsymbol{a})$. Then the first $(n-1)/2$ bits of $\boldsymbol{a}'$ are exactly the last $(n-1)/2$ bits of $\boldsymbol{a}$ and so

$$\mathrm{wt}_2(\boldsymbol{a}) \leq \mathrm{wt}_1(\boldsymbol{a}') \leq \mathrm{wt}_2(\boldsymbol{a}) + 1.$$

We first consider the case when $\mathrm{wt}(\boldsymbol{a})$ is even. Assume that $\mathrm{wt}(\boldsymbol{a}) = 2w$. If $\mathrm{wt}_1(\boldsymbol{a}) \leq w$, then $\mathrm{wt}_1(\boldsymbol{a}') \geq \mathrm{wt}_2(\boldsymbol{a}) = 2w - \mathrm{wt}_1(\boldsymbol{a}) \geq w$. Note that shifting the components of $\boldsymbol{a}$ once only increases or decreases the value of $\mathrm{wt}_1(\boldsymbol{a})$ by at most one. It follows that we can find an integer i such that $\mathrm{wt}_1(\boldsymbol{\sigma}^i(\boldsymbol{a})) = w$, and so

$$\begin{aligned}\mathrm{wt}(\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))) &= \mathrm{wt}_1(\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))) + \mathrm{wt}_2(\phi(\boldsymbol{\sigma}^i(\boldsymbol{a})))\\ &= ((n+1)/2 - w) + w = (n+1)/2.\end{aligned}$$

Similarly, if $\mathrm{wt}_1(\boldsymbol{a}) > w$, since $\mathrm{wt}_1(\boldsymbol{a}') \leq \mathrm{wt}_2(\boldsymbol{a}) + 1 = 2w - \mathrm{wt}_1(\boldsymbol{a}) + 1 \leq w$, we can still find i such that $\mathrm{wt}_1(\boldsymbol{\sigma}^i(\boldsymbol{a})) = w$ and $\mathrm{wt}(\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))) = (n+1)/2$.

Next, we assume that the weight is odd, or, $\mathrm{wt}(\boldsymbol{a}) = 2w+1$. If $\mathrm{wt}_1(\boldsymbol{a}) < w+1$, then $\mathrm{wt}_1(\boldsymbol{a}') \geq \mathrm{wt}_2(\boldsymbol{a}) = 2w+1 - \mathrm{wt}_1(\boldsymbol{a}) \geq w+1$; if $\mathrm{wt}_1(\boldsymbol{a}) \geq w+1$, then $\mathrm{wt}_1(\boldsymbol{a}') \leq \mathrm{wt}_2(\boldsymbol{a}) + 1 = 2w+1 - \mathrm{wt}_1(\boldsymbol{a}) + 1 \leq w+1$. In both cases we can always find i such that $\mathrm{wt}_1(\boldsymbol{\sigma}^i(\boldsymbol{a})) = w+1$, and so

$$\begin{aligned}\mathrm{wt}(\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))) &= \mathrm{wt}_1(\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))) + \mathrm{wt}_2(\phi(\boldsymbol{\sigma}^i(\boldsymbol{a})))\\ &= ((n+1)/2 - w - 1) + w = (n-1)/2.\end{aligned}$$

Remark: In Lemma 7, we show that we can balance some shift of $\boldsymbol{a}$ by flipping its first $(n+1)/2$ bits. In fact, we can also balance a shift of $\boldsymbol{a}$ (not necessary the same shift) by flipping its first $(n-1)/2$ bits. This observation is used in the construction of DNA computing codes.

We can further show that there are at least two shifts of $\boldsymbol{a}$ which can be balanced. This can be used to increase the size of the balanced code as long as the two shifts are not the same.

Lemma 8: Let n be odd. For $\boldsymbol{a} \in \mathbb{F}_2^n$, there are at least two different indices $i, j \in [\![n]\!]$ such that both $\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))$ and $\phi(\boldsymbol{\sigma}^j(\boldsymbol{a}))$ have the weight $(n-1)/2$ or $(n+1)/2$.

Proof: We first assume $\mathrm{wt}(\boldsymbol{a}) = 2w$. According to the proof of Lemma 7, there is $i \in [\![n]\!]$ such that $\mathrm{wt}_1(\boldsymbol{\sigma}^i(\boldsymbol{a})) = w$, and so $\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))$ has weight $(n+1)/2$. Now let $j = i + (n+1)/2$. Then $\mathrm{wt}_1(\boldsymbol{\sigma}^j(\boldsymbol{a})) = w$ or $w+1$. If $\mathrm{wt}_1(\boldsymbol{\sigma}^j(\boldsymbol{a})) = w$, $j \pmod n$ is the other index. If $\mathrm{wt}_1(\boldsymbol{\sigma}^j(\boldsymbol{a})) = w+1$, we claim that there is an index k, $i < k < j$, such that $\mathrm{wt}_1(\boldsymbol{\sigma}^k(\boldsymbol{a})) = w$. Otherwise, we have $\mathrm{wt}_1(\boldsymbol{\sigma}^\ell(\boldsymbol{a})) = w+1$, for all $i < \ell < j$. It is easy to see that $\boldsymbol{\sigma}^i(\boldsymbol{a}) = 1\boldsymbol{\chi}0\boldsymbol{\chi}1$ for some $\boldsymbol{\chi} \in \mathbb{F}_2^{(n-3)/2}$. Then $\boldsymbol{\sigma}^j(\boldsymbol{a}) = 0\boldsymbol{\chi}11\boldsymbol{\chi}0$ and $\mathrm{wt}_1(\boldsymbol{\sigma}^j(\boldsymbol{a})) = w$, a contradiction.

For $\mathrm{wt}(\boldsymbol{a}) = 2w+1$, the proof is the same. According to the proof of Lemma 7, there is $i \in [\![n]\!]$ such that $\mathrm{wt}_1(\boldsymbol{\sigma}^i(\boldsymbol{a})) = w+1$. Now let $j = i + (n+1)/2$. Then $\mathrm{wt}_1(\boldsymbol{\sigma}^j(\boldsymbol{a})) = w$ or $w+1$. If $\mathrm{wt}_1(\boldsymbol{\sigma}^j(\boldsymbol{a})) = w+1$, $j \pmod n$ is the other index. If $\mathrm{wt}_1(\boldsymbol{\sigma}^j(\boldsymbol{a})) = w$, we claim that there is an index k, $i < k < j$, such that $\mathrm{wt}_1(\boldsymbol{\sigma}^k(\boldsymbol{a})) = w+1$. Otherwise, we have $\mathrm{wt}_1(\boldsymbol{\sigma}^\ell(\boldsymbol{a})) = w$ for all $i < \ell < j$. Then $\boldsymbol{\sigma}^i(\boldsymbol{a}) = 0\boldsymbol{y}1\boldsymbol{y}0$ for some $\boldsymbol{y} \in \mathbb{F}_2^{(n-3)/2}$, and so $\mathrm{wt}_1(\boldsymbol{\sigma}^j(\boldsymbol{a})) = w+1$, a contradiction. □

Before we describe our construction, we introduce the notion of cyclic equivalence classes. Given a cyclic code $\mathcal{B}$ of length n, we define the following equivalence relation: $\boldsymbol{a} \underset{\text{cyc}}{\sim} \boldsymbol{b}$ if and only if $\boldsymbol{a} = \boldsymbol{\sigma}^i(\boldsymbol{b})$ for some $i \in [n]$; and partition the codewords $\mathcal{B}$ into classes. We use $\mathcal{B}/\underset{\text{cyc}}{\sim}$ to denote a set of representatives.

Construction A: Let n be an odd integer.

INPUT: An $[n,k,d]_2$-cyclic code $\mathcal{B}$.

OUTPUT: A balanced $(n+1,d')_2$-code $\mathcal{C}$ of size at least $2^k/n$, where $d' = 2\lceil d/2 \rceil$.

- Let $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m$ be the set of representatives $\mathcal{B}/\underset{\text{cyc}}{\sim}$.

- For each $\boldsymbol{u}_i$, find $j_i \in [n]$ such that $\phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))$ has weight $(n-1)/2$ or $(n+1)/2$.
- For $i \in [m]$, append a check bit to $\phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))$ so that its weight is $(n+1)/2$ and denote the modified vector as $\boldsymbol{v}_i$. In other words,

$$\boldsymbol{v}_i = \begin{cases} \phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))0, & \text{if } \mathrm{wt}(\phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))) = \frac{n+1}{2}; \\ \phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))1, & \text{if } \mathrm{wt}(\phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))) = \frac{n-1}{2}. \end{cases}$$

- Set $\mathcal{C} = \{\boldsymbol{v}_i : 1 \le i \le m\}$.

Example 9: Let $n = 9$ and $d = 3$. Consider two codewords 001001001 and 000000000 from a $(9, 3)_2$-cyclic code.

Since $\phi(\boldsymbol{\sigma}^2(001001001)) = \phi(010010010) = 101100010$, which has weight 4, we append a symbol 1 and obtain 1011000101 as one codeword.

Since $\phi(000000000) = 111110000$, which has weight 5, we append a symbol 0 and get 1111100000 as another codeword.

Both of the words 1011000101 and 1111100000 are balanced and the distance between them is 4.

Theorem 10: The code $\mathcal{C}$ obtained in Construction A is indeed a balanced $(n+1, 2\lceil d/2 \rceil)_2$-code of size at least $2^k/n$.

Proof: It is easy to see that $\mathcal{C}$ is a balanced code of length $n+1$ and size m. Since the m cyclic classes are pairwise disjoint and each of them consists of at most n codewords, we have that $m \ge |\mathcal{B}|/n = 2^k/n$.

Since $\mathcal{B}$ is an $[n, k, d]_2$-cyclic code and the map ϕ does not change the distance between any two vectors, the minimum distance of $\mathcal{C}$ is at least d. Moreover, when d is odd, the minimum distance is at least $d+1$, as the distance between any two binary balanced words is even. □

Let d be even and set $t = d/2 - 1$. If we apply Construction A to the family of primitive narrow-sense BCH $[n', k, d]_2$-cyclic codes in Theorem 1, where $n' = 2^m - 1 = n - 1$. we obtain a family of balanced codes with redundancy at most $(t+1)\log_2\ n + 1$.

Corollary 11: Let $d \ge 4$ be even. For any integer m such that $d \le 2^{\lceil m/2 \rceil} - 2$, there exists a family of $(n, d)_2$-balanced codes with redundancy at most $(t+1)\log_2\ n + 1$, where $n = 2^m$ and $t = d/2 - 1$.

In contrast, if we apply the technique of Weber *et al.* [10] to the same family of codes, the balanced $(n, d)_2$-codes have redundancy approximately $(t+1)\log_2\ n + (t + 1/2)\log_2 \log_2\ n$. Hence, we reduce the redundancy by $(t + 1/2)\log_2 \log_2\ n$ bits.

We also note that in some cases our codes can outperform the codes in [11]. For instance, let $n = 128$ and $t = 4$, according to Corollary 11 we have a balanced code of length 128 which can correct up to four errors and has redundancy 36, while in [11, Table VII] the corresponding code has redundancy $128 - 75 = 53$.

Finally, we consider the encoding complexity for our construction. Given a vector $\boldsymbol{u}$, we can find in linear time the index i such that $\mathrm{wt}(\phi(\boldsymbol{\sigma}^i(\boldsymbol{u}))) \in \{(n-1)/2, (n+1)/2\}$. Thus, it remains to provide an efficient method to enumerate a set of representatives for the cyclic classes. This problem was solved completely by Tavares *et al.* [9], [23] and the solution uses the polynomial representation of cyclic codewords. Furthermore, the encoding method can be adapted for Constructions E and F in the later sections. Hence, we review Tavares' method in detail and discuss our modifications in Section VI.

Before closing this subsection, we state our results for even n. The proof is similar to that of Theorem 10.

Lemma 12: Let n be even and $\phi : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be the map where $\phi(\boldsymbol{a}) = \boldsymbol{a} + 1^{n/2}0^{n/2}$. For $\boldsymbol{a} \in \mathbb{F}_2^n$, there are at least two different indices $i, j \in [\![n]\!]$ such that both $\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))$ and $\phi(\boldsymbol{\sigma}^j(\boldsymbol{a}))$ have the weight $n/2$ or $(n+2)/2$.

Construction A′: Let n be an even integer.

INPUT: An $[n, k, d]_2$-cyclic code $\mathcal{B}$.

OUTPUT: A balanced $(n+1, d)_2$-code $\mathcal{C}$ of size at least $2^k/n$.

- Let $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m$ be the set of representatives $\mathcal{B}/\underset{\text{cyc}}{\sim}$.
- For each $\boldsymbol{u}_i$, find $j_i \in [n]$ such that $\phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))$ has weight $n/2$ or $(n+2)/2$, where ϕ is defined in Lemma 12.
- For $i \in [m]$, append a bit 0 to $\phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))$ and denote the modified vector as $\boldsymbol{v}_i$.
- Set $\mathcal{C} = \{\boldsymbol{v}_i : 1 \le i \le m\}$.

B. GC-Balanced Error-Correcting Codes

A direct application of the coupling construction in Lemma 6 and Corollary 11 yields a family of GC-balanced $(n, d)_4$-codes with redundancy at most $d \log_4\ n$. However, this construction requires cyclic codes of length $n-1$.

The following construction removes the need for cyclic codes. The main idea here is that instead of using a balanced (n, d)-code as the input of the coupling construction, we may choose a systematic $(n+p, d)$-code whose information part is balanced, and use an (n, d)-code (not necessarily balanced) to protect the check part. The systematic code can be easily obtained by applying Knuth's balancing technique and a systematic encoding of a linear code.

Construction B:

INPUT: An $[n+p, n, d]_2$-linear code $\mathcal{A}$ and an $(n, d)_2$-code $\mathcal{B}$ of size $2^p nM$, where p is the number of check bits in each codeword of $\mathcal{A}$.

OUTPUT: A GC-balanced $(n, d)_4$-code $\mathcal{C}$ code of size $2^n M$.

- Given $\boldsymbol{m} \in \mathbb{F}_2^n$, let $j_{\boldsymbol{m}}$ be the balancing index of $\boldsymbol{m}$ and $\boldsymbol{a}_m$ be the balanced word obtained from $\boldsymbol{m}$ by flipping the first $j_{\boldsymbol{m}}$ bits.
- Consider a systematic encoder for $\mathcal{A}$. For $\boldsymbol{a}_{\boldsymbol{m}} \in \mathbb{F}_2^n$, let $\boldsymbol{a}_{\boldsymbol{m}}\boldsymbol{p}_{\boldsymbol{m}}$ be the corresponding codeword in $\mathcal{A}$. So $\boldsymbol{p}_{\boldsymbol{m}}$ is the check part of codeword $\boldsymbol{a}_{\boldsymbol{m}}\boldsymbol{p}_{\boldsymbol{m}}$.
- Finally, since $\mathcal{B}$ is of size $2^p nM$, we may assume without loss of generality an encoder $\boldsymbol{\phi}_{\mathcal{B}} : [M] \times [n] \times \mathbb{F}_2^p \to \mathcal{B}$. We set $\boldsymbol{b}_{\boldsymbol{m},i} = \boldsymbol{\phi}_{\mathcal{B}}(i, j_{\boldsymbol{m}}, \boldsymbol{p}_{\boldsymbol{m}})$.
- Set $\mathcal{C} \triangleq \{\Psi(\boldsymbol{a}_{\boldsymbol{m}}, \boldsymbol{b}_{\boldsymbol{m},i}) : \boldsymbol{m} \in \mathbb{F}_2^n, i \in [M]\}$.

Theorem 13: The code $\mathcal{C}$ obtained in Construction B is indeed a GC-balanced $(n, d)_4$-code of size $2^n M$.

Proof: The size of $\mathcal{C}$ follows from its definition.

For all words $\boldsymbol{c} = \Psi(\boldsymbol{a}, \boldsymbol{b})$ in $\mathcal{C}$, since $\boldsymbol{a}$ is balanced, we have that $\boldsymbol{c}$ is GC-balanced. Hence, $\mathcal{C}$ is GC-balanced.

Finally, to prove that $\mathcal{C}$ has distance d, we show that $\mathcal{C}$ can always correct $t = \lfloor (d-1)/2 \rfloor$ errors. Specifically, let $\boldsymbol{c} \in \mathcal{C}$ and let $\hat{\boldsymbol{c}}$ be a word over Σ such that $d(\boldsymbol{c}, \hat{\boldsymbol{c}}) \le t$. Suppose that $\boldsymbol{c} = \Psi(\boldsymbol{a}, \boldsymbol{b})$ and $\hat{\boldsymbol{c}} = \Psi(\hat{a}, \hat{b})$. Then $d(\boldsymbol{a}, \hat{\boldsymbol{a}}) \le t$

and $d(\boldsymbol{b}, \hat{\boldsymbol{b}}) \leq t$. Since $\boldsymbol{b}$ belongs to $\mathcal{B}$, which is an $(n, d)_2$-code, we are able to correct the errors in $\hat{\boldsymbol{b}}$ to recover $\boldsymbol{b}$.

Suppose that $\boldsymbol{b} = \phi(i, j, \boldsymbol{p})$. Then we have that $\boldsymbol{ap}$ is a codeword in $\mathcal{A}$. Since $\mathcal{A}$ is an $[n+p, n, d]_2$-code, we can correct the errors in $\hat{\boldsymbol{ap}}$ to recover $\boldsymbol{ap}$ and hence, recover $\boldsymbol{a}$. Therefore, $\mathcal{C}$ is an $(n, d)_4$-code. □

Corollary 14: Fix d and set $t = \lceil (d-1)/2 \rceil$. There exists a GC-balanced $(n, d)_4$-code with redundancy at most $(2t+1)\lceil \log_4 \ n \rceil + t$.

Proof: For sufficiently large n, we shorten appropriate BCH codes given in Theorem 1 to obtain an $[n+p, n, d]_2$-linear code and an $[n, k, d]_2$-linear code so that $p \leq t\lceil \log_2 \ n \rceil + t$ and $n - k \leq t\lceil \log_2 \ n \rceil + t$. Then applying Construction B, we obtain a GC-balanced code of size $2^n M$, where $\log_2 M \geq n - 2(t\lceil \log_2 \ n \rceil + t) - \log_2 \ n$. Hence, the redundancy is at most

$$n - \log_4 2^n M = n/2 - (\log_2 \ M)/2 \leq (2t+1)\lceil \log_4 \ n \rceil + t.$$

□

IV. Primer Codes

In this section, we provide three constructions of primer codes: one direct modification of a construction of Yazdi *et al.* that yields primer codes for general parameters and the other two that rely on cyclic codes and have lower redundancy for a specific set of parameters. Recall that an $(n, d; \kappa, f)_q$-primer code satisfies the following three conditions:

(P1) $\mathcal{C}$ is an $(n, d)_q$-code;

(P2) $\mathcal{C}$ is κ-WMU, i.e., for any two codewords $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$, not necessarily distinct, and $\kappa \leq \ell < n$,

$$\boldsymbol{a}[1, \ell] \neq \boldsymbol{b}[n - \ell + 1, n];$$

(P3) $\mathcal{C}$ is an f-APD code, i.e., for any two codewords $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$, not necessarily distinct, and $1 \leq i, j \leq n + 1 - f$, we have

$$\overline{\boldsymbol{a}}[i, i+f-1] \notin \{\boldsymbol{b}[j, j+f-1], \boldsymbol{b}[j+f-1, j]\}.$$

Furthermore, if it is balanced or GC-balanced, then it is an $(n, d; \kappa, f)_q$-balanced primer code.

A. $(n, d; \kappa, f)_2$-Primer Codes and $(n, d; \kappa, f)_4$-Balanced Primer Codes

Our first construction modifies the work of Yazdi *et al.* to yield a class of $(n, d; \kappa, f)_2$-primer codes. Then applying the coupling construction (Lemma 6), and taking these primer codes together with the balanced error-correcting codes in Section III as inputs, we are able to obtain a class of $(n, d; \kappa, f)_4$-balanced primer codes.

Yazdi *et al.* [6] constructed a set of mutually uncorrelated primers that avoids primer dimer byproducts.

Definition 15: A binary code $\mathcal{A} \subseteq \mathbb{F}_2^n$ is *ℓ-APD-constrained* if for each $\boldsymbol{a} \in \mathcal{A}$,

- $\boldsymbol{a}$ ends with one,
- $\boldsymbol{a}$ contains $01^\ell 0$ as a substring exactly once,
- $\boldsymbol{a}$ does not contain 0^ℓ as a substring.

Lemma 16 (Yazdi et al. *[6, Lemma 5]):* Let n, f, ℓ, r be positive integers such that $n = rf + \ell + 1$ and $\ell + 3 \leq f$. Suppose that $\mathcal{A}$ is an ℓ-APD-constrained code of length f. Then the code

$$\mathtt{C} = \{0^\ell 1 \boldsymbol{a}_1 \boldsymbol{a}_2 \ldots \boldsymbol{a}_r : \boldsymbol{a}_i \in \mathcal{A} \text{ for each } i \in [r]\}$$

is both MU (i.e., 1-WMU) and $2f$-APD, and its size is $|\mathcal{A}|^r$.

The following construction equips the code in Lemma 16 with error-correcting capabilities.

Construction C: Let f, r, d, p and ℓ be positive integers, where $\ell + 3 \leq f$ and $p + \lfloor p/(\ell-1) \rfloor + 1 \leq f$.

INPUT: An $[rf + p, rf, d]_2$-linear code $\mathcal{B}$ and a binary ℓ-APD-constrained code $\mathcal{A}$ of length f.

OUTPUT: A binary $(n, d; 1, 2f)$-primer code $\mathcal{C}$ of length $n = rf + p + \lfloor p/(\ell-1) \rfloor + \ell + 2$ and size $|\mathcal{A}|^r$.

- Consider a systematic encoder for $\mathcal{B}$.
- For every message $\boldsymbol{a} \in \mathbb{F}_2^{rf}$, let $\boldsymbol{ap_a}$ be the corresponding codeword in $\mathcal{B}$.
- For the vector $\boldsymbol{p_a}$, we insert a one after every $(\ell-1)$ bits and append a one. In other words, we insert $\lfloor p/(\ell-1) \rfloor + 1$ ones and we call the resulting vector $\boldsymbol{p'_a}$.
- Set $\mathcal{C} \triangleq \{0^\ell 1 \boldsymbol{ap'_a} : \boldsymbol{a} \in \mathcal{A}^r\}$.

It is easy to see that the code $\mathcal{C}$ obtained in Construction C has minimum distance at least d. The proof that shows $\mathcal{C}$ is MU and $2f$-APD is the same as that for Lemma 16 and we omit here.

Next, for fixed values of r and d, we use Construction C to obtain a family of $(n, d; 1, f)_2$-primer codes with $n = rf + o(f)$ and redundancy at most $t \log_2 \ n + O(1)$, where $t = \lceil (d-1)/2 \rceil$. Specifically, we provide the constructions for the input codes $\mathcal{A}$ and $\mathcal{B}$ in Construction C.

Lemma 17: For $\ell \geq 8$, set $f = 2^{\ell-4}$. Then there exists an ℓ-APD-constrained code $\mathcal{A}$ of size $(f - \ell - 2)2^{f-\ell-4}$. Furthermore, there is a linear-time encoding algorithm that maps $[f - \ell - 2] \times \mathbb{F}_2^{f-\ell-4}$ to $\mathcal{A}$.

Proof: We first construct a code $\mathcal{A}_0$ of length $f - \ell - 3$, where all codewords do not contain either $0^{\ell-1}$ or $1^{\ell-1}$ as substrings. Then $\mathcal{A}$ can be constructed by inserting $01^\ell 0$ to the codewords in $\mathcal{A}_0$ and appending a symbol 1. Since there are $f - \ell - 2$ possible positions to insert $01^\ell 0$, we have $|\mathcal{A}| = (f - \ell - 2)|\mathcal{A}_0|$.

To construct the code $\mathcal{A}_0$, we use the encoding algorithm ϕ proposed by Schoeny *et al.* [24] that maps a binary sequence of length $(f - \ell - 4)$ to a binary sequence of length $(f - \ell - 3)$ that avoids $0^{\ell-1}$ and $1^{\ell-1}$ as substrings. Furthermore, the encoding map ϕ has running time $O(f)$. □

Remark 18: Due to our construction of $\mathcal{A}$ (Lemma 17), the runs of any symbols in our proposed primer codes have lengths at most 2ℓ.

Hence, for $\ell \geq 8$, we choose $f = 2^{\ell-4}$. For the input code $\mathcal{B}$, we shorten an appropriate BCH code given in Theorem 1 to obtain an $[rf + p, rf, d]$-linear code with redundancy $p \leq t \log_2 \ n + t$. Hence, applying Construction C, we obtain an $(n, d; 1, f)_2$-primer codes with $n = rf + p + \lfloor p/(\ell-1) \rfloor + \ell + 2$.

We assume r is a constant with respect to ℓ. Observe that for sufficiently large ℓ, we have that $rf < n < (r+1)f$. By choice of ℓ, we have that $\log_2 n + C_1 \leq \ell \leq \log_2 n + C_2$, for some constants C_1, C_2 dependent only on r.

To analyse the redundancy of the construction, we have that

$$\begin{aligned}\log_2 |\mathcal{A}|^r &= r(f-\ell-4)+r\log_2(f-\ell-2)\\ &\geq r(f-\ell-4)+r\log_2(f/2)\\ &= r(f-\ell-4)+r(\ell-5)=rf-9r.\end{aligned}$$

Therefore, the redundancy is given by $n-\log_2|\mathcal{A}|^r$, which is at most

$$\begin{aligned}&p+\lfloor p/(\ell-1)\rfloor+\ell+2+9r\\ &\leq (t\log_2 n+t)+\frac{t\log_2\ n+t}{\log_2\ n+C_1-1}+(\log_2\ n+C_2)+2+9r\\ &= (t+1)\log_2\ n+O(1).\end{aligned}$$

In summary, we have the following theorem.

Theorem 19: Fix r and d. Then there exists a family of $(n,d;1,f)_2$-primer codes with $n=rf+o(f)$ and redundancy at most $(t+1)\log_2\ n+O(1)$, where $t=\lceil(d-1)/2\rceil$. Furthermore, there exists a linear-time encoding algorithm for these primer codes.

Applying Lemma 6, we obtain primer codes over $\{\mathtt{A},\mathtt{T},\mathtt{C},\mathtt{G}\}$.

Corollary 20: Fix r and d, and set $t=\lceil(d-1)/2\rceil$.

(i) There exists a family of $(n,d;1,f)_4$-primer codes with $n=rf+o(f)$ and redundancy at most $(2t+1)\log_4 n+O(1)$.

(ii) There exists a family of $(n,d;1,f)_4$-balanced primer codes with $n=rf+o(f)$ and redundancy at most $(d+1)\log_4\ n+O(1)$.

Recall that the iterative construction of balanced primer codes proposed by Yazdi *et al.* requires short balanced primer codes with a collection of disjoint subcodes [6, Construction 11]. Our construction is more explicit and Corollary 20 provides the first infinite family of balanced primer codes with efficient encoding.

B. Almost GC-Balanced $(n,d)_4$-Codes That Are κ-Mutually Uncorrelated

Yazdi *et al.* proposed two constructions of balanced $(n,d)_q$-codes which are κ-WMU [6, Constructions 8 and 9]. One of them requires at least $n/2$ redundant symbols. The other is based on the coupling construction, and requires a binary balanced error-correcting code as well as a binary κ-WMU error-correcting code as inputs. In the same paper, Yazdi *et al.* also derived a lower bound on the size of a binary κ-WMU error-correcting code, without giving any explicit construction. We note that Construction C could be modified to produce κ-WMU error-correcting codes. However, the redundancy of the output codes would become large when d grows with n.

In this subsection, we modify Construction A and use cyclic codes to obtain *almost balanced* $(n,d)_q$-codes that are κ-WMU, increasing the redundancy only by $\log_q n$. Here, a code is *almost balanced* if the weight (or GC-content) of every word belongs to $\{\lfloor n/2\rfloor-1,\lfloor n/2\rfloor,\lceil n/2\rceil,\lceil n/2\rceil+1\}$.

Let n be odd and we abuse notation by using ϕ to also denote the map $\phi:\mathbb{F}_4^n\to\mathbb{F}_4^n$, where $\phi(\boldsymbol{a})=\boldsymbol{a}+\omega^{(n+1)/2}0^{(n-1/2)}$. In other words, ϕ switches A with C and T with G, and vice versa, in the first $(n+1)/2$ coordinates of $\boldsymbol{a}$. We have the following analogue of Lemma 7.

Lemma 21: For $\boldsymbol{a}\in\mathbb{F}_4^n$, we can find $i\in[\![n]\!]$ such that $\phi(\boldsymbol{\sigma}^i(\boldsymbol{a}))$ is GC-balanced.

Construction D: Let n be odd, $k\leq\lceil(n+1)/4\rceil$ and $q\in\{2,4\}$

INPUT: An $[n,k,d]_q$-cyclic code $\mathcal{B}$ containing 1^n.

OUTPUT: An almost balanced $(n,d)_q$-code $\mathcal{C}$ which is $(k+1)$-WMU and has size at least q^k/n.

- Let $\boldsymbol{u}_1,\boldsymbol{u}_2,\ldots,\boldsymbol{u}_m$ be the set of representatives $\mathcal{B}/\underset{\text{cyc}}{\sim}$.
- For each $\boldsymbol{u}_i$, find $j_i\in[n]$ such that $\phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))$ is either balanced or GC-balanced.
- Let $\mu=(n-1)/2$. For each $\boldsymbol{u}_i$, set

$$\boldsymbol{v}_i=\begin{cases}\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i)+1^{\mu+1}0^{\mu-1}1, & \text{if } q=2,\\ \boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i)+\omega^{\mu+1}0^{\mu-1}\omega, & \text{if } q=4.\end{cases}$$

- Set $\mathcal{C}=\{\boldsymbol{v}_i:1\leq i\leq m\}$.

Example 22: Let $g(X)=X^{11}+x^{10}+\omega X^8+\omega^2\ X^7+\omega^2\ X^6+X^5+\omega^2\ X^4+X^3+X+\omega^2$. Then $g(X)$ is the generator polynomial of an optimal $[15,4,10]_4$-cyclic code $\mathcal{B}$. Consider two codewords

$$\boldsymbol{u}_1=(0,0,\omega,\omega,\omega,1+\omega,0,\omega,0,1+\omega,0,1+\omega,1+\omega,1,\omega)$$

and

$$\boldsymbol{u}_2=(1+\omega,0,1+\omega,1,\omega,\omega,1+\omega,\omega,1,0,0,\omega,0,1,1+\omega)$$

from $\mathcal{B}$. For $\boldsymbol{u}_1$, the shift

$$\boldsymbol{\sigma}^1(\boldsymbol{u}_1)=(\omega,0,0,\omega,\omega,\omega,1{+}\omega,0,\omega,0,1{+}\omega,0,1{+}\omega,1{+}\omega,1)$$

can be balanced by flipping the first 8 symbols, and so we let

$$\begin{aligned}\boldsymbol{v}_1&=\boldsymbol{\sigma}^1(\boldsymbol{u}_1)+\omega^8 0^6\omega\\ &=(0,\omega,\omega,0,0,0,1,\omega,\omega,0,1+\omega,0,1+\omega,1+\omega,1+\omega).\end{aligned}$$

Similarly, for $\boldsymbol{u}_2$, we let

$$\begin{aligned}\boldsymbol{v}_2&=\boldsymbol{\sigma}^2(\boldsymbol{u}_2)+\omega^8 0^6\omega\\ &=(1+\omega,1,1,\omega,1,1+\omega,0,0,1+\omega,\omega,1,0,0,\omega,\omega).\end{aligned}$$

We can see that $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are GC-balanced, and they have distance 14. Furthermore, they are 4-WMU. By considering the following codewords from $\mathcal{B}$, we can obtain an almost balanced $(15,10)_4$-code $\mathcal{C}$ which is 5-WMU and has 17 codewords.

$$\begin{aligned}\boldsymbol{u}_1&=(0,0,\omega,\omega,\omega,1+\omega,0,\omega,0,1+\omega,0,1+\omega,1+\omega,1,\omega)\\ \boldsymbol{u}_2&=(1+\omega,0,1+\omega,1,\omega,\omega,1+\omega,\omega,1,0,0,\omega,0,1,1+\omega)\\ \boldsymbol{u}_3&=(\omega,\omega,0,0,0,1,\omega,0,\omega,1,\omega,1,1,1+\omega,0)\\ \boldsymbol{u}_4&=(1,0,0,1+\omega,\omega,1,1,\omega,\omega,\omega,0,1,\omega,1,0)\\ \boldsymbol{u}_5&=(1+\omega,\omega,\omega,1,0,1+\omega,1+\omega,0,0,0,\omega,1+\omega,0,1+\omega,\omega)\\ \boldsymbol{u}_6&=(1+\omega,1,0,1,1+\omega,1,1+\omega,1+\omega,\omega,0,1,1,0,0,0)\\ \boldsymbol{u}_7&=(\omega,1,\omega,0,1+\omega,1+\omega,\omega,1+\omega,0,1,1,1+\omega,1,0,\omega)\\ \boldsymbol{u}_8&=(1,1+\omega,\omega,1+\omega,1,1+\omega,1,1,0,\omega,1+\omega,1+\omega,\omega,\omega,\omega)\\ \boldsymbol{u}_9&=(1,\omega,1,1+\omega,0,0,1,0,1+\omega,\omega,\omega,0,\omega,1+\omega,1)\\ \boldsymbol{u}_{10}&=(0,1+\omega,0,\omega,1,1,0,1,\omega,1+\omega,1+\omega,1,1+\omega,\omega,0)\\ \boldsymbol{u}_{11}&=(0,\omega,1+\omega,\omega,0,\omega,0,0,1,1+\omega,\omega,\omega,1+\omega,1+\omega,1+\omega)\\ \boldsymbol{u}_{12}&=(\omega,1+\omega,1+\omega,0,1,\omega,\omega,1,1,1,1+\omega,\omega,1,\omega,1+\omega)\\ \boldsymbol{u}_{13}&=(1,1,1+\omega,1+\omega,1+\omega,\omega,1,1+\omega,1,\omega,1,\omega,\omega,0,1+\omega)\end{aligned}$$

$\boldsymbol{u}_{14} = (1+\omega, 1+\omega, 1, 1, 1, 0, 1+\omega, 1, 1+\omega, 0, 1+\omega, 0, 0, \omega, 1)$
$\boldsymbol{u}_{15} = (0, 1, 1, \omega, 1+\omega, 0, 0, 1+\omega, 1+\omega, 1+\omega, 1, 0, 1+\omega, 0, 1)$
$\boldsymbol{u}_{16} = (\omega, 0, 1, 0, \omega, 0, \omega, \omega, 1+\omega, 1, 0, 0, 1, 1, 1)$
$\boldsymbol{u}_{17} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

Theorem 23: The code $\mathcal{C}$ obtained in Construction D is an almost balanced $(n, d)_q$-code which is $(k+1)$-WMU.

To prove Theorem 23, we require the following technical lemma modified from Yazdi *et al.* [6].

Lemma 24: Let $\mathcal{B}$ be a cyclic code of dimension k containing 1^n. Then the run of any symbols in any non-constant codeword is at most $k-1$.

Proof of Theorem 23: Since $\mathcal{C}$ is a subset of a coset of $\mathcal{B}$, we have that $\mathcal{C}$ is an $(n, d)_q$-code. For $i \in [m]$, since $\phi(\boldsymbol{\sigma}^{j_i}(\boldsymbol{u}_i))$ is balanced and $\boldsymbol{v}_i$ differs from the former in one symbol, we have that $\boldsymbol{v}_i$ is almost balanced.

Now, we demonstrate weakly mutually uncorrelatedness for the case of $q = 2$. The case of $q = 4$ can be proceeded in the same way. Suppose on the contrary that $\mathcal{C}$ is not $(k+1)$-WMU. Then there is a proper prefix $\boldsymbol{p}$ of length ℓ, $\ell \geq k+1$ such that both $\boldsymbol{pa}$ and $\boldsymbol{bp}$ belong to $\mathcal{C}$. It follows that $\mathcal{B}$ contains the words

$$\boldsymbol{pa} + 1^{\mu+1}0^{\mu-1}1 \text{ and } \boldsymbol{bp} + 1^{\mu+1}0^{\mu-1}1,$$

where $\mu = (n-1)/2$. Consequently, since $\mathcal{B}$ is cyclic, we have that $\boldsymbol{pb} + \boldsymbol{\sigma}^{\ell}(1^{\mu+1}0^{\mu-1}1)$ belongs to $\mathcal{B}$. Hence, by linearity of $\mathcal{B}$, the word

$$\boldsymbol{c} \triangleq 0^{\ell}(\boldsymbol{a} - \boldsymbol{b}) + 1^{\mu+1}0^{\mu-1}1 + \boldsymbol{\sigma}^{\ell}(1^{\mu+1}0^{\mu-1}1)$$

belongs to $\mathcal{B}$. We look at the prefix of length ℓ of $\boldsymbol{c}$.

- When $\ell \leq \mu$, the word $\boldsymbol{c}$ has prefix $1^{\ell-1}0$. Hence, $\boldsymbol{c}$ is a non-constant codeword of $\mathcal{C}$ and since $\ell - 1 \geq k$, this contradicts Lemma 24.
- When $\ell = \mu + 1$, the word $\boldsymbol{c}$ has prefix $01^{\mu-1}$. Hence, $\boldsymbol{c}$ is a non-constant codeword of $\mathcal{C}$ and since $\mu - 1 \geq k$, this contradicts Lemma 24.
- When $\ell \geq \mu + 2$, the word $\boldsymbol{c}$ has prefix $0^{\ell-\mu}1^{2\mu+1-\ell}$. Since either $\ell - \mu$ or $2\mu + 1 - \ell$ is at least $\lceil \mu + 1/2 \rceil = \lceil (n+1)/4 \rceil \geq k$, the word $\boldsymbol{c}$ contains a run of ones or zeros of length k, contradicting Lemma 24.

C. $(n, d; \kappa, f)_4$-Primer Codes With $\kappa = f$

Using reversible cyclic codes, we further reduce the redundancy for primer codes in the case when $\kappa = f$.

Definition 25: Let $g(X)$ be the generator polynomial of a reversible cyclic code $\mathcal{B}$ of length n and dimension k that contains 1^n. Set $h(X) = (X^n - 1)/g(X)$. The set $\{h^*(X), p_1(X), p_2(X), \ldots, p_P(X)\}$ of polynomials is (g, k)-*rc-generating* if the following hold:

(R1) $h^*(X)$ divides $h(X)$;
(R2) $h^*(1) \neq 0$;
(R3) $h^*(X) = X^{d^*}h^*(X^{-1})/h^*(0)$, where $d^* = \deg h^*$;
(R4) $h^*(X)$ does not divide $X^s p_i(X) - p_j(X)$ for all $i, j \in [P]$ and $s \in [n-1]$.
(R5) $h^*(X)$ does not divide $X^s p_i(X) - X^{k-1}p_j(X^{-1})$ for all $i, j \in [P]$ and $0 \leq s \leq n-k$.
(R6) $h^*(X)$ does not divide $X^{s+k-1}p_i(X^{-1}) - p_j(X)$ for all $i, j \in [P]$ and $0 \leq s \leq n-k$.
(R7) $\deg p_i(X) < \deg h^*$ for $i \in [P]$.

Construction E:

INPUT: An $[n, k, d]_q$-reversible cyclic code $\mathcal{B}$ containing 1^n with generator polynomial $g(X)$ and a (g, k)-rc-generating set of polynomials $\{h^*(X), p_1(X), p_2(X), \ldots, p_P(X)\}$.
OUTPUT: An $(n, d; k, k)_q$-primer code $\mathcal{C}$ of size $q^{k^*}P$, where $k^* = k - \deg h^*$.

- Set

$$\mathcal{C} \triangleq \{(m(X)h^*(X) + p_i(X))g(X) : \deg m < k^*, i \in [P]\}.$$

We note that the code $\mathcal{C}$ is a subcode of the $[n, k, d]_q$-reversible cyclic code $\mathcal{B}$, and so the runs in each codeword have lengths at most $k-1$.

Using conditions in Definition 25, we are able to show that the code obtained in Construction E is indeed a primer code. We defer the detailed verification to Section VI as it involves more notions and techniques. The following example illustrates that the conditions in Definition 25 are crucial to our construction.

Example 26: Set $n = 15$ and $q = 4$. Let $g(x) = x^6 + x^5 + (\omega+1)x^4 + x^3 + (\omega+1)x^2 + x + 1$ be the generator polynomial of a $[15, 9, 5]_4$-reversible cyclic code that contains 1^n. Consider $h^*(X) = X^4 + \omega X^3 + \omega X^2 + \omega X + 1$, $p_1(X) = 1$ and $p_2(X) = X^3$.

We have that $X^{12}p_2(X) - p_1(X) = X^{15} - 1$, which is 0 in $\mathbb{F}_q[X]/\langle X^n - 1\rangle$. So, $h^*(X)$ divides $X^{13}p_2(X) - p_1(X)$. The set $\{h^*(X), p_1(X), p_{12}(X)\}$ does not satisfy condition (R4) of Definition 25.

Now, we look at the code $\mathcal{C} = \{(m(X)h^*(X) + p_i(X))g(X) : \deg m < k^*, i \in [2]\}$. Both $p_1(X)g(X)$ and $p_2(X)g(X)$ are in this code. However, the corresponding words are

$$(\mathbf{1}, \mathbf{1}, \omega+\mathbf{1}, \mathbf{1}, \omega+\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, 0, 0, 0)$$

and

$$(0, 0, 0, \mathbf{1}, \mathbf{1}, \omega+\mathbf{1}, \mathbf{1}, \omega+\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}).$$

The prefix of length 12 of $p_1(X)g(X)$ appears as a suffix of $p_2(X)g(X)$. Then $\mathcal{C}$ is not 9-WMU.

By careful design of the (g, k)-rc-generating set, we could construct primer codes of large size.

Example 27: Set $n = 15$ and $q = 4$. Let $g(x) = x^6 + x^5 + (\omega+1)x^4 + x^3 + (\omega+1)x^2 + x + 1$ be the generator polynomial of a $[15, 9, 5]_4$-reversible cyclic code that contains 1^n. Let $h^*(X) = X^4 + \omega X^3 + \omega X^2 + \omega X + 1$ and

$$\begin{aligned}
p_1(X) &= 1,\\
p_2(X) &= \omega,\\
p_3(X) &= \omega + 1,\\
p_4(X) &= \omega X + \omega,\\
p_5(X) &= (\omega+1)X + \omega + 1,\\
p_6(X) &= X + 1,\\
p_7(X) &= \omega X^2 + \omega X + \omega + 1,
\end{aligned}$$

$$
\begin{aligned}
p_8(X) &= \omega X^3 + (\omega+1)\ X^2 + 1,\\
p_9(X) &= \omega X^3 + (\omega+1)X^2 + X,\\
p_{10}(X) &= \omega X^3 + (\omega+1)X^2 + X + \omega + 1,\\
p_{11}(X) &= \omega X^3 + (\omega+1)X^2 + X + 1,\\
p_{12}(X) &= \omega X^3 + X^2,\\
p_{13}(X) &= \omega X^3 + X^2 + X + 1,\\
p_{14}(X) &= (\omega+1)\ X^3 + \omega X^2 + \omega X + \omega,\\
p_{15}(X) &= (\omega+1)X^3 + \omega X^2 + (\omega+1)X + \omega + 1,\\
p_{16}(X) &= (\omega+1)X^3 + X^2 + \omega X + 1,\\
p_{17}(X) &= X^3 + \omega X^2 + (\omega+1)X + \omega + 1.
\end{aligned}
$$

We can verify that the set $\{h^*(X), p_1(X), \ldots, p_{17}(X)\}$ is $(g, 9)$-rc-generating. Therefore, $k^* = 9 - 4 = 5$ and the size of the $(15, 5; 9, 9)_4$-primer code have size $17(4^5) \geq 2^{14}$.

In contrast, for their experiment, Yazdi *et al.* constructed a set of weakly mutually uncorrelated primers of length 16, distance four and size four. Specifically, they set $\mathcal{C}_1 = \{01^701^7, 10^710^7\}$ and $\mathcal{C}_2$ to be an extended BCH $[16, 11, 4]$-cyclic code. Then they applied the coupling construction to obtain an $(16, 4; 9, 16)$-primer code of size 2^{12}.

Therefore, Construction E provides a larger set of primers using less bases, while improving the minimum distance and avoiding primer dimer products at the same time.

In the following, we describe a way to find the (g, k)-rc-generating set. To do so, we recall some concepts in finite field theory.

For $m \geq 2$, we consider the finite field $F \triangleq \mathbb{F}_{4^m}$. A nonzero element $\alpha \in F$ is said to be *primitive* if $\alpha^i \neq 1$ for $i \in [4^m - 2]$. For $\alpha \in F$, we let $M_\alpha(X)$ denote the *minimal polynomial* of α in the base field $\mathbb{F}_4$.

Given a polynomial $p(X) = \sum_{i=0}^{n-1} c_i X^i \in \mathbb{F}_q[X]/\langle X^n - 1\rangle$, we define the *reciprocal polynomial* of $p(X)$ to be $p^\dagger(X) = X^{\deg p} p(X^{-1})$ and we say $p(X)$ is *self-reciprocal* if $p(0) \neq 0$ and $p(X) = p^\dagger(X)/p(0)$.

Then the following facts are useful in establishing our results.

Lemma 28: Let F be a field with 4^m elements.

(a) For nonzero $\alpha \in F$, the polynomial $M_\alpha(X)\ M_{\alpha^{-1}}(X)$ is self-reciprocal.
(b) If $\alpha \in F$ is primitive, then $M_\alpha(X)$ does not divide $X^s - 1$ for $s \in [4^m - 2]$.
(c) There are $\varphi(4^m - 1)$ primitive elements in F.

The following lemma provides a set of polynomials that satisfies Definition 25.

Lemma 29: Let $g(X)$ be the generator polynomial of the $[n, k, d]_q$-reversible cyclic code $\mathcal{B}$ from Theorem 2, where $n = q^m - 1$. Let α be a primitive element of F such that $g(\alpha) \neq 0$ and $g(\alpha^{-1}) \neq 0$. Denote $h^*(X) = M_\alpha(X) M_{\alpha^{-1}}(X)$. If $n - k < k - 1$, then the set $\{h^*(X), 1\}$ is (g, k)-rc-generating.

Proof: We verify conditions (R1) to (R7) in Definition 25.

(R1) follows from the fact that both α and α^{-1} are not roots of g. Since h^* is the product of two minimal polynomials of primitive elements, $h^*(1)$ is not zero and so (R2) holds. (R3) follows from Lemma 28(a).

Next, observe that $P = 1$ with $p_1(X) = 1$. Hence, (R7) trivially holds. Also, (R4) to (R6) reduces to verifying that

(i) $h^*(X)$ does not divide $X^s - 1$ for $s \in [n - 1]$; and
(ii) $h^*(X)$ does not divide $X^s - X^{k-1}$ for $0 \leq s \leq n - k$.

Since $n - k < k - 1$, we have that $X^s - X^{k-1}$ is nonzero for $0 \leq s \leq n - k$. Then both (i) and (ii) follows from Lemma 28(b). □

Applying Construction E to the reversible cyclic codes in Theorem 2 and the (g, k)-rc-generating above, we have the following result.

Corollary 30: Let $m \geq 6$ and $1 \leq \tau \leq \lceil m/2 \rceil$. Set $n = 4^m - 1$ and $d = 4^\tau - 1$. There exists an $(n, d; k, k)_4$-primer code of size 4^{k-2m}, where

$$
k = \begin{cases} n - (d-3)m, & \text{if } m \text{ is odd and } \tau = \frac{m+1}{2};\\ n - (d-1)m, & \text{otherwise.}\end{cases}
$$

Therefore, there is a family of $(n, d; k, k)_4$-primer codes with $d \approx \sqrt{n}$, $k \approx n - \sqrt{n}\log_4 n$, and redundancy at most $(d+1)\log_4(n+1)$.

Proof: Let $g(X)$ be the generator polynomial of the reversible cyclic code $\mathcal{C}$ constructed in Theorem 2.

Consider the set $\Lambda = \{\alpha \in F : g(\alpha) = 0 \text{ or } g(\alpha^{-1}) = 0\}$. Since the degree of g is $n - k \leq (d-1)m$, we have that $|\Lambda| \leq 2(d-1)m$. Since $\varphi(4^m - 1) > 2(d-1)m$ for $m \geq 6$, there exists a primitive element $\alpha \in F$ that does not belong to Λ. In other words, $g(\alpha) \neq 0$ and $g(\alpha^{-1}) \neq 0$. By Lemma 29, the set $\{h^*(X) \triangleq M_\alpha(X) M_{\alpha^{-1}}(X), 1\}$ is (g, k)-rc-generating and therefore, Construction E yields an $(n, d; k, k)$-primer code of size 4^{k-2m}. □

V. Codes for DNA Computing

Since Adleman demonstrated the use of DNA hybridization to solve a specific instance of the directed Hamiltonian path problem [25], the coding community have investigated the possibility of error control via code design [26], [27].

In this paper, we focus on designing codes with the following constraints.

Definition 31: A *balanced* (n, d)*-DNA computing code* is a GC-balanced $(n, d)_4$-code such that

(C1) $d(\boldsymbol{a}, \boldsymbol{b}^r) \geq d$ for all $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$.
(C2) $d(\boldsymbol{a}, \boldsymbol{b}^{rc}) \geq d$ for all $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$.

As always, the fundamental problem for DNA computing codes is to find the largest possible codes satisfying the constraints above. Many approaches have been considered for this problem. These include search algorithms, template-based constructions and constructions over certain algebraic rings (see, Limbachiya *et al.* [21] for a survey).

There are few explicit families of balanced DNA computing codes for large n. In particular, ElDin and Matsui [12] proposed an algorithm, based on weight enumerators, to obtain balanced DNA-computing codes of length $n > 1000$ with minimum distance nearly $n/2$. In this section we propose a class of balanced DNA computing codes that satisfies both the constraints (C1) and (C2). Compared with the work of ElDin and Matsui, our codes have distance nearly $\sqrt{n}$ with much larger size.

We modify our balancing techniques in Sections III and IV. Recall that by flipping, we mean exchanging A with C and T with G. Then Lemma 21 states that for $\boldsymbol{a} \in \mathbb{F}_4^n$, we can balance one of its cyclic shifts by flipping its first $\lceil n/2 \rceil$ components. However, in order to accommodate the reverse and reverse-complement distance constraints, we do the following.

Let n be odd and set s be the integer nearest to $n/4$. In other words, s is the unique integer in the set $\{(n-1)/4, (n+1)/4\}$. Let $\pi : \mathbb{F}_4^n \to \mathbb{F}_4^n$ be the map such that $\pi(\boldsymbol{a}) = \boldsymbol{a} + \omega^s 0^{n-2s} \omega^s$ for any $\boldsymbol{a} \in \mathbb{F}_4^n$. In other words, π flips the first s and the last s symbols of $\boldsymbol{a}$. The following lemma follows directly from Lemma 21.

Lemma 32: Let n be odd. For any $\boldsymbol{a} \in \mathbb{F}_4^n$, there exists $i \in [\![n]\!]$ such that $\pi(\boldsymbol{\sigma}^i(\boldsymbol{a}))$ is GC-balanced.

As before, we next define a set of polynomials that enables us to generate our code efficiently.

Definition 33: Let $g(X)$ be the generator polynomial of a reversible cyclic code $\mathcal{B}$ of length n and dimension k that contains 1^n. Set $h(X) = (X^n - 1)/g(X)$. The set $\{h^*(X), p_1(X), p_2(X), \ldots, p_P(X)\}$ of polynomials is (g, k)-*rc2-generating* if the set obeys conditions (R1) to (R4), (R7) in Definition 25, i.e.,

(R1) $h^*(X)$ divides $h(X)$;
(R2) $h^*(1) \neq 0$;
(R3) $h^*(X) = X^{d^*} h^*(X^{-1})/h^*(0)$, where $d^* = \deg h^*$;
(R4) $h^*(X)$ does not divide $X^s p_i(X) - p_j(X)$ for all $i, j \in [P]$ and $s \in [n-1]$.
(R7) $\deg p_i(X) < \deg h^*$ for $i \in [P]$;

and

(R5') $h^*(X)$ does not divide $X^s p_i(X) - X^{k-1} p_j(X^{-1})$ for all $i, j \in [P]$ and $s \in [\![n]\!]$.

Construction F: Let n be odd.

INPUT: An $[n, k, d]_4$-reversible cyclic code $\mathcal{B}$ containing 1^n with generator polynomial $g(X)$ and a (g, k)-rc2-generating set of polynomials $\{h^*(X), p_1(X), p_2(X), \ldots, p_P(X)\}$.
OUTPUT: A balanced $(n, d)_4$-DNA computing code $\mathcal{C}$ of size $4^{k^*} P$, where $k^* = k - \deg h^*$.

- Set

$$\mathcal{A} \triangleq \{(m(X)h^*(X) + p_i(X))g(X) : \deg m < k^*, i \in [P]\}.$$

- For $\boldsymbol{u} \in \mathcal{A}$, find $i_{\boldsymbol{u}} \in [\![n-1]\!]$ such that $\boldsymbol{v}_{\boldsymbol{u}} = \pi(\boldsymbol{\sigma}^{i_{\boldsymbol{u}}}(\boldsymbol{u}))$ is GC-balanced.
- Set $\mathcal{C} = \{\boldsymbol{v}_{\boldsymbol{u}} : \boldsymbol{u} \in \mathcal{A}\}$.

As before, we defer the verification of Construction F to Section VI.

Next, we provide a set of polynomials that satisfies Definition 33.

Lemma 34: Let $g(X)$ be the generator polynomial of a reversible cyclic code $\mathcal{B}$ of length n and dimension k that contains 1^n. Let α be a primitive element of F such that $g(\alpha) \neq 0$ and $g(\alpha^{-1}) \neq 0$. Denote $h^*(X) = M_\alpha(X) M_{\alpha^{-1}}(X)$ and $p(X) = M_\alpha(X)$, then the set $\{h^*(X), p(X)\}$ is (g, k)-rc2-generating.

Proof: We verify conditions in Definition 33. Conditions (R1) to (R4) and (R7) follows directly from the proof of Lemma 29.

Hence, we verify (R5') which reduces to verifying that

$h^*(X)$ does not divide $X^s p(X) - X^{k-1} p(X^{-1})$ for $s \in [\![n]\!]$.

This is equivalent to showing that $r(X) \triangleq X^s p(X) - X^{k-1} p(X^{-1})$ is nonzero for some root of $h^*(X)$. Observe that since α is primitive, we have that α^{-1} is not a root of $M_\alpha(X)$. In other words, $p(\alpha^{-1}) \neq 0$. Since $p(\alpha) = M_\alpha(\alpha) = 0$ and $p(\alpha^{-1}) \neq 0$, we have that $r(\alpha) = \alpha^{k-1} p(\alpha^{-1}) \neq 0$. □

Applying Construction F to the reversible cyclic code in Theorem 2 and the set of polynomials above, we obtain the following family of balanced DNA computing codes.

Corollary 35: Let $m \geq 6$ and $1 \leq \tau \leq \lceil m/2 \rceil$ Set $n = 4^m - 1$, $d = 4^\tau - 1$, and

$$k = \begin{cases} n - (d-3)m, & \text{if } m \text{ is odd and } \tau = \frac{m+1}{2}; \\ n - (d-1)m, & \text{otherwise.} \end{cases}$$

Then there exists a balanced (n, d)-DNA computing code of size at least 4^{k-2m}. Therefore, there exists a family of balanced (n, d)-DNA computing codes with $d \approx \sqrt{n}$ and redundancy at most $(d+1)\log_4(n+1)$. Furthermore, these codes have efficient encoding algorithms.

Setting $m = 6$ and $\tau = 3$, Corollary 35 yields a balanced $(4095, 63)$-DNA computing code of size 4^{3711}. In contrast, ElDin and Matsui [12, Example 3] obtained a balanced $(4095, 2049)$-DNA computing code of size $589968 \approx 4^{9.59}$.

VI. Efficient Encoding Into Cyclic Classes

In this section, unless stated otherwise, all words are of length n and we index them using $[\![n]\!]$. Recall that a word $\boldsymbol{c} \in \mathbb{F}_q^n$ is identified with the polynomial $c(X) = \sum_{i=0}^{n-1} c_i X^i$. We further set $X^n = 1$ and hence, all polynomials reside in the quotient ring $\mathbb{F}_q[X]/\langle X^n - 1 \rangle$.

Hence, in this quotient ring, we have the following properties. Let $c(X) \in \mathbb{F}_q[X]/\langle X^n - 1 \rangle$ be the polynomial corresponding to the word $\boldsymbol{c}$.

- For $s \in [\![n]\!]$, the polynomial $X^s c(X)$ corresponds to the word $\boldsymbol{\sigma}^s(\boldsymbol{c})$.
- $X^{n-1} c(X^{-1})$ corresponds to the word $\boldsymbol{c}^r$.
- $(X^n - 1)/(X - 1)$ corresponds to 1^n, and so, $c(X) + (X^n - 1)/(X - 1)$ corresponds to $\overline{\boldsymbol{c}}$.
- $X^{n-1} c(X^{-1}) + (X^n - 1)/(X - 1)$ corresponds to $\boldsymbol{c}^{rc}$.

From these observations, we can then easily characterise when a cyclic code contains 1^n or when a cyclic code is reversible.

Proposition 36: Let $\mathcal{C}$ be a cyclic code with generator polynomial $g(X)$. Then

(i) $\mathcal{C}$ contains 1^n if and only if $(X - 1)$ does not divide $g(X)$, i.e., $g(1) \neq 0$.
(ii) $\mathcal{C}$ is reversible if and only if $g(X)$ is self-reciprocal.

Next, we review the method of Tavares *et al.* that efficiently encodes into distinct cyclic classes. We restate a special case of their method and reproduce the proof here as the proof is instructive for the subsequent encoding methods.

Theorem 37 (Tavares et al. *[9]):* Let $\mathcal{B}$ be a cyclic code of dimension k with generator polynomial $g(X)$ and define $h(x) = (X^n - 1)/g(X)$. Let $h^*(X)$ be a factor of $h(X)$

such that $h^*(X)$ does not divide X^s-1 for $s \in [n-1]$. Set $k^* = k - \deg h^*(X)$

$$\mathcal{B}^* = \{(m(X)h^*(X)+1)g(X) : \deg m < k^*\}.$$

Then $\mathcal{B}^* \subseteq \mathcal{B}/\underset{\text{cyc}}{\sim}$.

Proof: It suffices to show for distinct polynomials $m(X)$ and $m'(X)$ with $\deg m, \deg m' < k^*$ and $s \in [n-1]$, we have that

$$\begin{aligned} &X^s(m(X)h^*(X)+1)g(X) \\ \neq &(m'(X)h^*(X)+1)g(X) \pmod{X^n-1}. \end{aligned}$$

To do so, we prove by contradiction and suppose that equality holds. In other words, there exists a polynomial $f(X)$ such that

$$\begin{aligned} &X^s(m(X)h^*(X)+1)g(X) \\ =&(m'(X)h^*(X)+1)g(X) + f(X)(X^n-1). \end{aligned}$$

Dividing throughout by $g(X)$ and rearranging the terms, we have that

$$(X^s-1) + (X^s m(X) - m'(X))h^*(X) = f(X)h(X).$$

Since $h^*(X)$ divides $h(X)$, then $h^*(X)$ must divide X^s-1, yielding a contradiction. □

Suppose $n = 2^m - 1$ in Theorem 37. We can choose $h^*(X)$ such that its degree is m. Thus, Theorem 37 encodes into $2^{k-m} = 2^k/(n+1)$ cyclic classes. Since the size of the cyclic code is 2^k and each class contains at most n words, the theorem in fact encodes most cyclic classes.

The method of Tavares *et al.* [9] encodes more classes by considering more factors of $h(X)$ that satisfy the conditions of the theorem. In some special cases, like n is a prime, this iterative process can encode all the cyclic classes.

We also note that all the cyclic classes encoded in Theorem 37 are nonperiodic, that is, each class consists of n distinct shifts. According to Lemma 8, we can obtain two different balanced vectors from each of these classes and then increase the size of the output balanced error-correcting code to $2^{k+1}/(n+1)$.

It is an interesting problem to enumerate all the nonperiodic cyclic classes of a given cyclic code. A partial solution to this problem for Reed-Solomon codes was given by Song and Golomb [28]. However, for binary cyclic codes, this problem remains open.

A. Verification of Construction E

In this subsection, we demonstrate that the code obtained by Construction E is a primer code. The proof consists of two steps. First, we provide in Lemma 38 certain *combinatorial* sufficiency conditions for a subcode of a reversible cyclic code to be a primer code. Next, using the *algebraic* properties of the polynomials in Construction E, we then show that $\mathcal{C}$ satisfies the combinatorial conditions in Lemma 38.

Lemma 38: Let $\mathcal{B}$ be an $(n,k,d)_q$ reversible cyclic code containing 1^n. Let $\mathcal{C} \subseteq \mathcal{B}$ be a subcode such that for any two codewords $\boldsymbol{u}, \boldsymbol{v}$ in $\mathcal{C}$, not necessarily distinct, the following holds.

(S1) $\boldsymbol{\sigma}^i(\boldsymbol{u}) \neq \boldsymbol{v}$ for $k \leq i < n$;
(S2) $\boldsymbol{\sigma}^i(\boldsymbol{u}) \neq \overline{\boldsymbol{v}}$ for $0 \leq i \leq n-k$;
(S3) $\boldsymbol{\sigma}^i(\boldsymbol{u}) \neq \boldsymbol{v}^{rc}$ and $\boldsymbol{\sigma}^i(\boldsymbol{u}^{rc}) \neq \boldsymbol{v}$ for $0 \leq i \leq n-k$.

Then $\mathcal{C}$ is an $(n,d;k,k)_q$-primer code.

Proof: Since $\mathcal{C}$ is a subcode of $\mathcal{B}$, we have that $\mathcal{C}$ is an $(n,d)_q$-code. It remains to show the WMU and APD properties.

We first show that $\mathcal{C}$ is k-WMU. Suppose to the contrary that there is a proper sequence $\boldsymbol{p}$ of length ℓ, where $k \leq \ell < n$, such that both $\boldsymbol{pa}$ and $\boldsymbol{bp}$ belong to $\mathcal{C}$. Since $\mathcal{C} \subseteq \mathcal{B}$ and $\mathcal{B}$ is a cyclic code, the word $\boldsymbol{pa} - \boldsymbol{pb} = 0^\ell(\boldsymbol{a}-\boldsymbol{b})$ belongs to $\mathcal{B}$. Since $\ell \geq k$, Lemma 24 implies that $\boldsymbol{a} = \boldsymbol{b}$ and so,

$$\boldsymbol{\sigma}^\ell(\boldsymbol{bp}) = \boldsymbol{pb} = \boldsymbol{pa}.$$

Since $k \leq \ell < n$ and $\boldsymbol{pa}$ and $\boldsymbol{bp}$ belong to $\mathcal{C}$, we obtain a contradiction for condition (S1).

Now we show that $\mathcal{C}$ is a k-APD code. Towards a contradiction, we suppose that there is a proper sequence $\boldsymbol{p}$ of length k, such that both $\boldsymbol{a}_1\boldsymbol{p}\boldsymbol{b}_1$ and $\boldsymbol{a}_2\overline{\boldsymbol{p}}\boldsymbol{b}_2$ belong to $\mathcal{C}$. Since $\mathcal{C} \subseteq \mathcal{B}$ and $\mathcal{B}$ is a cyclic code containing 1^n, the word $\boldsymbol{p}\boldsymbol{b}_1\boldsymbol{a}_1 - \boldsymbol{p}\overline{\boldsymbol{b}_2}\overline{\boldsymbol{a}_2} = \boldsymbol{0}(\boldsymbol{b}_1\boldsymbol{a}_1 - \overline{\boldsymbol{b}_2}\overline{\boldsymbol{a}_2})$ also belongs to $\mathcal{B}$. It follows from Lemma 24 that

$$\boldsymbol{p}\boldsymbol{b}_1\boldsymbol{a}_1 = \boldsymbol{p}\overline{\boldsymbol{b}_2}\overline{\boldsymbol{a}_2}.$$

Without loss of generality, we assume that $|\boldsymbol{a}_2| \geq |\boldsymbol{a}_1|$. Then

$$\boldsymbol{\sigma}^{\ell'}\left(\boldsymbol{a}_1\boldsymbol{p}\boldsymbol{b}_1\right) = \overline{\boldsymbol{a}_2}\boldsymbol{p}\overline{\boldsymbol{b}_2} = \overline{\boldsymbol{a}_2\overline{\boldsymbol{p}}\boldsymbol{b}_2},$$

where $\ell' = |\boldsymbol{a}_2| - |\boldsymbol{a}_1|$. Since the length of $\boldsymbol{p}$ is k, we have that $\ell' \leq n-k$, which contradicts condition (S2).

Finally, suppose that $\boldsymbol{a}_1\boldsymbol{p}\boldsymbol{b}_1$ and $\boldsymbol{a}_2\boldsymbol{p}^{rc}\boldsymbol{b}_2$ belong to $\mathcal{C}$, where $\boldsymbol{p}$ is a proper sequence of length k. Proceeding as before, we can show that

$$\boldsymbol{p}\boldsymbol{b}_1\boldsymbol{a}_1 = \boldsymbol{p}\boldsymbol{a}_2^{rc}\boldsymbol{b}_2^{rc},$$

or equivalently,

$$\boldsymbol{b}_1\boldsymbol{a}_1 = \boldsymbol{a}_2^{rc}\boldsymbol{b}_2^{rc} \text{ and } \boldsymbol{a}_1^{rc}\boldsymbol{b}_1^{rc} = \boldsymbol{b}_2\boldsymbol{a}_2.$$

If $|\boldsymbol{b}_2| \geq |\boldsymbol{a}_1|$, we have that

$$\boldsymbol{\sigma}^{\ell'}\left(\boldsymbol{a}_1\boldsymbol{p}\boldsymbol{b}_1\right) = \boldsymbol{b}_2^{rc}\boldsymbol{p}\boldsymbol{a}_2^{rc} = (\boldsymbol{a}_2\boldsymbol{p}^{rc}\boldsymbol{b}_2)^{rc},$$

where $\ell' = |\boldsymbol{b}_2| - |\boldsymbol{a}_1| \leq n-k$, contradicting the first inequality of Condition (S3); if $|\boldsymbol{b}_2| < |\boldsymbol{a}_1|$, then $|\boldsymbol{a}_2| > |\boldsymbol{b}_1|$ and we have

$$\boldsymbol{\sigma}^{\ell'}\left((\boldsymbol{a}_1\boldsymbol{p}\boldsymbol{b}_1)^{rc}\right) = \boldsymbol{\sigma}^{\ell'}\left(\boldsymbol{b}_1^{rc}\boldsymbol{p}^{rc}\boldsymbol{a}_1^{rc}\right) = \boldsymbol{a}_2\boldsymbol{p}^{rc}\boldsymbol{b}_2,$$

where $\ell' = |\boldsymbol{a}_2| - |\boldsymbol{b}_1| \leq n-k$, contradicting the second inequality of Condition (S3). □

Borrowing ideas from Tavares *et al.*, we demonstrate that the code obtained by Construction E is a primer code. That is, we have the following result.

Theorem 39: Let $g(X)$ be the generator polynomial of a reversible cyclic code $\mathcal{B}$ of length n and dimension k that contains 1^n. Let $\{h(X), p_1(X), p_2(X), \ldots, p_P(X)\}$ be a (g,k)-rc-generating set and $k^* = k - \deg h^*$. Then the subcode $\mathcal{C} = \{(m(X)h^*(X) + p_i(X))g(X) : \deg m < k^*, i \in [P]\}$ satisfies conditions (S1) to (S3) in Lemma 38, and so it is an $(n,d;k,k)_q$-primer code.

Proof: Here we only prove condition (S3). The other two conditions can be proved similarly. In particular, we demonstrate that the violation of condition (S3) contradicts either condition (R5) or condition (R6) in Definition 25, i.e.,

(R5) $h^*(X)$ does not divide $X^s p_i(X) - X^{k-1} p_j(X^{-1})$ for all $i, j \in [P]$ and $0 \leq s \leq n-k$;
(R5) $h^*(X)$ does not divide $X^{s+k-1} p_i(X^{-1}) - p_j(X)$ for all $i, j \in [P]$ and $0 \leq s \leq n-k$.

Suppose to the contrary of (S3). We first assume there are two codewords $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{C}$ such that $\boldsymbol{\sigma}^s(\boldsymbol{u}) = \boldsymbol{v}^{rc}$ for some $s \in [\![n-k]\!]$. The other case can be treated similarly. Hence, there exist two polynomials $m(X)$ and $m'(X)$ with degrees strictly less than k^*, two polynomials $p_i(X)$ and $p_j(X)$ with $i, j \in [P]$ such that the following equality holds with some polynomial $f(X)$.

$$\begin{aligned} & X^s \left(m(X)h^*(X) + p_i(X)\right) g(X) \\ =& X^{n-1} \left(m'(X^{-1})h^*(X^{-1}) + p_j(X^{-1})\right) g(X^{-1}) \\ & + \frac{X^n - 1}{X - 1} + f(X)(X^n - 1). \end{aligned}$$

Since $\mathcal{B}$ is a reversible code, $g(x)$ is self-reciprocal, i.e., $X^{n-k} g(X^{-1}) = g(X)$. Recall that condition (R3) in Definition 25 is that $h^*(X) = X^{\deg h^*} h^*(X^{-1})/h^*(0)$. Dividing the equation by $g(X)$ and rearranging the terms, we have the following equality.

$$\begin{aligned} & \left(X^s p_i(X) - X^{k-1} p_j(X^{-1})\right) \\ & + \left(X^s m(X) - X^{k^*-1} m'(X^{-1}) h^*(0)\right) h^*(X) \\ =& \frac{h(X)}{X-1} + f(X)h(X). \end{aligned}$$

Due to conditions (R1) and (R2) in Definition 25, i.e., $h^*(X)$ divides $h(X)$ and $h^*(1) \neq 0$, we have that $h^*(X)$ divides $h(X)/(X-1)$. Therefore,

$$h^*(X) \text{ divides } X^s p_i(X) - X^{k-1} p_j(X^{-1}),$$

contradicting condition (R5) in Definition 25. □

B. Verification of Construction F

In this subsection, we demonstrate that the code $\mathcal{C}$ obtained in Construction F is a balanced $(n, d)_4$-DNA computing code. As before, we first provide certain *combinatorial* sufficiency conditions for a subcode of a reversible cyclic code to be a DNA computing code, and then show that $\mathcal{C}$ satisfies these combinatorial conditions.

Lemma 40: Let $\mathcal{B}$ be an $(n, k, d)_q$ reversible cyclic code containing 1^n. Let $\mathcal{A} \subseteq \mathcal{B}$ be a subcode such that for any two codewords $\boldsymbol{u}, \boldsymbol{v}$ in $\mathcal{A}$, not necessarily distinct, the following holds.

(S1') $\boldsymbol{\sigma}^i(\boldsymbol{u}) \neq \boldsymbol{v}$ for $i \in [n-1]$;
(S2') $\boldsymbol{\sigma}^i(\boldsymbol{u}) \neq \boldsymbol{v}^r$ for $i \in [\![n]\!]$;
(S3') $\boldsymbol{\sigma}^i(\boldsymbol{u}) \neq \boldsymbol{v}^{rc}$ for $i \in [\![n]\!]$.

For each $\boldsymbol{u} \in \mathcal{A}$, find $i_{\boldsymbol{u}} \in [\![n-1]\!]$ such that $\boldsymbol{v}_{\boldsymbol{u}} = \pi(\boldsymbol{\sigma}^{i_{\boldsymbol{u}}}(\boldsymbol{u}))$ is GC-balanced. Set $\mathcal{C} = \{\boldsymbol{v}_{\boldsymbol{u}} : \boldsymbol{u} \in \mathcal{A}\}$. Then $\mathcal{C}$ is a balanced (n, d)-DNA computing code of size $|\mathcal{A}|$.

Proof: First, condition (S1') ensures that the codewords $\boldsymbol{v}_{\boldsymbol{u}}$ and $\boldsymbol{v}'_{\boldsymbol{u}}$ are distinct whenever $\boldsymbol{u} \neq \boldsymbol{u}'$. Therefore, the size of $\mathcal{C}$ is given by $|\mathcal{A}|$.

Next, by the choice of $i_{\boldsymbol{u}}$, we have that all codewords in $\mathcal{C}$ are GC-balanced. Since $\mathcal{C}$ belongs to a coset of $\mathcal{B}$, we have that $\mathcal{C}$ is an $(n, d)_4$-code.

Therefore, it remains to demonstrate that for any $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}$, $d(\boldsymbol{a}, \boldsymbol{b}^r) \geq d$ and $d(\boldsymbol{a}, \boldsymbol{b}^{rc}) \geq d$. Let $\boldsymbol{u}, \boldsymbol{v}$ be the corresponding vectors in $\mathcal{A}$. In other words,

$$\boldsymbol{a} = \pi(\boldsymbol{\sigma}^{i_{\boldsymbol{u}}}(\boldsymbol{u})) \text{ and } \boldsymbol{b} = \pi(\boldsymbol{\sigma}^{i_{\boldsymbol{v}}}(\boldsymbol{v})).$$

We first show that $d(\boldsymbol{a}, \boldsymbol{b}^r) \geq d$. Since $\mathcal{A}$ satisfies condition (S2'), we have that $\boldsymbol{\sigma}^{i_{\boldsymbol{u}}}(\boldsymbol{u}) \neq \boldsymbol{\sigma}^{i_{\boldsymbol{v}}}(\boldsymbol{v})^r$. Since $\boldsymbol{\sigma}^{i_{\boldsymbol{u}}}(\boldsymbol{u})$, $\boldsymbol{\sigma}^{i_{\boldsymbol{v}}}(\boldsymbol{v})^r$ belongs to $\mathcal{B}$, we have that $d(\boldsymbol{\sigma}^{i_{\boldsymbol{u}}}(\boldsymbol{u}), \boldsymbol{\sigma}^{i_{\boldsymbol{v}}}(\boldsymbol{v})^r) \geq d$. Now, $\omega^s 0^{n-2s} \omega^s = (\omega^s 0^{n-2s} \omega^s)^r$, and so, $\boldsymbol{b}^r = \pi(\boldsymbol{\sigma}^{i_{\boldsymbol{v}}}(\boldsymbol{v}))^r = \pi(\boldsymbol{\sigma}^{i_{\boldsymbol{v}}}(\boldsymbol{v})^r)$. Therefore,

$$d(\boldsymbol{a}, \boldsymbol{b}^r) = d(\pi(\boldsymbol{\sigma}^{i_{\boldsymbol{u}}}(\boldsymbol{u})), \pi(\boldsymbol{\sigma}^{i_{\boldsymbol{v}}}(\boldsymbol{v})^r)) = d((\boldsymbol{\sigma}^{i_{\boldsymbol{u}}}(\boldsymbol{u}), \boldsymbol{\sigma}^{i_{\boldsymbol{v}}}(\boldsymbol{v})^r)) \geq d.$$

The fact that $d(\boldsymbol{a}, \boldsymbol{b}^{rc}) \geq d$ can be similarly demonstrated. □

Lemma 41: Let $g(X)$ be the generator polynomial of a reversible cyclic code $\mathcal{B}$ of length n and dimension k that contains 1^n. Let $\{h(X), p_1(X), p_2(X), \ldots, p_P(X)\}$ be a (g, k)-rc2-generating set and $k^* = k - \deg h^*$. Then the subcode $\mathcal{A} = \{(m(X)h^*(X) + 1)g(X) : \deg m < k^*\}$ satisfies conditions (S1') to (S3') in Lemma 40.

Proof: Here we only prove condition (S2'). The other two conditions can be proved similarly. In particular, we demonstrate that the violation of condition (S2') contradicts the condition

(R5') $h^*(X)$ does not divide $X^s p_i(X) - X^{k-1} p_j(X^{-1})$ for all $i, j \in [P]$ and $s \in [\![n]\!]$,

in Definition 33.

Suppose to the contrary of (S2') that we have two codewords $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{A}$ such that $\boldsymbol{\sigma}^s(\boldsymbol{u}) = \boldsymbol{v}^r$ for some $s \in [\![n]\!]$. Hence, there exists two polynomials $m(X)$ and $m'(X)$ with degrees strictly less than k^*, two polynomials $p_i(X)$ and $p_j(X)$ with $i, j \in [P]$ such that the following equality holds with some polynomial $f(X)$.

$$\begin{aligned} & X^s \left(m(X)h^*(X) + p_i(X)\right) g(X) \\ =& X^{n-1} \left(m'(X^{-1})h^*(X^{-1}) + p_j(X^{-1})\right) g(X^{-1}) \\ & + f(X)(X^n - 1). \end{aligned}$$

As before, by the choice of g and h, we have that $X^{n-k} g(X^{-1}) = g(X)$ and $h^*(X) = X^{\deg h^*} h^*(X^{-1})/h^*(0)$. Dividing the equation by $g(X)$ and rearranging the terms, we have the following equality.

$$\begin{aligned} & \left(X^s p_i(X) - X^{k-1} p_j(X^{-1})\right) \\ & + \left(m(X) - X^{k^*} m'(X^{-1}) h^*(0)\right) h^*(X) \\ & = f(X)h(X). \end{aligned}$$

Therefore,

$$h^*(X) \text{ divides } p_i(X) - X^{k-1} p_j(X^{-1}),$$

contradicting condition (R5') in Definition 33. □

Combining Lemma 40 and Lemma 41, we have the following result.

Corollary 42: The code $\mathcal{C}$ obtained in Construction F is a balanced (n, d)-DNA computing code.

VII. Conclusion

We provide efficient and explicit methods to construct balanced codes, primer codes and DNA computing codes with error-correcting capabilities. Using certain classes of BCH codes as inputs, we obtain infinite families of $(n, d)_q$-codes satisfying our constraints with redundancy $C_d \log n + O(1)$. Here, C_d is a constant dependent only on d. Note that in all our constructions, we have $C_d \leq d + 1$. On the other hand, the sphere-packing bound requires $C_d \geq \lfloor (d-1)/2 \rfloor$. Therefore, it remains open to provide efficient and explicit constructions that reduce the value of C_d further.

References

[1] Y. M. Chee, H. M. Kiah, and H. Wei, "Efficient and explicit balanced primer codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 91–95.

[2] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Sci. Rep.*, vol. 7, no. 1, p. 5011, Jul. 2017.

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

[4] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," 2018, *arXiv:1801.04882*. [Online]. Available: http://arxiv.org/abs/1801.04882

[5] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, no. 1, p. 14138, Sep. 2015.

[6] S. M. H. T. Yazdi, H. M. Kiah, R. Gabrys, and O. Milenkovic, "Mutually uncorrelated primers for DNA-based data storage," *IEEE Trans. Inf. Theory*, vol. 64, no. 9, pp. 6283–6296, Sep. 2018.

[7] D. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. 32, no. 1, pp. 51–53, Jan. 1986.

[8] J. L. Massey, "Reversible codes," *Inf. Control*, vol. 7, no. 3, pp. 369–380, Sep. 1964.

[9] S. E. Tavares, P. E. Allard, and S. G. S. Shiva, "On the decomposition of cyclic codes into cyclic classes," *Inf. Control*, vol. 18, no. 4, pp. 342–354, May 1971.

[10] J. H. Weber, K. A. S. Immink, and H. C. Ferreira, "Error-correcting balanced knuth codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 1, pp. 82–89, Jan. 2012.

[11] S. Al-Bassam and B. Bose, "Design of efficient error-correcting balanced codes," *IEEE Trans. Comput.*, vol. 42, no. 10, pp. 1261–1266, Oct. 1993.

[12] R. T. ElDin and H. Matsui, "On constant GC-content cyclic DNA codes with long codewords," in *Proc. IEEE Int. Symp. Int. Symp. Inf. Theory Appl. (ISITA)*, Oct. 2018, pp. 21–25.

[13] F. J. MacWilliams and N. J. A. Sloane, *The Theory Error-Correcting Codes*. Amsterdam, The Netherlands: Elsevier, 1977.

[14] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli, "On quantum and classical BCH codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 3, pp. 1183–1188, Mar. 2007.

[15] C. Li, C. Ding, and S. Li, "LCD cyclic codes over finite fields," *IEEE Trans. Inf. Theory*, vol. 63, no. 7, pp. 4344–4356, Jul. 2017.

[16] S. Li, C. Li, C. Ding, and H. Liu, "Two families of LCD BCH codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5699–5717, Jul. 2017.

[17] K. Tzeng and C. Hartmann, "On the minimum distance of certain reversible cyclic codes (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-16, no. 5, pp. 644–646, Sep. 1970.

[18] H. van Tilborg and M. Blaum, "On error-correcting balanced codes," *IEEE Trans. Inf. Theory*, vol. 35, no. 5, pp. 1091–1095, Sep. 1989.

[19] F. W. Fu and V. K. W. Wei, "Self-complementary balanced codes and quasi-symmetric designs," *Designs, Codes Cryptogr.*, vol. 27, no. 3, pp. 271–279, 2002.

[20] A. Mazumdar, R. M. Roth, and P. O. Vontobel, "On linear balancing sets," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2009, pp. 2699–2703.

[21] D. Limbachiya, B. Rao, and M. K. Gupta, "The art of DNA strings: Sixteen years of DNA coding theory," 2016, *arXiv:1607.00266*. [Online]. Available: http://arxiv.org/abs/1607.00266

[22] K. A. S. Immink and K. Cai, "Properties and constructions of constrained codes for DNA-based data storage," 2018, *arXiv:1812.06798*. [Online]. Available: http://arxiv.org/abs/1812.06798

[23] P. E. Allard, S. G. S. Shiva, and S. E. Tavares, "A note on the decomposition of cyclic codes into cyclic classes," *Inf. Control*, vol. 22, no. 1, pp. 100–106, Feb. 1973.

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

[25] L. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 5187, pp. 1021–1024, Nov. 1994. [Online]. Available: http://science.sciencemag.org/content/266/5187/1021

[26] A. Marathe, A. E. Condon, and R. M. Corn, "On combinatorial DNA word design," *J. Comput. Biol.*, vol. 8, no. 3, pp. 201–219, 2001.

[27] O. Milenkovic and N. Kashyap, "On the design of codes for DNA computing," in *Proc. Int. Conf. Coding Cryptogr. (WCC)*. Berlin, Germany: Springer-Verlag, 2006, pp. 100–119.

[28] H. Y. Song and S. W. Golomb, "On the nonperiodic cyclic equivalence classes of reed-solomon codes," *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1431–1434, Jul. 1993.

Yeow Meng Chee received the B.Math., M.Math., and Ph.D. degrees in computer science from the University of Waterloo in 1988, 1989, and 1996, respectively.

He is currently a Professor with the Department of Industrial Systems Engineering and Management, and the Associate Vice President (Innovation & Enterprise) with the National University of Singapore (NUS). Prior to joining NUS, he was the Head of the Division of Mathematical Sciences from 2008 to 2010, the Chair of the School of Physical and Mathematical Sciences from 2011 to 2017, and the Interim Dean of the College of Science, Nanyang Technological University, from 2018 to 2019. He has held senior positions in public service, including the Head of Security (Information Infrastructure) and an Assistant Director of Internationalization at the National Computer Board, the Deputy Director of Strategic Programs at the Infocomm Development Authority (IDA), and a Program Director of Interactive Digital Media Research and Development in the Media Development Authority. He deployed South East Asia's first certification authority Netrust in 1997, and also founded the Singapore Computer Emergency Response Team (SingCERT). His research interests lie in the interplay between combinatorics and computer science, especially coding theory, extremal set systems, and their applications.

Dr. Chee has represented Singapore in various international forums, including a member of the APEC Electronic Commerce Task Force in 1998, a member of the ASEAN Coordinating Committee on Electronic Commerce in 1998, a member of the Working Group on ASEAN Information Infrastructure in 1999, the Secretariat of the Canada-Singapore IT Joint Council in 1998, the Co-Chair of the APEC TEL Public Key Interoperability Expert Group in 1999, the Secretariat of the Australia-Singapore ICT Joint Council in 1999, and a member of APEC Project DARE (Data Analytics Raising Employment) Advisory Board in 2017. He is also a Fellow and a Council Member of the Institute of Combinatorics and its Applications.

Han Mao Kiah (Member, IEEE) received the Ph.D. degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2014. From 2014 to 2015, he was a Post-Doctoral Research Associate with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. From 2015 to 2018, he was a Lecturer with the School of Physical and Mathematical Sciences (SPMS), NTU, where he is currently an Assistant Professor. His research interests include DNA-based data storage, coding theory, enumerative combinatorics, and combinatorial design theory.

Hengjia Wei received the Ph.D. degree in applied mathematics from Zhejiang University, Hangzhou, Zhejiang, China, in 2014.

He was a Post-Doctoral Fellow with Capital Normal University, Beijing, China, from 2014 to 2016, and a Research Fellow with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, from 2016 to 2019. He is currently a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel. His research interests include combinatorial design theory, coding theory, and their intersections.

Dr. Wei received the 2017 Kirkman Medal from the Institute of Combinatorics and its Applications.