Non-binary Two-Deletion Correcting Codes and Burst-Deletion Correcting Codes

Wentu Song and Kui Cai

Science, Mathematics and Technology Cluster

Singapore University of Technology and Design, Singapore 487372

Email: {wentu_song, cai_kui}@sutd.edu.sg

Abstract

In this paper, we construct systematic q-ary two-deletion correcting codes and burst-deletion correcting codes, where $q \geq 2$ is an even integer. For two-deletion codes, our construction has redundancy $5 \log n + O(\log q \log \log n)$ and has encoding complexity near-linear in n, where n is the length of the message sequences. For burst-deletion codes, we first present a construction of binary codes with redundancy $\log n + 9 \log \log n + \gamma_t + o(\log \log n)$ bits (γ_t is a constant that depends only on t) and capable of correcting a burst of at most t deletions, which improves the Lenz-Polyanskii Construction (ISIT 2020). Then we give a construction of q-ary codes with redundancy $\log n + (8 \log q + 9) \log \log n + o(\log q \log \log n) + \gamma_t$ bits and capable of correcting a burst of at most t deletions.

I. INTRODUCTION

DNA-based data storage has been a hot topic in information theory society. As deletion/insertion are common in DNA data storage [1], codes correcting such errors have attracted significant attention in recent years.

It was proved in [2] that the optimal redundancy of binary t-deletion correcting codes is asymptotically between $t \log n + o(\log n)$ and $2t \log n + o(\log n)$, where n is the length of the code and the redundancy of a binary code $\mathcal{C}$ is defined as $n - \log |\mathcal{C}|$.[1] The well-known Varshamov-Tenengolts (VT) codes [3], which is defined as

$$\mathrm{VT}_a(n) = \left\{ (c_1, \ldots, c_n) \in \{0,1\}^n : \sum_{i=1}^{n} i c_i \equiv a \bmod (n+1) \right\},$$

is a class of binary single-deletion correcting codes with asymptotically optimal redundancy. Construction of multiple-deletion correcting codes with low redundancy were considered in [4]−[12]. By using the higher order VT syndromes and the syndrome compression technique [10], Sima *et al.* constructed a family of systematic t-deletion correcting codes with $4t \log n + o(\log n)$ bits [11]. The method in [11] was improved in [12] to give a construction of t-deletion correcting codes with redundancy $(4t - 1) \log n + o(\log n)$, which is the best known result in redundancy. For the special case of $t = 2$, an explicit construction of 2-deletion correcting codes with redundancy $4 \log n + o(\log n)$ was proposed by Guruswami and Håstad [7], which matches the existential upper bound of the asymptotically optimal codes.

As a special case of deletion errors, a burst of t deletions (or a t-burst-deletion) refers to t deletions that occur at consecutive positions. It was proved in [13] that the redundancy of a t-burst-deletion-correcting code is approximately lower bounded by $\log n + t - 1$. Levenshtein [14] constructed a class of binary codes that can correct a burst of at most two deletions with asymptotically optimal redundancy of $\log n + 1$. Binary codes capable of correcting a burst of *exact* t deletions for $t \geq 2$ are constructed in [13], which also have an asymptotically optimal redundancy of $\log n + (t-1) \log \log n + t - \log t$. In [15], binary codes capable of correcting a burst of *at most* t deletions are constructed, which also have an asymptotically optimal redundancy of $\log n + (t(t-1)/2) \log \log n + \gamma_t$, where γ_t is a constant that depends only on t.

Besides binary codes, nonbinary deletion correcting codes are also investigated in the literature. In [16], it was shown that the optimal redundancy of a q-ary t-deletion correcting code is asymptotically lower bounded by $t \log n + t \log q + o(\log q \log n)$ and upper bounded by $2t \log n + t \log q + o(\log q \log n)$ in bits ($q \geq 2$). A class of q-ary single-deletion correcting codes with redundancy close to the asymptotic optimality was constructed in [17]. For q-ary t-deletion correcting codes, the best known construction is presented in [18], which achieve optimal redundancy up to a constant factor. Quaternary codes capable of correcting a single edit error for DNA data storage were studied in [19]. In [20], a q-ary code that can correct a burst of at most 2 deletions with redundancy $\log n + O(\log q \log \log n)$ bits was constructed, where $q \geq 2$ is an even integer.

In this paper, we construct nonbinary two-deletion correcting codes and burst-deletion correcting codes. Our contributions includes:

1) We construct a class of systematic q-ary two-deletion correcting codes, with redundancy $5 \log n + O(\log q \log \log n)$, where $q \geq 2$ is an even integer and n is the length of the message sequences.

---

[1]In this paper, for any positive real number x, $\log_q x$ is the logarithm of x with base q, where $q \geq 2$ is a positive integer. If the base $q = 2$, then for simplicity, we write $\log_2 x = \log x$.

2) We present a construction of binary codes with redundancy $\log n + 9 \log \log n + \gamma_t + o(\log \log n)$ bits (γ_t is a constant that depends only on t) and capable of correcting a burst of at most t deletions, which improves the Lenz-Polyanskii Construction (ISIT 2020).

2) We give a construction of q-ary codes with redundancy $\log n + (8 \log q + 9) \log \log n + o(\log q \log \log n) + \gamma_t$ bits and capable of correcting a burst of at most t deletions, where $q \geq 2$ is an even integer.

Note that each symbol in $\mathbb{Z}_q$ can be viewed as a binary string of length $\lceil \log q \rceil$, so a binary code of length $\lceil \log q \rceil n$ and capable of correcting a burst of $\lceil \log q \rceil t$ deletions can also be viewed as a q-ary code of length n and capable of correcting a burst of t deletions. By this observation and by the construction in [15], we can obtain a q-ary code of length n and capable of correcting a burst of t deletions that has redundancy

$$\log(n \log q) + \frac{t \log q (t \log q + 1)}{2} \log \log(n \log q) + \gamma_t.$$

Our construction has improved redundancy than this naive construction.

The rest of this paper is organized as follows. In Section II, we introduce some basic concepts and notations of deletion correcting codes, and review some related constructions in the literature. In Section III, we construct q-ary two-deletion correcting codes. In Section IV, we present an improved construction of binary codes correcting a burst of at most t deletions. In Section V, we construct of q-ary codes correcting a burst of at most t deletions. The paper is concluded in Section VI.

II. PRELIMINARIES

For any integers m and n such that $m \leq n$, we denote $[m, n] = \{m, m+1, \ldots, n\}$ and call it an *interval*. If $m > n$, let $[m, n] = \emptyset$. For simplicity, denote $[n] = [1, n]$ for any positive integer n. For any positive real number x, $\log x$ is the logarithm of x with base 2, i.e., $\log x = \log_2 x$. The size (cardinality) of any set S is denoted by $|S|$. For any positive integer $q \geq 2$, denote $\mathbb{Z}_q = \{0, 1, 2, \cdots, q-1\}$, which will be used as the alphabet of q-ary codes.

For any string (also called a sequence) $\boldsymbol{x} \in \mathbb{Z}_q^n$, n is called the length of $\boldsymbol{x}$ and denote $|\boldsymbol{x}| = n$. Unless otherwise specified, we use x_i to denote the ith coordinate of $\boldsymbol{x}$, where $i \in [n]$. Usually, we denote $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ or $\boldsymbol{x} = x_1 x_2 \cdots x_n$. For any $I = \{i_1, i_2, \ldots, i_d\} \subseteq [n]$ such that $i_1 < i_2 < \cdots < i_d$, denote $x_I = x_{i_1} x_{i_2} \cdots x_{i_d}$ and call x_D a *subsequence* of $\boldsymbol{x}$. If $I \subseteq [n]$ is an interval (i.e., $I = [i, j]$ for some $i, j \in [1, n]$, $i \leq j$), then $x_I = x_{[i,j]} = x_i x_{i+1} \cdots x_j$ is called a *substring* of $\boldsymbol{x}$. In other words, a substring of $\boldsymbol{x}$ is a subsequence of $\boldsymbol{x}$ consisting of some consecutive symbols of $\boldsymbol{x}$. We say that $\boldsymbol{x}$ contains $\boldsymbol{p}$ (or $\boldsymbol{p}$ is contained in $\boldsymbol{x}$) if $\boldsymbol{p}$ is a substring of $\boldsymbol{x}$. For two substrings x_I and $x_{I'}$ of $\boldsymbol{x}$, where $I, I' \subseteq [n]$ are two intervals, we say that x_I and $x_{I'}$ are *disjoint* if $I \cap I' = \emptyset$.

Let $t \leq n$ be a nonnegative integer. For any $\boldsymbol{x} \in \mathbb{Z}_q^n$, let $\mathcal{D}_t(\boldsymbol{x})$ denote the set of subsequences of $\boldsymbol{x}$ of length $n - t$, and let $\mathcal{B}_t(\boldsymbol{x})$ denote the set of subsequences $\boldsymbol{y}$ of $\boldsymbol{x}$ that can be obtained from $\boldsymbol{x}$ by a burst of t deletions, that is $\boldsymbol{y} = x_I$ such that $I = [n] \backslash D$ for some interval $D \subseteq [n]$ of length t (i.e., $D = [i, i+t-1]$ for some $i \in [n-t+1]$). Moreover, let $\mathcal{B}_{\leq t}(\boldsymbol{x}) = \bigcup_{t'=0}^{t} \mathcal{B}_{t'}(\boldsymbol{x})$ be the set of subsequences of $\boldsymbol{x}$ that can be obtained from $\boldsymbol{x}$ by a burst of at most t deletions. Clearly, $\mathcal{D}_1(\boldsymbol{x}) = \mathcal{B}_1(\boldsymbol{x}) = \mathcal{B}_{\leq 1}(\boldsymbol{x})$. However, $\mathcal{B}_t(\boldsymbol{x}) \subseteq \mathcal{D}_t(\boldsymbol{x}) \cap \mathcal{B}_{\leq t}(\boldsymbol{x})$ for $t \geq 2$.

A code $\mathcal{C} \subseteq \mathbb{Z}_q^n$ is said to be a *t-deletion correcting code* if for any $\boldsymbol{x} \in \mathcal{C}$ and any $\boldsymbol{y} \in \mathcal{D}_t(\boldsymbol{x})$, $\boldsymbol{x}$ can be uniquely recovered from $\boldsymbol{y}$; the code $\mathcal{C} \subseteq \mathbb{Z}_q^n$ is said to be capable of *correcting a burst of at most t deletions* if for any $\boldsymbol{x} \in \mathcal{C}$ and any $\boldsymbol{y} \in \mathcal{B}_{\leq t}(\boldsymbol{x})$, $\boldsymbol{x}$ can be uniquely recovered from $\boldsymbol{y}$.

A. Some Constructions Related to Binary Single-deletion and Two-deletion Correcting Codes

From the VT construction, we can obtain the following lemma about single-deletion correcting codes.

Lemma 1: For any integer $n \geq 3$, there exists a function $\text{VT} : \{0, 1\}^n \to \{0, 1\}^{\log n}$, computable in linear time, such that for any $\boldsymbol{c} \in \{0, 1\}^n$, given $\text{VT}(\boldsymbol{c})$ and any $\boldsymbol{b} \in \mathcal{D}_1(\boldsymbol{c})$, one can uniquely recover $\boldsymbol{c}$.

The following lemma can be obtained from the results of [6], and so its proof is omitted.

Lemma 2: For any integer $n \geq 3$, there exists a function $\xi : \{0, 1\}^n \to \{0, 1\}^{7 \log n + o(\log n)}$, computable in linear time, such that for any $\boldsymbol{c} \in \{0, 1\}^n$, given $\xi(\boldsymbol{c})$ and any $\boldsymbol{b} \in \mathcal{D}_2(\boldsymbol{c})$, one can uniquely recover $\boldsymbol{c}$.

Lemma 2 can be used to construct systematic binary two-deletion correcting codes with redundancy not greater than $7 \log n + o(\log n)$. Another construction, which uses the so-called regular strings and has lower redundancy, was proposed in [7], but it is not systematic.

Definition 1 (Regularity): A binary string $\boldsymbol{c} \in \{0, 1\}^n$ is said to be *regular* if each (contiguous) sub-string of $\boldsymbol{c}$ of length at least $d \log n$ contains both 00 and 11.

In Definition 1, d is a constant that can be chosen properly. In this paper, we will always choose $d = 7$. The following two lemmas are from [7].

Lemma 3: [7, Lemma 11] There exist an integer $M \geq 2^{n-1}$ and a one-to-one mapping $\text{RegEnc} : \{1, 2, \cdots, M\} \to \{0, 1\}^n$ such that its image is contained in the set of regular strings. Moreover, the function RegEnc can be computed in near-linear time with a polynomial size lookup table.

Lemma 4: [7, Theorem 7] There is a function η, computable in linear time, that maps n bits to $4\log n + 10\log\log n + O(1)$ bits such that for any regular $\boldsymbol{c} \in \{0,1\}^n$, given $\eta(\boldsymbol{c})$ and any $\boldsymbol{b} \in \mathcal{D}_2(\boldsymbol{c})$, one can uniquely recover $\boldsymbol{c}$.

B. Some Constructions Related to Binary Burst-Deletion Correcting Codes

The following lemma can be obtained from the results in Section IV of [10].

Lemma 5: Suppose t is a constant with respect to n. There is a function $\phi : \{0,1\}^n \to \{0,1\}^{4\log n + o(\log n)}$, computable in time $O(2^t n^3)$, such that for any $\boldsymbol{c} \in \{0,1\}^n$, given $\phi(\boldsymbol{c})$ and any $\boldsymbol{b} \in \mathcal{B}_{\leq t}(\boldsymbol{c})$, one can uniquely recover $\boldsymbol{c}$.

Let $m \leq \delta \leq n$ be positive integers and $\boldsymbol{p} \in \{0,1\}^m$, where $\boldsymbol{p}$ is called a *pattern*. A string $\boldsymbol{c} \in \{0,1\}^n$ is called $(\boldsymbol{p}, \delta)$-*dense*, if each substring of $\boldsymbol{c}$ of length δ contains at least one pattern $\boldsymbol{p}$.

As in [15], in this paper, we take

$$\delta = t2^{t+1}\log n^2$$

and

$$\boldsymbol{p} = 0^t 1^t,$$

where 0^t is the string consists of t symbol 0s, and 1^t is the string consists of t symbol 1s. In other words, $\boldsymbol{p} = p_1 p_2 \cdots p_{2t}$ such that $p_1 = p_2 = \cdots = p_t = 0$ and $p_{t+1} = p_{t+2} = \cdots = p_{2t} = 1$. It was proven in [15] that one bit of redundancy is sufficient to construct $(\boldsymbol{p}, \delta)$-dense string.

Lemma 6: [15, Lemma 1] For any $n \geq 5$, the number of $(\boldsymbol{p}, \delta)$-dense strings of length n is at least

$$2^n(1 - n^{1-\log e}) \geq 2^{n-1}.$$

The following lemma can be obtained from Construction 1 and Lemma 2 of [15] and so its proof is omitted.

Lemma 7: For any positive integer n, there is a function μ, computable in linear time, that maps n bits to $\log n + 3$ bits such that for any $(\boldsymbol{p}, \delta)$-dense $\boldsymbol{c} \in \{0,1\}^n$, given $\mu(\boldsymbol{c})$ and any $\boldsymbol{b} \in \mathcal{B}_{\leq t}(\boldsymbol{c})$, one can find in time $O(n)$ an interval $L \subseteq [n]$ of length at most $\delta + t$ such that $\boldsymbol{b} = \boldsymbol{c}_{[n]\backslash D}$ for some interval $D \subseteq L$ (i.e., the deletions are located in the interval L).

C. Matrix Representation of q-ary Strings

In the rest of this paper, we always assume $q > 2$ is a fixed even integer. As in [18], each q-ary string $\boldsymbol{x} = x_1 x_2 \ldots x_n \in \mathbb{Z}_q^n$ can be represented by a $\lceil \log q \rceil \times n$ binary matrix

$$M_{\boldsymbol{x}} = (c_{i,j}) = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{\lceil \log q \rceil, 1} & \cdots & c_{\lceil \log q \rceil, n} \end{pmatrix}, \tag{1}$$

where $c_{i,j} \in \{0,1\}$, such that the jth column of $M_{\boldsymbol{x}}$ is the binary representation of x_j. Specifically, $x_j = \sum_{i=1}^{\lceil \log q \rceil} c_{i,j} 2^{i-1}$. We call $M_{\boldsymbol{x}}$ the *matrix representation* of $\boldsymbol{x}$. For any $i \in \{1, 2, \cdots, \lceil \log q \rceil\}$ and any interval $J = [j_1, j_2] = \{j_1, j_1 + 1, \cdots, j_2\} \subseteq [n]$, where $1 \leq j_1 < j_2 \leq n$, denote

$$c_{i,J} \triangleq c_{i,j_1} c_{i,j_1+1} \cdots c_{i,j_2}, \tag{2}$$

which is a substring of the ith row of $M_{\boldsymbol{x}}$ consisting of $c_{i,j_1}, c_{i,j_1+1}, \cdots, c_{i,j_2}$. In particular, $c_{i,[n]}$ is the ith row of $M_{\boldsymbol{x}}$.

Clearly, if $\boldsymbol{y} \in \mathbb{Z}_q^{n-t}$ is obtained from $\boldsymbol{x}$ by deleting $x_{j_1}, \cdots, x_{j_t}$, then the matrix representation $M_{\boldsymbol{y}}$ of $\boldsymbol{y}$ can be obtained from $M_{\boldsymbol{x}}$ by deleting columns $j_1, \cdots, j_t$ of $M_{\boldsymbol{x}}$. Moreover, $\boldsymbol{x}$ can be recovered from $\boldsymbol{y}$ if and only if its matrix representation $M_{\boldsymbol{x}}$ can be recovered from $M_{\boldsymbol{y}}$.

Lemma 8: Suppose $\mathcal{E}_0 : \{0,1\}^{n-1} \to \{0,1\}^n$ is a one-to-one mapping and $q > 2$ is an even integer. Then there is a one-to-one mapping $\bar{\mathcal{E}}_0 : \mathbb{Z}_q^{n-1} \to \mathbb{Z}_q^n$, with the same computing time as $\mathcal{E}_0$, such that for any $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$ and $\boldsymbol{x} = \bar{\mathcal{E}}_0(\boldsymbol{u})$, if $M_{\boldsymbol{u}} = (b_{i,j})_{\lceil \log q \rceil \times (n-1)}$ and $M_{\boldsymbol{x}} = (c_{i,j})_{\lceil \log q \rceil \times n}$ are the matrix representation of $\boldsymbol{u}$ and $\boldsymbol{x}$ respectively, then

$$c_{1,[n]} = \mathcal{E}_0(b_{1,[n-1]}).$$

Proof: For each $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$, where the matrix representation of $\boldsymbol{u}$ is

$$M_{\boldsymbol{u}} = \begin{pmatrix} b_{1,1} & \cdots & b_{1,n-1} \\ b_{2,1} & \cdots & b_{2,n-1} \\ \vdots & \ddots & \vdots \\ b_{\lceil \log q \rceil, 1} & \cdots & b_{\lceil \log q \rceil, n-1} \end{pmatrix},$$

---

[2]In [15], δ is taken to be $t2^{t+1}\lceil \log n \rceil^2$. In this paper, for notational simplicity, we omit the ceiling function and write $\delta = t2^{t+1}\log n$.

denote $\mathcal{E}_0(b_{1,[n-1]}) = \boldsymbol{c} = c_{1,1} \cdots c_{1,n-1} c_{1,n}$ and let

$$M = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n-1} & c_{1,n} \\ b_{2,1} & \cdots & b_{2,n-1} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ b_{\lceil \log q \rceil,1} & \cdots & b_{\lceil \log q \rceil,n-1} & 0 \end{pmatrix}.$$

Specifically, $M = (c_{i,j})$ is a $\lceil \log q \rceil \times n$ binary matrix satisfying the following three properties: i) the first row of M is equal to $\boldsymbol{c}$; ii) $c_{2,j} \cdots c_{\lceil \log q \rceil,j} = b_{2,j} \cdots b_{\lceil \log q \rceil,j}$ for each $j \in [n-1]$; iii) $c_{2,n} \cdots c_{\lceil \log q \rceil,n} = 0^{\lceil \log q \rceil - 1}$, where $0^{\lceil \log q \rceil - 1}$ is the string consisting of $\lceil \log q \rceil - 1$ symbol 0s.

Let $\bar{\mathcal{E}}_0(\boldsymbol{u}) = \boldsymbol{x}$ such that the matrix representation of $\boldsymbol{x}$ is $M_{\boldsymbol{x}} = M$. It is easy to see that $c_{1,[n]} = \mathcal{E}_0(b_{1,[n-1]})$ and the computing time of $\bar{\mathcal{E}}_0$ is the same as that of $\mathcal{E}_0$. Moreover, since $\mathcal{E}_0$ is a one-to-one mapping, it is also easy to see that $\bar{\mathcal{E}}_0$ is a one-to-one mapping.

It remains to prove that $\boldsymbol{x} \in \mathbb{Z}_q^n$, equivalently, each column of $M_{\boldsymbol{x}}$ is the binary representation of some integer in $\mathbb{Z}_q$.

According to property iii) of the constructed matrix M, we have $c_{\lceil \log q \rceil,n} \cdots c_{2,n} c_{1,n} = 0^{\lceil \log q \rceil - 1} c_{1,n}$, so the last column of M is the binary representation of $c_{1,n} \in \{0,1\} \subseteq \mathbb{Z}_q$. For each $j \in [n-1]$, according to property ii) of M, we have $c_{\lceil \log q \rceil,j} \cdots c_{2,j} c_{1,j} = b_{\lceil \log q \rceil,j} \cdots b_{2,j} c_{1,j}$, so $x_j = \sum_{i=1}^{\lceil \log q \rceil} c_{i,j} 2^{i-1} = \sum_{i=2}^{\lceil \log q \rceil} b_{i,j} 2^{i-1} + c_{1,j} = u_j - b_{1,j} + c_{1,j}$, where the last equality holds because according to the definition of the matrix representation, $b_{\lceil \log q \rceil,j} \cdots b_{2,j} b_{1,j}$ is the binary representation of u_j. If $b_{1,j} = 1$, then $x_j = u_j - b_{1,j} + c_{1,j} \leq u_j \leq q-1$. If $b_{1,j} = 0$, then u_j is even. Noticing that q is even, so $u_j \leq q-2$, and hence $x_j = u_j - b_{1,j} + c_{1,j} = u_j + c_{1,j} \leq q-1$. In both cases, we have $x_j \in \mathbb{Z}_q$. Thus, each column of M is the binary representation of some integer in $\mathbb{Z}_q$, and so $\boldsymbol{x} \in \mathbb{Z}_q^n$. ∎

III. Nonbinary Two-deletion Correcting Codes

In this section, we consider q-ary two-deletion correcting codes. We assume that $q > 2$ is an even integer and is a constant with respect to the code length n. Each binary sequence $\boldsymbol{a}$ will also be viewed as a non-negative integer whose binary representation is $\boldsymbol{a}$, and conversely, each non-negative integer m will also be viewed as a binary sequence with length $\lceil \log(m+1) \rceil$, i.e., the binary representation of m. Therefore, summation and multiplication of binary strings and integers are performed in the set of integers.

We need to introduce some concepts and notations for binary strings, which will be used in our construction.

Let $\boldsymbol{c} \in \{0,1\}^n$ be a binary string of length n. A *run* of $\boldsymbol{c}$ is a maximal substring of $\boldsymbol{c}$ consisting of identical symbols.[3] A substring $c_{[i_1,i_2]}$ of $\boldsymbol{c}$, where $i_1 < i_2$, is called an *alternative substring* of $\boldsymbol{c}$ if $c_{i+1} \neq c_i$ for all $i \in [i_1, i_2 - 1]$.

Remark 1: From Definition 1, it is easy to see that if $\boldsymbol{c} \in \{0,1\}^n$ is regular, then each substring of $\boldsymbol{c}$ of length $d \log n$ can not be a run or an alternative substring of $\boldsymbol{c}$ because it contains both 00 and 11. Equivalently, each run and each alternative substring of $\boldsymbol{c}$ have length at most $d \log n$.

Definition 2: For each $\boldsymbol{c} \in \{0,1\}^n$, let c_{I_i} be the ith run (counting from the left) of $\boldsymbol{c}$, where $I_i \subseteq [n]$ is the index set of c_{I_i}. Then we denote $\mathcal{I}_{\boldsymbol{c}} = \{c_{I_1}, \cdots, c_{I_{n'}}\}$ and call it *the set of runs* of $\boldsymbol{c}$, where n' is the number of runs of $\boldsymbol{c}$.

Let VT, ξ and η be the functions constructed by Lemma 1, Lemma 2 and Lemma 4, respectively. Denote

$$\rho = 3d \log n$$

and let

$$J_j = \begin{cases} [(j-1)\rho + 1, (j+1)\rho], & \text{for } j \in \{1, \cdots, \lceil n/\rho \rceil - 2\}, \\ [(j-1)\rho + 1, n], & \text{for } j = \lceil n/\rho \rceil - 1. \end{cases} \tag{3}$$

Note that each interval J_j has length 2ρ and the intersection of two successive intervals J_j and J_{j+1} is an interval of length ρ. It is easy to see the following remark.

Remark 2: The intervals J_j, $j = 1, \cdots, \lceil n/\rho \rceil - 1$ satisfies:

1) For any interval $J \subseteq [n]$ of length at most ρ, we can find an $j_0 \in \{1, 2, \cdots, \lceil n/\rho \rceil - 1\}$ such that $J \subseteq J_{j_0}$.
2) $J_j \cap J_{j'} = \emptyset$ for all $j, j' \in \{1, 2, \cdots, \lceil n/\rho \rceil - 1\}$ such that $|j - j'| \geq 2$.

For each q-ary string $\boldsymbol{x} \in \mathbb{Z}_q^n$, let $M_{\boldsymbol{x}} = (c_{i,j})$ be the matrix representation of $\boldsymbol{x}$ as defined by (1) and $c_{1,[n]}$ be the first row of $M_{\boldsymbol{x}}$. We construct a function f as follows.

Construction 1: For each $\boldsymbol{x} \in \mathbb{Z}_q^n$, let $\mathcal{I}_{\boldsymbol{c}} = \{c_{I_1}, \cdots, c_{I_{n'}}\}$ be the set of runs of $\boldsymbol{c} = c_{1,[n]}$ as defined in Definition 2. For each $i \in [n']$, let

$$g_i(\boldsymbol{x}) = \left(\text{VT}(c_{2,I_i}), \text{VT}(c_{3,I_i}), \cdots, \text{VT}(c_{\lceil \log q \rceil, I_i}) \right),$$

---

[3] We say that a substring of $\boldsymbol{c}$ satisfying a certain property is maximal if it is contained by no other substring of $\boldsymbol{c}$ that satisfies the same property. Hence, a maximal run of the string $\boldsymbol{c}$ is not contained by any other run of $\boldsymbol{c}$.

and for each $\ell \in \{0, 1\}$, let

$$g^{(\ell)}(\boldsymbol{x}) = \sum_{i=1}^{n'} i^\ell g_i(\boldsymbol{x}) \bmod 2n^\ell N_1, \qquad (4)$$

where

$$N_1 = q^{\log \log n + 3}.$$

Moreover, for each $j \in \{1, \cdots, \lceil n/\rho \rceil - 1\}$, let

$$h_j(\boldsymbol{x}) = \left(\xi(c_{2,J_j}), \xi(c_{3,J_j}), \cdots, \xi(c_{\lceil \log q \rceil, J_j}) \right),^4$$

and for each $\ell \in \{0, 1\}$, let

$$h^{(\ell)}(\boldsymbol{x}) = \sum_{\substack{j \in \{1, \cdots, \lceil n/\rho \rceil - 1\}: \\ j \equiv \ell \bmod 2}} h_j(\boldsymbol{x}) \bmod N_2 \qquad (5)$$

where

$$N_2 = q^{7 \log \log n + o(\log \log n)}.$$

Finally, let

$$f(\boldsymbol{x}) = \left(\eta(c_{1,[n]}), g^{(0)}(\boldsymbol{x}), g^{(1)}(\boldsymbol{x}), h^{(0)}(\boldsymbol{x}), h^{(1)}(\boldsymbol{x}) \right).$$

Let $\mathcal{R}_n$ denote the set of all $\boldsymbol{x} \in \mathbb{Z}_q^n$ such that $c_{1,[n]}$ is a regular string with $d = 7$ (according to Definition 1). Then we have the following Theorem.

Theorem 1: The function $f(\boldsymbol{x})$ is computable in linear time and the length $|f(\boldsymbol{x})|$ of $f(\boldsymbol{x})$ satisfies

$$|f(\boldsymbol{x})| \le 5 \log n + O(\log q \log \log n).$$

Moreover, if $\boldsymbol{x} \in \mathcal{R}_n$, then $\boldsymbol{x}$ can be uniquely recovered from $f(\boldsymbol{x})$ and any given $\boldsymbol{y} \in \mathcal{D}_2(\boldsymbol{x})$.

To prove Theorem 1, we need the following lemma.

Lemma 9: Suppose $\boldsymbol{c} \in \{0, 1\}^n$ is regular and $\boldsymbol{b} \in \{0, 1\}^{n-2}$ such that $\boldsymbol{b}$ can be obtained from $\boldsymbol{c}$ by deleting two symbols of $\boldsymbol{c}$. Then exact one of the following holds.

1) There are two distinct runs $c_{I_{j_1}}$ and $c_{I_{j_2}}$ of $\boldsymbol{c}$, uniquely determined by $\boldsymbol{b}$ and $\boldsymbol{c}$, such that $\boldsymbol{b}$ can be obtained from $\boldsymbol{c}$ by deleting one symbol in $c_{I_{j_1}}$ and one symbol in $c_{I_{j_2}}$.
2) There is an interval $J \subseteq [n]$ of length at most ρ such that $\boldsymbol{b}$ can be obtained from $\boldsymbol{c}$ by deleting two symbols in c_J.

Proof: This Lemma is proved in Appendix A. ∎

Now, we can prove Theorem 1.

Proof of Theorem 1: Note that by Lemma 1, Lemma 2 and Lemma 4, the functions VT, ξ and η are all computable in linear time. By Construction 1, the functions $g^{(\ell)}(\boldsymbol{x})$ and $h^{(\ell)}(\boldsymbol{x})$, $\ell \in \{0, 1\}$, are computable in linear time. Hence, $f(\boldsymbol{x}) = \left(\eta(c_{1,[n]}), g^{(0)}(\boldsymbol{x}), g^{(1)}(\boldsymbol{x}), h^{(0)}(\boldsymbol{x}), h^{(1)}(\boldsymbol{x}) \right)$ is computable in linear time.

For each $\boldsymbol{x} \in \mathbb{Z}_q^n$, by Construction 1, the length $|g^{(\ell)}(\boldsymbol{x})|$ of $g^{(\ell)}(\boldsymbol{x})$, $\ell \in \{0, 1\}$, satisfies

$$|g^{(\ell)}(\boldsymbol{x})| \le \log(2n^\ell N_1)$$
$$= \ell \log n + \log q \log \log n + 1$$

Similarly, the length $|h^{(\ell)}(\boldsymbol{x})|$ of $h^{(\ell)}(\boldsymbol{x})$, $\ell \in \{0, 1\}$, satisfies

$$|h^{(\ell)}(\boldsymbol{x})| \le \log N_2$$
$$= \log q (7 \log \log n + o(\log \log n))$$

Moreover, by Lemma 4, the length of $\eta(c_{1,[n]})$ satisfies

$$|\eta(c_{1,[n]})| \le 4 \log n + 10 \log \log n + O(1).$$

---

[4]Note that Lemma 2 requires that each $|I_j| \ge 3$. If $|I_j| < 3$, we can just let $\xi(c_{i,I_j}) = c_{i,I_j}$. Then c_{i,I_j} can also be recovered from $\xi(c_{i,I_j})$. This is feasible because in our construction, we only need that each $\xi(c_{i,I_j})$ is a sequence of length not greater than $7 \log \log n + o(\log \log n)$.

Thus, by Construction 1, the length of $f(\boldsymbol{x})$ satisfies

$$
\begin{aligned}
|f(\boldsymbol{x})| &= |\eta(c_{1,[n]})| + |g^{(0)}(\boldsymbol{x})| + |g^{(1)}(\boldsymbol{x})| \\
&\quad + |h^{(0)}(\boldsymbol{x})| + |h^{(1)}(\boldsymbol{x})| \\
&\leq 5\log n + (16\log q + 10)\log\log n \\
&\quad + o(\log q \log\log n) \\
&= 5\log n + O(\log q \log\log n).
\end{aligned}
$$

It remains to prove that for each $\boldsymbol{x} \in \mathcal{R}_n$, given $f(\boldsymbol{x})$ and any $\boldsymbol{y} \in \mathcal{D}_2(\boldsymbol{x})$, one can uniquely recover $\boldsymbol{x}$. To prove this, we first prove that $g_j(\boldsymbol{x}) < N_1$ for each $j \in [n']$ and $h_j(\boldsymbol{x}) < N_2$ for each $j \in \{1, 2, \cdots, \lceil n/\rho \rceil - 1\}$.

Since $\boldsymbol{x} \in \mathcal{R}_n$, by Remark 1, each run of $c_{1,[n]}$ has length at most $7\log n$ (noticing that we take $d = 7$ in this paper), so for each $i \in [2, \lceil \log q \rceil]$ and $j \in [n']$, we have $c_{i,I_j} \in \{0,1\}^{\leq 7\log n}$. By Lemma 1, for each $j \in [2, \lceil \log q \rceil]$, the length of $\mathrm{VT}(c_{j,I_i})$ satisfies $|\mathrm{VT}(c_{j,I_i})| \leq \log(7\log n) \leq \log\log n + 3$. By Construction 1, the length of $g_j(\boldsymbol{x})$ satisfies

$$
\begin{aligned}
|g_j(\boldsymbol{x})| &\leq (\lceil \log q \rceil - 1)(\log\log n + 3) \\
&< (\log q)(\log\log n + 3),
\end{aligned}
$$

so

$$
g_j(\boldsymbol{x}) < q^{\log\log n + 3} = N_1.
$$

Similarly, for each $i \in [2, \lceil \log q \rceil]$ and each $j \in \{1, 2, \cdots, \lceil n/\rho \rceil - 1\}$, by (3), the length of the interval c_{i,J_j} satisfies $|c_{i,J_j}| \leq 2\rho = 42\log n$, so by Lemma 2, we have

$$
\begin{aligned}
|\xi(c_{i,I_j})| &\leq 7\log(42\log n) + o(\log(42\log n)) \\
&= 7\log\log n + o(\log\log n).
\end{aligned}
$$

By Construction 1,

$$
\begin{aligned}
|h_j(\boldsymbol{x})| &\leq (\lceil \log q \rceil - 1)(7\log\log n + o(\log\log n)) \\
&< (\log q)(7\log\log n + o(\log\log n)).
\end{aligned}
$$

Hence,

$$
h_j(\boldsymbol{x}) < q^{7\log\log n + o(\log\log n)} = N_2.
$$

Now, we prove that each $\boldsymbol{x} \in \mathcal{R}_n$ can be uniquely recovered from $f(\boldsymbol{x}) = \big(\eta(c_{1,[n]}), g^{(0)}(\boldsymbol{x}), g^{(1)}(\boldsymbol{x}), h^{(0)}(\boldsymbol{x}), h^{(1)}(\boldsymbol{x})\big)$ and any given $\boldsymbol{y} \in \mathcal{D}_2(\boldsymbol{x})$. Let

$$
M_{\boldsymbol{y}} = (d_{i,j})_{\lceil \log q \rceil \times (n-2)}
$$

be the matrix representation of $\boldsymbol{y}$. Then $M_{\boldsymbol{y}}$ can be obtained from $M_{\boldsymbol{x}}$ by deleting two columns, so $d_{1,[n-2]} \in \mathcal{D}_2(c_{1,[n]})$, where $d_{1,[n-2]}$ is the first row of $M_{\boldsymbol{y}}$. Since $\boldsymbol{x} \in \mathcal{R}_n$, then $c_{1,[n]}$ is regular. By Lemma 4, $\boldsymbol{c} \triangleq c_{1,[n]}$ can be correctly recovered from $\boldsymbol{d} \triangleq d_{1,[n-2]}$ and $\eta(c_{1,[n]})$. Moreover, by Lemma 9, exact one of the following two cases holds:

Case 1: There are two distinct runs $c_{1,I_{j_1}}$ and $c_{1,I_{j_2}}$ of $c_{1,[n]}$ such that $d_{1,[n-2]}$ is obtained from $c_{1,[n]}$ by deleting one symbol in $c_{1,I_{j_1}}$ and one symbol $c_{1,I_{j_2}}$. Correspondingly, $M_{\boldsymbol{y}}$ can be obtained from $M_{\boldsymbol{x}}$ by deleting one column in I_{j_1} and one column in I_{j_2}. Without loss of generality, assume $j_1 < j_2$. Denoting $I_j = [p_{j-1} + 1, p_j]$, where $p_0 = 0 < p_1 < p_2 < \cdots < p_{n'} = n$, then by comparing the symbols of $M_{\boldsymbol{x}}$ and $M_{\boldsymbol{y}}$, we have the following observation:

i) $c_{i,I_j} = d_{i,I_j}$ for $1 \leq j < j_1$ and each $i \in [2, \lceil \log q \rceil]$;
ii) $d_{i,[p_{j_1-1}+1,p_{j_1}-1]} \in \mathcal{D}_1(c_{i,I_{j_1}})$ for each $i \in [2, \lceil \log q \rceil]$;
iii) $c_{i,I_j} = d_{i,I_j-1}$ for each $j_1 < j < j_2$ and $i \in [2, \lceil \log q \rceil]$, where $I_j - 1 = \{\ell - 1 : \ell \in I_j\} = [p_{j-1}, p_j - 1]$;
iv) $d_{i,[p_{j_2-1},p_{j_2}-2]} \in \mathcal{D}_1(c_{i,I_{j_2}})$ for each $i \in [2, \lceil \log q \rceil]$;
v) $c_{i,I_j} = d_{i,I_j-2}$ for each $j_2 < j \leq n'$ and $i \in [2, \lceil \log q \rceil]$, where $I_j - 2 = \{\ell - 2 : \ell \in I_j\} = [p_{j-1} - 1, p_j - 2]$.

Then $c_{i,[n]}$, $i \in [2, \lceil \log q \rceil]$, can be recovered by the following three steps:

Step 1: By observations i), iii) and v), c_{i,I_j} can be directly obtained from $M_{\boldsymbol{y}}$ for all $i \in [2, \lceil \log q \rceil]$ and $j \in [n'] \backslash \{j_1, j_2\}$.

Step 2: Compute $g_j(\boldsymbol{x})$ from $c_{2,I_j}, \cdots, c_{\lceil \log q \rceil, I_j}$ for all $j \in [n'] \backslash \{j_1, j_2\}$. This is possible because for all $i \in [2, \lceil \log q \rceil]$ and $j \in [n'] \backslash \{j_1, j_2\}$, c_{i,I_j} have been obtained from $M_{\boldsymbol{y}}$ in Step 1. Then by taking $\ell = 0$ in (4), we can obtain

$$
g_{j_1}(\boldsymbol{x}) + g_{j_2}(\boldsymbol{x}) \equiv \Bigg(g^{(0)}(\boldsymbol{x}) - \sum_{j \in [n'] \backslash \{j_1, j_2\}} g_j(\boldsymbol{x})\Bigg) \bmod 2N_1.
$$

Since $g_j(\boldsymbol{x}) < N_1$ for all $j \in [n']$, we in fact have

$$g_{j_1}(\boldsymbol{x}) + g_{j_2}(\boldsymbol{x}) = \left(g^{(0)}(\boldsymbol{x}) - \sum_{j \in [n'] \setminus \{j_1, j_2\}} g_j(\boldsymbol{x}) \right) \bmod 2N_1. \tag{6}$$

Similarly, taking $\ell = 1$ in (4) and noticing that $0 < j_1 g_{j_1}(\boldsymbol{x}) + j_2 g_{j_2}(\boldsymbol{x}) < 2nN_1$, we can obtain

$$j_1 g_{j_1}(\boldsymbol{x}) + j_2 g_{j_2}(\boldsymbol{x})$$
$$= \left(g^{(1)}(\boldsymbol{x}) - \sum_{j \in [n'] \setminus \{j_1, j_2\}} j g_j(\boldsymbol{x}) \right) \bmod 2nN_1. \tag{7}$$

So, $g_{j_1}(\boldsymbol{x})$ and $g_{j_2}(\boldsymbol{x})$ can be solved from (6) and (7). By Construction 1, we have

$$g_{j_1}(\boldsymbol{x}) = \left(\mathrm{VT}(c_{2,I_{j_1}}), \mathrm{VT}(c_{3,I_{j_1}}), \cdots, \mathrm{VT}(c_{\lceil \log q \rceil, I_{j_1}}) \right)$$

and

$$g_{j_2}(\boldsymbol{x}) = \left(\mathrm{VT}(c_{2,I_{j_2}}), \mathrm{VT}(c_{3,I_{j_2}}), \cdots, \mathrm{VT}(c_{\lceil \log q \rceil, I_{j_2}}) \right).$$

Step 3: By observation ii), $d_{i,[p_{j_1}-1+1,p_{j_1}-1]} \in \mathcal{D}_1(c_{i,I_{j_1}})$ for each $i \in [2, \lceil \log q \rceil]$. Then by Lemma 1, $c_{i,I_{j_1}}$ can be recovered from $\mathrm{VT}(c_{i,I_{j_1}})$ and $d_{i,[p_{j_1}-1+1,p_{j_1}-1]}$. Similarly, since by observation iv), $d_{i,[p_{j_2}-1,p_{j_2}-2]} \in \mathcal{D}_1(c_{i,I_{j_2}})$ for each $i \in [2, \lceil \log q \rceil]$, then by Lemma 1, $c_{i,I_{j_2}}$ can be recovered from $\mathrm{VT}(c_{i,I_{j_2}})$ and $d_{i,[p_{j_2}-1,p_{j_2}-2]}$.

Thus, for case 1, $c_{i,[n]}$, $i \in [2, \lceil \log q \rceil]$, can be recovered from $\eta(c_{1,[n]})$, $g^{(0)}(\boldsymbol{x})$, $g^{(1)}(\boldsymbol{x})$ and $\boldsymbol{y}$.

Case 2: There is an interval $J \subseteq [n]$ of length at most $\rho = 3d \log n$ such that $\boldsymbol{d} \triangleq d_{1,[n-2]}$ can be obtained from $\boldsymbol{c} \triangleq c_{1,[n]}$ by deleting two symbols in $c_{1,J}$. Correspondingly, $M_{\boldsymbol{y}}$ can be obtained from $M_{\boldsymbol{x}}$ by deleting two columns in J. By 1) of Remark 2, we can always find an J_{j_0} for some $j_0 \in \{1, 2, \cdots, \lceil n/\rho \rceil - 1\}$ such that $J \subseteq J_{j_0}$. Denoting $J_{j_0} = [\lambda, \lambda']$, then by comparing the symbols of $M_{\boldsymbol{x}}$ and $M_{\boldsymbol{y}}$, we obtain that for each $i \in [2, \lceil \log q \rceil]$,

$$c_{i,[1,\lambda-1]} = d_{i,[1,\lambda-1]},$$

$$c_{i,[\lambda'+1,n]} = d_{i,[\lambda'-1,n-2]}$$

and

$$d_{i,[\lambda,\lambda'-2]} \in \mathcal{D}_{\leq 2}(c_{i,[\lambda,\lambda']}).$$

Hence, $c_{i,[1,\lambda-1]}$ and $c_{i,[\lambda'+1,n]}$ can be directly obtained from $M_{\boldsymbol{y}}$. Moreover, each $c_{i,[\lambda,\lambda']}$ can be recovered from $d_{i,[\lambda,\lambda'-2]}$ and $h^{(\ell)}(\boldsymbol{x})$, $\ell \in \{0, 1\}$, as follows.

By 2) of Remark 2, $J_j \subseteq [1, \lambda-1]$ for all $j \in \{1, 2, \cdots, j_0 - 2\}$, so c_{i,J_j} can be obtained from $d_{i,[1,\lambda-1]} = c_{i,[1,\lambda-1]}$. Similarly, c_{i,J_j} can be obtained from $d_{i,[\lambda'+1-t,n]} = c_{i,[\lambda'+1,n]}$ for all $j \in \{j_0 + 2, \cdots, \lceil n/\delta' \rceil - 1\}$. Hence, we can compute $h_j(\boldsymbol{x}) = \left(\xi(c_{2,J_j}), \xi(c_{3,J_j}), \cdots, \xi(c_{\lceil \log q \rceil, J_j}) \right)$ for each $j \in \{1, 2, \cdots, \lceil n/\delta' \rceil - 1\} \setminus \{j_0\}$. Let $\ell \in \{0, 1\}$ be such that $j_0 \equiv \ell \bmod 2$. Then by (5), and noticing that $h_{j_0}(\boldsymbol{x}) < N_2$, we can obtain

$$h_{j_0}(\boldsymbol{x}) = h^{(\ell)}(\boldsymbol{x}) - \sum_{\substack{j \in \{1, 2, \cdots, \lceil n/\delta' \rceil - 1\} \setminus \{j_0\}: \\ j \equiv \ell \bmod 2}} h_j(\boldsymbol{x}) \bmod N_2.$$

Note that $b_{i,[\lambda,\lambda'-2]} \in \mathcal{D}_{\leq 2}(c_{i,[\lambda,\lambda']}) = \mathcal{D}_{\leq 2}(c_{i,J_{j_0}})$, and by Construction 1,

$$h_{j_0}(\boldsymbol{x}) = \left(\xi(c_{2,J_{j_0}}), \xi(c_{3,J_{j_0}}), \cdots, \xi(c_{\lceil \log q \rceil, J_{j_0}}) \right).$$

Then by Lemma 2, for each $i \in [2, \lceil \log q \rceil]$, $c_{i,[\lambda,\lambda']} = c_{i,J_{j_0}}$ can be recovered from $d_{i,[\lambda,\lambda'-2]}$ and $h_{j_0}(\boldsymbol{x})$.

Thus, for case 2, $c_{i,[n]}$, $i \in [2, \lceil \log q \rceil]$, can be recovered from $\eta(c_{1,[n]})$, $h^{(\ell)}(\boldsymbol{x})$ and $\boldsymbol{y}$.

By the above discussions, we proved that $M_{\boldsymbol{x}}$ (and so $\boldsymbol{x}$) can be uniquely recovered from $f(\boldsymbol{x})$ and $\boldsymbol{y}$, which completes the proof. ∎

By representing each binary string of length at most $\lfloor \log q \rfloor$ as an integer in $\mathbb{Z}_q$, each binary string $\boldsymbol{a}$ can be represented as a q-ary string of length $\lceil |\boldsymbol{a}|/\lfloor \log q \rfloor \rceil$. We denote this q-ary string by $\mathcal{Q}(\boldsymbol{a})$ and call it the *q-ary representation* of $\boldsymbol{a}$ for convenience of use. Specifically, divide $\boldsymbol{a}$ into $\lceil |\boldsymbol{a}|/\lfloor \log q \rfloor \rceil$ disjoint substrings, each having length $\lfloor \log q \rfloor$ except the last substring which has length $|\boldsymbol{a}| - (\lceil |\boldsymbol{a}|/\lfloor \log q \rfloor \rceil - 1) \lfloor \log q \rfloor$. Then by representing each of these substrings as an integer in $\mathbb{Z}_q$, we can obtain a q-ary string $\mathcal{Q}(\boldsymbol{a})$ of length $\lceil |\boldsymbol{a}|/\lfloor \log q \rfloor \rceil$.

Let RegEnc : $\{1, 2, \cdots, M\} \to \{0, 1\}^n$ be the one-to-one mapping constructed in Lemma 3. Since $M \geq 2^{n-1}$, then RegEnc can also be viewed as a mapping from $\{0, 1\}^{n-1}$ to $\{0, 1\}^n$. By Lemma 8, the mapping RegEnc can be extended to a one-to-one mapping, denoted by

$$\mathcal{E}_{\text{Reg}} : \mathbb{Z}_q^{n-1} \to \mathbb{Z}_q^n,$$

such that for any $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$ and $\boldsymbol{x} = \mathcal{E}_{\text{Reg}}(\boldsymbol{u})$, if $M_{\boldsymbol{u}} = (b_{i,j})_{\lceil \log q \rceil \times (n-1)}$ and $M_{\boldsymbol{x}} = (c_{i,j})_{\lceil \log q \rceil \times n}$ are the matrix representation of $\boldsymbol{u}$ and $\boldsymbol{x}$ respectively, then

$$c_{1,[n]} = \text{RegEnc}(b_{1,[n-1]}).$$

By Lemma 3, $c_{1,[n]} = \text{RegEnc}(b_{1,[n-1]})$ is regular, so for any $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$, we have $\boldsymbol{x} = \mathcal{E}_{\text{Reg}}(\boldsymbol{u}) \in \mathcal{R}_n$.

Using the mapping $\mathcal{E}_{\text{Reg}} : \mathbb{Z}_q^{n-1} \to \mathcal{R}_n$ and the function f constructed in Construction 1, we can give an encoding function of a q-ary two-deletion correcting code as follows.

Let $\mathcal{E}$ be the function defined on $\mathbb{Z}_q^{n-1}$ of the form

$$\mathcal{E}(\boldsymbol{u}) = (\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}''), \ \forall \, \boldsymbol{u} \in \mathbb{Z}_q^{n-1}, \tag{8}$$

such that $\boldsymbol{v} = \mathcal{E}_{\text{Reg}}(\boldsymbol{u})$, $\boldsymbol{v}' = \mathcal{E}_{\text{Reg}}(\mathcal{Q}(f(\boldsymbol{v})))$ and $\boldsymbol{v}'' = \text{Rep}_3(\mathcal{Q}(f(\boldsymbol{v}')))$, where $\text{Rep}_3(\cdot)$ is the encoding function of the 3-fold repetition code.

Theorem 2: The code $\mathcal{C} = \{\mathcal{E}(\boldsymbol{u}) : \boldsymbol{u} \in \mathbb{Z}_q^{n-1}\}$, where $\mathcal{E}$ is given by (8), is a q-ary two-deletion correcting code with redundancy $5 \log n + O(\log q \log \log n)$ in bits. The encoding complexity of $\mathcal{C}$ is near-linear in n with a polynomial size lookup table.

Proof: Let

$$\boldsymbol{x} = \mathcal{E}(\boldsymbol{u}) = (\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}'') \in \mathcal{C},$$

where $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$, $\boldsymbol{v} = \mathcal{E}_{\text{Reg}}(\boldsymbol{u})$, $\boldsymbol{v}' = \mathcal{Q}(f(\boldsymbol{v}))$ and $\boldsymbol{v}'' = \text{Rep}_3(\mathcal{Q}(f(\boldsymbol{v}')))$ as in (8). Given any $\boldsymbol{y} \in \mathcal{D}_2(\boldsymbol{x})$, we have $y_{[1,m_1-2]} \in \mathcal{D}_2(\boldsymbol{v})$, $y_{[m_1,m_2-2]} \in \mathcal{D}_2(\boldsymbol{v}')$ and $y_{[m_2,m_3-2]} \in \mathcal{D}_2(\boldsymbol{v}'')$, where $|\boldsymbol{v}| = m_1, |(\boldsymbol{v}, \boldsymbol{v}')| = m_2$ and $|\boldsymbol{x}| = |(\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}'')| = m_3$. First, since $\boldsymbol{v}'' = \text{Rep}_3(\mathcal{Q}(f(\boldsymbol{v}')))$ is a codeword of a two-deletion code, then $\mathcal{Q}(f(\boldsymbol{v}'))$ can be recovered from $y_{[m_2,m_3-2]}$, and hence $f(\boldsymbol{v}')$ can be recovered from $\mathcal{Q}(f(\boldsymbol{v}'))$. Then by Theorem 1, $\boldsymbol{v}'$ can be recovered from $y_{[m_1,m_2-2]}$ and $f(\boldsymbol{v}')$, and hence $f(\boldsymbol{v})$ can be recovered from $\boldsymbol{v}' = \mathcal{E}_{\text{Reg}}(\mathcal{Q}(f(\boldsymbol{v})))$. Finally, by Theorem 1 again, $\boldsymbol{v}$ can be recovered from $y_{[1,m_1-2]}$ and $f(\boldsymbol{v})$. Thus, $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}'')$ can be recovered from any $\boldsymbol{y} \in \mathcal{D}_2(\boldsymbol{x})$, which proves that $\mathcal{C}$ is a two-deletion correcting code.

Since $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$ and $\boldsymbol{v} = \mathcal{E}_{\text{Reg}}(\boldsymbol{u}) \in \mathbb{Z}_q^n$, so $\boldsymbol{v}$ has $\log q$ bits redundancy. Moreover, by Theorem 1, the length of $\boldsymbol{v}'$ is

$$|\boldsymbol{v}'| = 5 \log n + O(\log q \log \log n)$$

bits and the length of $\boldsymbol{v}''$ is

$$|\boldsymbol{v}''| = 3(5 \log |\boldsymbol{v}'| + O(\log q \log \log |\boldsymbol{v}'|)) = O(\log \log n)$$

bits. So the total redundancy of $\boldsymbol{x} = \mathcal{E}(\boldsymbol{u})$ is

$$\text{redundancy of } \bar{\mathcal{C}} = \log q + |\boldsymbol{v}'| + |\boldsymbol{v}''|$$
$$= 5 \log n + O(\log q \log \log n)$$

in bits.

By Lemma 3 and Lemma 8, the encoding complexity of $\boldsymbol{v} = \mathcal{E}_{\text{Reg}}(\boldsymbol{u})$ is near-linear in n with a polynomial size lookup table. Moreover, by Theorem 1, the encoding complexity of $\boldsymbol{v}' = \mathcal{E}_{\text{Reg}}(\mathcal{Q}(f(\boldsymbol{v})))$ and $\boldsymbol{v}'' = \text{Rep}_3(\mathcal{Q}(f(\boldsymbol{v}')))$ is linear in n and $\log n$ respectively. Therefore, the encoding complexity of $\mathcal{E}(\boldsymbol{u}) = (\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}'')$ is near-linear in n with a polynomial size lookup table, which completes the proof. ∎

IV. Binary Codes Correcting a Burst of at most t Deletions

In this section, we present a construction of binary codes that are capable of correcting a bursting of at most t deletions improving the Lenz-Polyanskii Construction in [15]. We assume that t is a constant with respect to the code length n, and for notational simplicity, we use γ_t to denote any constant that depends only on t. As in Section III, each binary sequence $\boldsymbol{a}$ is identified with the positive integer whose binary representation is $\boldsymbol{a}$, and summation and multiplication of binary strings and integers are performed in the set of integers.

Recall that a string $\boldsymbol{c} \in \{0, 1\}^n$ is called $(\boldsymbol{p}, \delta)$-dense, if each substring of $\boldsymbol{c}$ of length δ contains at least one pattern $\boldsymbol{p}$. As stated in Section II, we take

$$\delta = t2^{t+1} \log n$$

and

$$\boldsymbol{p} = 0^t 1^t.$$

The basic idea of our construction is to replace the shifted VT code in the Lenz-Polyanskii Construction with the function ϕ constructed in Lemma 5. To apply the function ϕ, we need to divide each binary string into substrings of length at most $2(\delta + t)$. Specifically, we denote $\delta' = \delta + t$ and let

$$L_i = \begin{cases} [(i-1)\delta'+1, (i+1)\delta'], & \text{for } i \in \{1, \cdots, \lceil n/\delta' \rceil - 2\}, \\ [(i-1)\delta'+1, n], & \text{for } i = \lceil n/\delta' \rceil - 1, \end{cases} \tag{9}$$

where $i \in \{1, \cdots, \lceil n/\delta' \rceil - 1\}$, be the index sets of the expected substrings. Then we can construct a function $\bar{f}^{\text{b}}$ over $\{0,1\}^n$ as follows, which is the main component of our construction of binary burst-deletion correcting codes.

Construction 2: Let ϕ and μ be the functions constructed in Lemma 5 and Lemma 7 respectively. For each $\boldsymbol{c} \in \{0,1\}^n$, let

$$\bar{f}^{\text{b}}(\boldsymbol{c}) = \left(\mu(\boldsymbol{c}), \bar{g}^{(0)}(\boldsymbol{c}), \bar{g}^{(1)}(\boldsymbol{c}) \right),$$

such that for each $\ell \in \{0,1\}$,

$$\bar{g}^{(\ell)}(\boldsymbol{c}) = \sum_{\substack{i \in \{1,2,\cdots,\lceil n/\lceil \delta' \rceil \rceil - 1\}: \\ i \equiv \ell \bmod 2}} \phi(c_{L_i}) \bmod \overline{N}^{\text{b}}, \tag{10}$$

where

$$\overline{N}^{\text{b}} \triangleq 2^{4 \log(2\delta') + o(\log(2\delta'))} = 2^{4 \log \log n + \gamma_t + o(\log \log n)}.^5$$

For Construction 2, we have the following theorem.

Theorem 3: For each $\boldsymbol{c} \in \{0,1\}^n$, $\bar{f}^{\text{b}}(\boldsymbol{c})$ is computable in linear time and the length $|\bar{f}^{\text{b}}(\boldsymbol{c})|$ of $\bar{f}^{\text{b}}(\boldsymbol{c})$ satisfies

$$|\bar{f}^{\text{b}}(\boldsymbol{c})| \le \log n + 8 \log \log n + \gamma_t + o(\log \log n).$$

Moreover, if $\boldsymbol{c}$ is $(\boldsymbol{p}, \delta)$-dense, then given $\bar{f}^{\text{b}}(\boldsymbol{c})$ and any $\boldsymbol{b} \in \mathcal{B}_{\le t}(\boldsymbol{c})$, one can uniquely recover $\boldsymbol{c}$.

Before proving Theorem 3, we give some remark on the properties of the sets $L_j, j = 1, 2, \cdots, \lceil n/\delta' \rceil - 1$.

Remark 3: Similar to Remark 2, it is easy to see that

1) For each interval $L \subseteq [n]$ of length at most $\delta' = \delta + t$, we can always find an $i_0 \in \{1, 2, \cdots, \lceil n/\delta' \rceil - 1\}$ such that $L \subseteq L_{i_0}$.
2) $L_i \cap L_{i'} = \emptyset$ for all $i, i' \in \{1, 2, \cdots, \lceil n/\delta' \rceil - 1\}$ such that $|i - i'| \ge 2$.

Now, we can prove Theorem 3.

Proof: Note that by Lemma 7, $\mu(\boldsymbol{c})$ is computable in linear time. By Lemma 5, each $\phi(c_{L_i})$ is computable in time $O(2^t(2\delta)^3) = O((\log n)^3)$, so $(\bar{g}^{(0)}(\boldsymbol{c}), \bar{g}^{(1)}(\boldsymbol{c}))$ are also computable in linear time. Hence, by Construction 2, $\bar{f}^{\text{b}}(\boldsymbol{c}) = \left(\mu(\boldsymbol{c}), \bar{g}^{(0)}(\boldsymbol{c}), \bar{g}^{(1)}(\boldsymbol{c}) \right)$ is computable in linear time. Moreover, by Lemma 7 and (10), the length $|\bar{f}^{\text{b}}(\boldsymbol{c})|$ of $\bar{f}^{\text{b}}(\boldsymbol{c})$ satisfies

$$\begin{aligned} |\bar{f}^{\text{b}}(\boldsymbol{c})| &= |\mu(\boldsymbol{c})| + |\bar{g}^{(0)}(\boldsymbol{c})| + |\bar{g}^{(1)}(\boldsymbol{c})| \\ &\le \log n + 3 + 2\left(4 \log \log n + \gamma_t + o(\log \log n)\right) \\ &= \log n + 8 \log \log n + \gamma_t + o(\log \log n). \end{aligned}$$

Suppose $\boldsymbol{c}$ is $(\boldsymbol{p}, \delta)$-dense and $\boldsymbol{b} \in \mathcal{B}_{\le t}(\boldsymbol{c})$. We need to prove that $\boldsymbol{c}$ can be uniquely recovered from $\boldsymbol{b}$ and $\bar{f}^{\text{b}}(\boldsymbol{c})$.

By Lemma 7, we can find an interval $L \subseteq [n]$ of length at most $\delta' = \delta + t$ such that $\boldsymbol{b} = c_{[n] \setminus D}$ for some interval $D \subseteq L$ of length $t' = |\boldsymbol{c}| - |\boldsymbol{b}|$. By 1) of Remark 3, we can always find an $i_0 \in \{1, 2, \cdots, \lceil n/\delta' \rceil - 1\}$ such that $L \subseteq L_{i_0}$. Denoting $L_{i_0} = [\lambda, \lambda']$, then we can obtain

$$c_{[1,\lambda-1]} = b_{[1,\lambda-1]},$$

$$c_{[\lambda'+1,n]} = b_{[\lambda'-t'+1,n]}$$

and

$$b_{[\lambda,\lambda'-t']} \in \mathcal{B}_{\le t}(c_{[\lambda,\lambda']}).$$

Therefore, $c_{[1,\lambda-1]}$ and $c_{[\lambda'+1,n]}$ can be directly obtained from $\boldsymbol{b}$. In the following, we will show how to recover $c_{[\lambda,\lambda']}$ from $b_{[\lambda,\lambda'-t']}$ and $\bar{g}^{(\ell)}(\boldsymbol{c})$ for some $\ell \in \{0,1\}$.

---

[5] Since $\delta' = \delta + t = t2^{t+1}(\log n + 2^{-t-1})$, so more accurately, it should be $\overline{N}^{\text{b}} \triangleq 2^{4 \log(2\delta') + o(\log(2\delta'))} = 2^{4 \log(\log n + 2^{-t-1}) + \gamma_t + o(\log \log n)}$. However, because $\overline{N}^{\text{b}}$ is an integer, so for sufficiently large n, we can always obtain $\overline{N}^{\text{b}} \triangleq 2^{4 \log(2\delta') + o(\log(2\delta'))} = 2^{4 \log \log n + \gamma_t + o(\log \log n)}$.

By 2) of Remark 3, for all $i \in \{1, 2, \cdots, i_0 - 2\}$, we have $L_i \subseteq [1, \lambda - 1]$, so c_{L_i} can be obtained from $b_{[1, \lambda - 1]}$ and hence $\phi(c_{L_i})$ can be computed. Similarly, for all $i \in \{i_0 + 2, \cdots, \lceil n/\delta' \rceil - 1\}$, c_{L_i} can be obtained from $b_{[\lambda' - t' + 1, n]}$ and hence $\phi(c_{L_i})$ can be computed. Let $\ell_0 \in \{0, 1\}$ be such that $\ell_0 \equiv i_0 \mod 2$. By (10), we have

$$\phi(c_{L_{i_0}}) \equiv \bar{g}^{(\ell_0)}(\boldsymbol{c}) - \sum_{\substack{i \in \{1, \cdots, \lceil n/\lceil \delta' \rceil \rceil - 1\}: \\ i \neq i_0 \text{ and } i \equiv \ell_0 \mod 2}} \phi(c_{L_i}) \mod \overline{N}^{\text{b}}.$$

By (9), $|L_{i_0}| = 2\delta' = 2(\delta + t)$, so by Lemma 5, $\phi(c_{L_{i_0}}) \leq 2^{4 \log(2\delta') + o(\log(2\delta'))} = \overline{N}^{\text{b}}$. Therefore, we actually have

$$\phi(c_{L_{i_0}}) = \bar{g}^{(\ell_0)}(\boldsymbol{c}) - \sum_{\substack{i \in \{1, \cdots, \lceil n/\lceil \delta' \rceil \rceil - 1\}: \\ i \neq i_0 \text{ and } i \equiv \ell_0 \mod 2}} \phi(c_{L_i}) \mod \overline{N}^{\text{b}}.$$

Since $b_{[\lambda, \lambda' - t']} \in \mathcal{B}_{\leq t}(c_{[\lambda, \lambda']})$, again by Lemma 5, we can recover $c_{[\lambda, \lambda']}$ from $b_{[\lambda, \lambda' - t]}$ and $\phi(c_{L_{i_0}})$. Note that we have obtained $c_{[1, \lambda - 1]} = b_{[1, \lambda - 1]}$ and $c_{[\lambda' + 1, n]} = b_{[\lambda' - t + 1, n]}$, so $\boldsymbol{c}$ can be uniquely recovered, which completes the proof. $\blacksquare$

Let $\mathcal{S}_n^{\text{b}}$ be the set of all $(\boldsymbol{p}, \delta)$-dense binary strings $\boldsymbol{c} \in \{0, 1\}^n$, where $\boldsymbol{p} = 0^t 1^t$ and $\delta = t2^{t+1}\lceil \log n \rceil$. By Lemma 6, there is a one-to-one mapping that maps each binary string of length $n - 1$ to a string in $\mathcal{S}_n^{\text{b}}$. For convenience, we denote this mapping by

$$\mathcal{E}_{\text{Den}}^{\text{b}} : \{0, 1\}^{n-1} \rightarrow \mathcal{S}_n^{\text{b}}. \tag{11}$$

Using the function $\bar{f}^{\text{b}}$ constructed in Construction 2, we can construct an encoding function of a binary code capable of correcting a burst of at most t deletions.

Let $\bar{\mathcal{E}}^{\text{b}}$ be a function defined on $\{0, 1\}^{n-1}$ of the form

$$\bar{\mathcal{E}}^{\text{b}}(\boldsymbol{a}) = (\boldsymbol{b}, \boldsymbol{b}', \boldsymbol{b}''), \quad \forall \boldsymbol{a} \in \{0, 1\}^{n-1}, \tag{12}$$

such that $\boldsymbol{b} = \mathcal{E}_{\text{Den}}^{\text{b}}(\boldsymbol{a})$, $\boldsymbol{b}' = \mathcal{E}_{\text{Den}}^{\text{b}}(\bar{f}^{\text{b}}(\boldsymbol{b}))$ and $\boldsymbol{b}'' = \text{Rep}_{t+1}(\bar{f}^{\text{b}}(\boldsymbol{b}'))$, where $\text{Rep}_{t+1}(\cdot)$ is the encoding function of the $(t+1)$-fold repetition code.

Theorem 4: The code $\bar{\mathcal{C}}^{\text{b}} = \{\bar{\mathcal{E}}^{\text{b}}(\boldsymbol{a}) : \boldsymbol{a} \in \{0, 1\}^{n-1}\}$, where $\bar{\mathcal{E}}^{\text{b}}$ is given by (12), is a binary code with redundancy $\log n + 9 \log \log n + \gamma_t + o(\log \log n)$ bits and capable of correcting a burst of at most t deletions.

Proof: Let

$$\boldsymbol{c} = \bar{\mathcal{E}}^{\text{b}}(\boldsymbol{a}) = (\boldsymbol{b}, \boldsymbol{b}', \boldsymbol{b}'') \in \bar{\mathcal{C}}^{\text{b}},$$

where $\boldsymbol{a} \in \{0, 1\}^{n-1}$, $\boldsymbol{b} = \mathcal{E}_{\text{Den}}^{\text{b}}(\boldsymbol{a})$, $\boldsymbol{b}' = \mathcal{E}_{\text{Den}}^{\text{b}}(\bar{f}^{\text{b}}(\boldsymbol{b}))$ and $\boldsymbol{b}'' = \text{Rep}_{t+1}(\bar{f}^{\text{b}}(\boldsymbol{b}'))$. Given any $\boldsymbol{d} \in \mathcal{B}_{\leq t}(\boldsymbol{c})$, denoting $t' = |\boldsymbol{c}| - |\boldsymbol{b}|$, then $t' \leq t$ and we have $d_{[1, m_1 - t']} \in \mathcal{B}_{\leq t}(\boldsymbol{b})$, $d_{[m_1, m_2 - t']} \in \mathcal{B}_{\leq t}(\boldsymbol{b}')$ and $d_{[m_2, m_3 - t']} \in \mathcal{B}_{\leq t}(\boldsymbol{b}'')$, where $m_1 = |\boldsymbol{b}|, m_2 = |(\boldsymbol{b}, \boldsymbol{b}')|$ and $m_3 = |\boldsymbol{c}| = |(\boldsymbol{b}, \boldsymbol{b}', \boldsymbol{b}'')|$. First, since $\boldsymbol{b}'' = \text{Rep}_{t+1}(\bar{f}^{\text{b}}(\boldsymbol{b}'))$ is a codeword of a t-deletion code, then $\bar{f}^{\text{b}}(\boldsymbol{b}')$ can be recovered from $d_{[m_2, m_3 - t']}$. Further, by Theorem 3, $\boldsymbol{b}'$ can be recovered from $d_{[m_1, m_2 - t']}$ and $\bar{f}^{\text{b}}(\boldsymbol{b}')$, and so $\bar{f}^{\text{b}}(\boldsymbol{b})$ can be recovered from $\boldsymbol{b}' = \mathcal{E}_{\text{Den}}^{\text{b}}(\bar{f}^{\text{b}}(\boldsymbol{b}))$. Finally, by Theorem 3 again, $\boldsymbol{b}$ can be recovered from $d_{[1, m_1 - t']}$ and $\bar{f}^{\text{b}}(\boldsymbol{b})$. Thus, $\boldsymbol{c} = (\boldsymbol{b}, \boldsymbol{b}', \boldsymbol{b}'')$ can be recovered from any $\boldsymbol{d} \in \mathcal{B}_{\leq t}(\boldsymbol{c})$, which proves that $\bar{\mathcal{C}}^{\text{b}}$ is capable of correcting a burst of at most t deletions.

Since $\boldsymbol{a} \in \{0, 1\}^{n-1}$ and $\boldsymbol{b} = \mathcal{E}_{\text{Den}}^{\text{b}}(\boldsymbol{a}) \in \mathcal{S}_n^{\text{b}} \subseteq \{0, 1\}^n$, so $\boldsymbol{b}$ has one bit redundancy. Moreover, by Theorem 3, the length of $\boldsymbol{b}'$ is

$$|\boldsymbol{b}'| = \log n + 8 \log \log n + \gamma_t + o(\log \log n)$$

bits and the length of $\boldsymbol{b}''$ is

$$|\boldsymbol{b}''| = \log |\boldsymbol{b}'| + 8 \log \log |\boldsymbol{b}'| + \gamma_t + o(\log \log |\boldsymbol{b}'|)$$
$$= \log \log n + \gamma_t + o(\log \log n)$$

bits. So the total redundancy of $\boldsymbol{c} = \bar{\mathcal{E}}^{\text{b}}(\boldsymbol{a})$ is

$$\text{redundancy of } \bar{\mathcal{C}} = 1 + |\boldsymbol{b}'| + |\boldsymbol{b}''|$$
$$= \log n + 9 \log \log n + \gamma_t + o(\log \log n)$$

bits. $\blacksquare$

V. q-ary Codes Correcting a Burst of at most t Deletions

In this section, we construct q-ary codes correcting a bursting of at most t deletions, where $q > 2$ is an even integer. We assume that q and t are constant with respect to the code length n. As in Section III, we identify each binary string $\boldsymbol{a}$ with the positive integer whose binary representation is $\boldsymbol{a}$. As stated in Section II, we take

$$\delta = t2^{t+1} \log n$$

and

$$\boldsymbol{p} = 0^t 1^t.$$

A string $\boldsymbol{c} \in \{0,1\}^n$ is called $(\boldsymbol{p}, \delta)$-dense, if each substring of $\boldsymbol{c}$ of length δ contains at least one pattern $\boldsymbol{p}$.

For each $\boldsymbol{x} \in \mathbb{Z}_q^n$, let $M_{\boldsymbol{x}} = (c_{i,j})_{\lceil \log q \rceil \times n}$ be the matrix representation of $\boldsymbol{x}$ as defined by (1). Then for each $t' \in [t]$, the deletion of $x_i, x_{i+1}, \cdots, x_{i+t'-1}$ results in the deletion of the columns $i, i+1, \cdots, i+t'-1$ of $M_{\boldsymbol{x}}$. A basic idea is to protect the first row $\boldsymbol{c} = c_{1,[n]}$ by a burst-deletion correcting code. However, in general, if $\boldsymbol{c}$ can be recovered from a $\boldsymbol{d} \in \mathcal{B}_{\leq t}(\boldsymbol{c})$, the location of the deleted symbols can not be determined. For example, consider $\boldsymbol{c} = 0111011011010010$ and $\boldsymbol{d} = 0111011010010$. Then $\boldsymbol{d}$ can be obtained from $\boldsymbol{c}$ by deleting $c_3 c_4 c_5 = 110$, or deleting $c_4 c_5 c_6 = 101$. In fact, $\boldsymbol{d}$ can be obtained from $\boldsymbol{c}$ by deleting $c_i c_{i+1} c_{i+2}$ for all $i \in [3, 10]$. To proceed, we need to consider period of binary strings.

Let ℓ and m be two positive integers such that $\ell \leq m$. A string $\boldsymbol{a} \in \{0,1\}^m$ is said to have *period* ℓ (or $\boldsymbol{a}$ is called a period-ℓ string) if $a_{i+\ell} = a_i$ for all $i \in [m-\ell] = \{1, 2, \cdots, m-\ell\}$. Clearly, a run of $\boldsymbol{c}$ of length m has period ℓ for any $\ell \in [m]$; a period-2 substring of $\boldsymbol{c}$ is either a run of $\boldsymbol{c}$ or an alternative substring of $\boldsymbol{c}$.

Lemma 10: Suppose $\boldsymbol{c} \in \{0,1\}^n$ is $(\boldsymbol{p}, \delta)$-dense. Given any $\boldsymbol{d} \in \mathcal{B}_{\leq t}(\boldsymbol{c})$, it is possible to find an interval $K \subseteq [n]$ of length at most $\delta - 1$ such that if $\boldsymbol{d} = c_{[n] \setminus D}$ and $D \subseteq [n]$ is an interval, then it always holds that $D \subseteq K$.

Proof: Since $\boldsymbol{d} \in \mathcal{B}_{\leq t}(\boldsymbol{c})$, there is an interval $D' \subseteq [n]$ such that $\boldsymbol{d} = c_{[n] \setminus D'}$. Let $K \subseteq [n]$ be the interval such that c_K is the maximal substring of $\boldsymbol{c}$ satisfying: 1) c_K has period $t' = |\boldsymbol{c}| - |\boldsymbol{d}|$; 2) c_K contains $c_{D'}$. We will prove that $D \subseteq K$ for any interval $D \subseteq [n]$ such that $\boldsymbol{d} = c_{[n] \setminus D}$.

Suppose $D = [i_1, i_1 + t' - 1]$ and $D' = [i_2, i_2 + t' - 1]$. Without loss of generality, assume $i_1 \leq i_2$. Since $c_{[n] \setminus D} = \boldsymbol{d} = c_{[n] \setminus D'}$, we have

$$c_1 \ \cdots \ c_{i_1-1} \ c_{i_1+t'} \ c_{i_1+t'+1} \ \cdots \ c_{i_2+t'-1} \ c_{i_2+t'} \ \cdots \ c_n$$
$$= \ c_1 \ \cdots \ c_{i_1-1} \ c_{i_1} \quad c_{i_1+1} \quad \cdots \quad c_{i_2-1} \quad c_{i_2+t'} \ \cdots \ c_n.$$

By comparing the symbols of $c_{[n] \setminus D'}$ and $c_{[n] \setminus D''}$ in each position, we can obtain $c_i = c_{i+t'}$ for each $i \in [i_1, i_2 - 1]$. So, $c_{[i_1, i_2 + t' - 1]}$ is a substring of $\boldsymbol{c}$ of period t' and contains both c_D and $c_{D'}$. As c_K is the maximal substring of $\boldsymbol{c}$ of period t' that contains $c_{D'}$, so $c_{[i_1, i_2 + t' - 1]}$ is contained in c_K. Thus, c_D is contained in c_K, which implies that $D \subseteq K$.

Since $\boldsymbol{c}$ is $(\boldsymbol{p}, \delta)$-dense, where $\boldsymbol{p} = 0^t 1^t$, then each substring of $\boldsymbol{c}$ of length δ contains at least one pattern $\boldsymbol{p}$. Note that for each $t' \in [t]$, we have $p_t = 0 \neq 1 = p_{t+t'}$, so each substring of $\boldsymbol{c}$ of length δ can not has period t'. In other words, the length of any period-t' substring of $\boldsymbol{c}$ is at most $\delta - 1$. Thus, the length of c_I (and the length of I) is at most $\delta - 1$. ∎

Let

$$K_j = \begin{cases} [(j-1)\delta + 1, (j+1)\delta], & \text{for } j \in \{1, \cdots, \lceil n/\delta \rceil - 2\}, \\ [(j-1)\delta + 1, n], & \text{for } j = \lceil n/\delta \rceil - 1. \end{cases} \tag{13}$$

Remark 4: Similar to Remark 2, it is easy to see that

1) For any interval $K \subseteq [n]$ of length at most δ, there is an $j_0 \in \{1, 2, \cdots, \lceil n/\delta \rceil - 1\}$ such that $K \subseteq K_{j_0}$.
2) $K_j \cap K_{j'} = \emptyset$ for all $j, j' \in \{1, 2, \cdots, \lceil n/\delta \rceil - 1\}$ such that $|j - j'| \geq 2$.

Let ϕ be the function constructed by Lemma 5 and $\bar{f}^{\mathrm{b}}$ be the function constructed in Construction 2. For each $\boldsymbol{x} \in \mathbb{Z}_q^n$, let $M_{\boldsymbol{x}} = (c_{i,j})_{\lceil \log q \rceil \times n}$ be the matrix representation of $\boldsymbol{x}$ as defined by (1). We have the following construction.

Construction 3: For each $\boldsymbol{x} \in \mathbb{Z}_q^n$ and each $j \in \{1, 2, \cdots, \lceil n/\delta \rceil - 1\}$, let

$$\bar{h}_j(\boldsymbol{x}) = \big(\phi(c_{2,K_j}), \phi(c_{3,K_j}), \cdots, \phi(c_{\lceil \log q \rceil, K_j})\big)$$

and for each $\ell \in \{0, 1\}$, let

$$\bar{h}^{(\ell)}(\boldsymbol{x}) = \sum_{\substack{j \in \{1, 2, \cdots, \lceil n/\delta \rceil - 1\}: \\ j \equiv \ell \bmod 2}} \bar{h}_j(\boldsymbol{x}) \bmod \overline{N}, \tag{14}$$

where

$$\overline{N} = q^{4 \log \log n + o(\log \log n) + \gamma_t}.$$

Finally, let

$$\bar{f}(\boldsymbol{x}) = \left(\bar{f}^{\mathrm{b}}(c_{1,[n]}), \bar{h}^{(0)}(\boldsymbol{x}), \bar{h}^{(1)}(\boldsymbol{x}) \right). \tag{15}$$

We have the following theorem.

Theorem 5: For any $\boldsymbol{x} \in \mathbb{Z}_q^n$, $\bar{f}(\boldsymbol{x})$ is computable in linear time, and when viewed as a binary string, the length $|\bar{f}(\boldsymbol{x})|$ of $\bar{f}(\boldsymbol{x})$ satisfies

$$|\bar{f}(\boldsymbol{x})| \leq \log n + 8(\log q + 1) \log \log n + o(\log q \log \log n) + \gamma_t,$$

where γ_t is a constant depending only on t. Moreover, if $\boldsymbol{c} = c_{1,[n]}$ is $(\boldsymbol{p}, \delta)$-dense, then given $\bar{f}(\boldsymbol{x})$ and any $\boldsymbol{y} \in \mathcal{B}_{\leq t}(\boldsymbol{x})$, one can uniquely recover $\boldsymbol{x}$.

Proof: Note that by Theorem 3, $\bar{f}^{\mathrm{b}}(c_{1,[n]})$ is computable in linear time. Moreover, by Lemma 5 and (13), each $\phi(c_{2,K_j})$ is computable in time $O(2^t(2\delta)^3) = O((\log n)^3)$, so by Construction 3, $\bar{h}^{(\ell)}(\boldsymbol{x}), \ell = 1, 2$, are computable in linear time. Hence, $\bar{f}(\boldsymbol{x}) = \left(\bar{f}^{\mathrm{b}}(c_{1,[n]}), \bar{h}^{(0)}(\boldsymbol{x}), \bar{h}^{(1)}(\boldsymbol{x}) \right)$ is computable in linear time.

By Theorem 3, the length of $\bar{f}^{\mathrm{b}}(c_{1,[n]})$ satisfies

$$|\bar{f}^{\mathrm{b}}(c_{1,[n]})| \leq \log n + 8 \log \log n + \gamma_t + o(\log \log n).$$

Moreover, by Construction 3, the length of $\bar{h}^{(\ell)}(\boldsymbol{x}), \ell = 1, 2$, satisfy

$$|\bar{h}^{(\ell)}(\boldsymbol{x})| \leq \log \overline{N} = \log q(4 \log \log n + o(\log \log n) + \gamma_t).$$

Hence, the length of $\bar{f}(\boldsymbol{x})$ satisfies

$$\begin{aligned}
|\bar{f}(\boldsymbol{x})| &= |\bar{f}^{\mathrm{b}}(c_{1,[n]})| + |\bar{g}^{(0)}(\boldsymbol{x})| + |\bar{g}^{(1)}(\boldsymbol{x})| \\
&\leq \log n + 8(\log q + 1) \log \log n + o(\log q \log \log n) \\
&\quad + \gamma_t.
\end{aligned}$$

It remains to prove that if $\boldsymbol{c} = c_{1,[n]}$ is $(\boldsymbol{p}, \delta)$-dense, then given $\bar{f}(\boldsymbol{x})$ and any $\boldsymbol{y} \in \mathcal{B}_{\leq t}(\boldsymbol{x})$, one can uniquely recover $\boldsymbol{x}$. To prove this, we first prove that

$$\bar{h}_j(\boldsymbol{x}) < \overline{N}$$

for each $j \in \{1, 2, \cdots, \lceil n/\delta \rceil - 1\}$. In fact, by (13), each c_{i,K_j}, $i \in [2, \lceil \log q \rceil]$, has length $2\delta = 2t2^{t+1} \log n$, so by Lemma 5, $\phi(c_{2,K_j})$ has length $4 \log(2\delta) + o(\log(2\delta)) = 4 \log \log n + \gamma_t + o(\log \log n)$. Hence, by Construction 3, we have

$$\begin{aligned}
|\bar{h}_j(\boldsymbol{x})| &= | \left(\phi(c_{2,K_j}), \phi(c_{3,K_j}), \cdots, \phi(c_{\lceil \log q \rceil, K_j}) \right) | \\
&= (\lceil \log q \rceil - 1) \left(4 \log \log n + \gamma_t + o(\log \log n) \right) \\
&< \log q \left(4 \log \log n + \gamma_t + o(\log \log n) \right),
\end{aligned}$$

which implies that $\bar{h}_j(\boldsymbol{x}) < q^{4 \log \log n + \gamma_t + o(\log \log n)} = \overline{N}$.

Now, we prove that $\boldsymbol{x}$ can be uniquely recovered from $\bar{f}(\boldsymbol{x})$ and any given $\boldsymbol{y} \in \mathcal{B}_{\leq t}(\boldsymbol{x})$, provided that $\boldsymbol{c} = c_{1,[n]}$ is $(\boldsymbol{p}, \delta)$-dense. Let

$$M_{\boldsymbol{y}} = (d_{i,j})_{\lceil \log q \rceil \times (n-t')}$$

be the matrix representation of $\boldsymbol{y}$. Since $\boldsymbol{y} \in \mathcal{B}_{\leq t}(\boldsymbol{x})$ can be obtained from $\boldsymbol{x}$ by deleting t' consecutive symbols from $\boldsymbol{x}$, where $t' = n - |\boldsymbol{y}|$ and $t' \in [t]$, then $M_{\boldsymbol{y}}$ can be obtained from $M_{\boldsymbol{x}}$ by deleting t' consecutive columns of $M_{\boldsymbol{x}}$. The process of recovering $\boldsymbol{x}$ from $\bar{f}(\boldsymbol{x}) = \left(\bar{f}^{\mathrm{b}}(c_{1,[n]}), \bar{g}^{(0)}(\boldsymbol{x}), \bar{g}^{(1)}(\boldsymbol{x}) \right)$ and $\boldsymbol{y}$ consists of the following three steps.

Step 1: Since $\boldsymbol{c} = c_{1,[n]}$ is $(\boldsymbol{p}, \delta)$-dense, then by Theorem 3, $c_{1,[n]}$ can be recovered from $d_{1,[n-t']}$ and $\bar{f}^{\mathrm{b}}(c_{1,[n]})$.

Step 2: According to Lemma 10, there is an interval $K \subseteq [n]$ of length at most $\delta - 1$ such that $d_{1,[n-t']}$ is obtained from $\boldsymbol{c} = c_{1,[n]}$ by deleting t' consecutive symbols in K. Correspondingly, $M_{\boldsymbol{y}}$ is obtained from $M_{\boldsymbol{x}}$ by deleting t' consecutive columns in K. By 1) of Remark 4, there is an $j_0 \in \{1, 2, \cdots, \lceil n/\delta \rceil - 1\}$ such that $K \subseteq K_{j_0}$. Denote $K_{j_0} = [\lambda, \lambda']$. Then we have the following observations:

i) $c_{i,[1,\lambda-1]} = d_{i,[1,\lambda-1]}$ for each $i \in [2, \lceil \log q \rceil]$.
ii) $d_{i,[\lambda,\lambda'-t']} \in \mathcal{B}_{\leq t}(c_{i,[\lambda,\lambda']})$ for each $i \in [2, \lceil \log q \rceil]$.
iii) $c_{i,[\lambda'+1,n]} = d_{i,[\lambda'-t'+1,n-t']}$ for each $i \in [2, \lceil \log q \rceil]$.

By observations i) and iii), for each $i \in [2, \lceil \log q \rceil]$, $c_{i,[1,\lambda-1]}$ and $c_{i,[\lambda'+1,n]}$ can be directly obtained from $M_{\boldsymbol{y}}$.

Step 3: For $j \in \{1, \cdots, j_0 - 2\}$, we have $K_j \cap K_{j_0} = \emptyset$, by 2) of Remark 4, so $K_j \subseteq [1, \lambda - 1]$ and by observation i), $\bar{h}_j(\boldsymbol{x}) = \left(\phi(c_{2,K_j}), \phi(c_{3,K_j}), \cdots, \phi(c_{\lceil \log q \rceil, K_j}) \right)$ can be computed from $d_{i,[1,\lambda-1]} = c_{i,[1,\lambda-1]}$, $i = 2, \cdots, \lceil \log q \rceil$. Similarly, for $j \in \{j_0 + 2, \cdots, \lceil n/\delta \rceil - 1\}$, by 2) of Remark 4, we have $K_j \cap K_{j_0} = \emptyset$, so $K_j \subseteq [\lambda' + 1, n]$ and by observation iii),

$\bar{h}_j(\boldsymbol{x}) = \big(\phi(c_{2,K_j}), \phi(c_{3,K_j}), \cdots, \phi(c_{\lceil \log q\rceil, K_j})\big)$ can be computed from $d_{i,[\lambda'-t'+1,n-t']} = c_{i,[\lambda'+1,n]}$, $i = 2, \cdots, \lceil \log q\rceil$. Let $\ell_0 \in \{0,1\}$ be such that $\ell_0 \equiv j_0 \mod 2$. By (14), we can obtain

$$\bar{h}_{j_0}(\boldsymbol{x}) \equiv \bar{h}^{(\ell_0)}(\boldsymbol{x}) - \sum_{\substack{j \in \{1,2,\cdots,\lceil n/\delta\rceil - 1\}\setminus\{j_0\}: \\ j \equiv \ell_0 \mod 2}} \bar{h}_j(\boldsymbol{x}) \mod \overline{N}.$$

Note that we have proved that $\bar{h}_j(\boldsymbol{x}) < \overline{N}$ for each $j \in \{1, 2, \cdots, \lceil n/\delta\rceil - 1\}$, so we actually have

$$\bar{h}_{j_0}(\boldsymbol{x}) = \bar{h}^{(\ell_0)}(\boldsymbol{x}) - \sum_{\substack{j \in \{1,2,\cdots,\lceil n/\delta\rceil - 1\}\setminus\{j_0\}: \\ j \equiv \ell_0 \mod 2}} \bar{h}_j(\boldsymbol{x}) \mod \overline{N},$$

where by Construction 3,

$$\bar{h}_{j_0}(\boldsymbol{x}) = \big(\phi(c_{2,K_{j_0}}), \phi(c_{3,K_{j_0}}), \cdots, \phi(c_{\lceil \log q\rceil, K_{j_0}})\big).$$

By observation ii) in Step 2, and by Lemma 5, $c_{i,[\lambda,\lambda']}$, $i = 2, \cdots, \lceil \log q\rceil$, can be recovered from $\bar{h}_{j_0}(\boldsymbol{x})$ and $d_{i,[\lambda,\lambda'-t']}$, $i = 2, \cdots, \lceil \log q\rceil$.

By the above discussions, $M_{\boldsymbol{x}}$ (and so $\boldsymbol{x}$) can be uniquely recovered from $\bar{f}(\boldsymbol{x})$ and any given $\boldsymbol{y} \in \mathcal{B}_{\leq t}(\boldsymbol{x})$. ∎

Let $\mathcal{S}_n$ be the set of all q-ary strings $\boldsymbol{x} \in \mathbb{Z}_q^n$ such that the first row $c_{1,[n]}$ of $M_{\boldsymbol{x}}$ is $(\boldsymbol{p}, \delta)$-dense, where $M_{\boldsymbol{x}} = (c_{i,j})_{\lceil \log q\rceil \times n}$ is the matrix representation of $\boldsymbol{x}$. Let

$$\mathcal{E}_{\mathrm{Den}}^{\mathrm{b}} : \{0,1\}^{n-1} \to \mathcal{S}_n^{\mathrm{b}}$$

be the one-to-one mapping constructed as in (11), where $\mathcal{S}_n^{\mathrm{b}}$ is the set of all $(\boldsymbol{p}, \delta)$-dense strings in $\{0,1\}^n$. By Lemma 8, the mapping $\mathcal{E}_{\mathrm{Den}}^{\mathrm{b}}$ can be extended to a one-to-one mapping, denoted by

$$\mathcal{E}_{\mathrm{Den}} : \mathbb{Z}_q^{n-1} \to \mathbb{Z}_q^n,$$

such that for each $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$ and $\boldsymbol{x} = \mathcal{E}_{\mathrm{Den}}(\boldsymbol{u})$, if $M_{\boldsymbol{u}} = (b_{i,j})_{\lceil \log q\rceil \times (n-1)}$ and $M_{\boldsymbol{x}} = (c_{i,j})_{\lceil \log q\rceil \times n}$ are the matrix representation of $\boldsymbol{u}$ and $\boldsymbol{x}$ respectively, then

$$c_{1,[n]} = \mathcal{E}_{\mathrm{Den}}^{\mathrm{b}}(b_{1,[n-1]}).$$

Since $\mathcal{E}_{\mathrm{Den}}^{\mathrm{b}}(b_{1,[n-1]})$ is $(\boldsymbol{p}, \delta)$-dense, then for any $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$, we have $\boldsymbol{x} = \mathcal{E}_{\mathrm{Den}}(\boldsymbol{u}) \in \mathcal{S}_n$.

As in Section III, for each binary string $\boldsymbol{a}$, let $\mathcal{Q}(\boldsymbol{a})$ be the q-ary representation of $\boldsymbol{a}$. Note that $\mathcal{Q}(\boldsymbol{a})$ is a q-ary string of length $\lceil |\boldsymbol{a}|/\lfloor \log q\rfloor\rceil$. Using the function $\bar{f}$ constructed in Construction 3 and the mapping $\mathcal{E}_{\mathrm{Den}}$, we can construct an encoding function of a q-ary code capable of correcting a burst of at most t deletions.

Let $\bar{\mathcal{E}}$ be a function defined on $\mathbb{Z}_q^{n-1}$ of the form

$$\bar{\mathcal{E}}(\boldsymbol{u}) = (\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}''), \ \forall \boldsymbol{u} \in \mathbb{Z}_q^{n-1}, \tag{16}$$

where $\boldsymbol{v} = \mathcal{E}_{\mathrm{Den}}(\boldsymbol{u})$, $\boldsymbol{v}' = \mathcal{E}_{\mathrm{Den}}(\mathcal{Q}(\bar{f}(\boldsymbol{v})))$ and $\boldsymbol{v}'' = \mathrm{Rep}_{t+1}(\mathcal{Q}(\bar{f}(\boldsymbol{v}')))$ such that $\mathrm{Rep}_{t+1}(\cdot)$ is the encoding function of the $(t+1)$-fold repetition code.

Theorem 6: The code $\bar{\mathcal{C}} = \{\bar{\mathcal{E}}(\boldsymbol{u}) : \boldsymbol{u} \in \mathbb{Z}_q^{n-1}\}$, where $\bar{\mathcal{E}}$ is given by (16), is a q-ary code capable of correcting a burst of at most t deletions. The redundancy of $\bar{\mathcal{C}}$ is at most $\log n + (8\log q + 9)\log\log n + o(\log q \log\log n) + \gamma_t$ in bits, where γ_t is a constant depending only on t.

Proof: To prove that $\bar{\mathcal{C}}$ is capable of correcting a burst of at most t deletions, we adopt a similar strategy as in the proof of Theorem 2. Specifically, let

$$\boldsymbol{x} = \bar{\mathcal{E}}(\boldsymbol{u}) = (\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}'') \in \bar{\mathcal{C}},$$

where $\boldsymbol{v} = \mathcal{E}_{\mathrm{Den}}(\boldsymbol{u})$, $\boldsymbol{v}' = \mathcal{E}_{\mathrm{Den}}(\mathcal{Q}(\bar{f}(\boldsymbol{v})))$ and $\boldsymbol{v}'' = \mathrm{Rep}_{t+1}(\mathcal{Q}(\bar{f}(\boldsymbol{v}')))$. Given any $\boldsymbol{y} \in \mathcal{B}_{\leq t}(\boldsymbol{x})$, denoting $t' = |\boldsymbol{x}| - |\boldsymbol{y}|$, then $t' \leq t$ and we have $\boldsymbol{y}_{[1,m_1-t']} \in \mathcal{B}_{\leq t}(\boldsymbol{v})$, $\boldsymbol{y}_{[m_1,m_2-t']} \in \mathcal{B}_{\leq t}(\boldsymbol{v}')$ and $\boldsymbol{y}_{[m_2,m_3-t']} \in \mathcal{B}_{\leq t}(\boldsymbol{v}'')$, where $m_1 = |\boldsymbol{v}|, m_2 = |(\boldsymbol{v}, \boldsymbol{v}')|$ and $m_3 = |\boldsymbol{x}| = |(\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}'')|$. First, since $\boldsymbol{v}'' = \mathrm{Rep}_{t+1}(\mathcal{Q}(\bar{f}(\boldsymbol{v}')))$ is a codeword of a t-deletion code, then $\mathcal{Q}(\bar{f}(\boldsymbol{v}'))$, and so $\bar{f}(\boldsymbol{v}')$, can be recovered from $\boldsymbol{y}_{[m_2,m_3-t']}$. Further, by Theorem 5, $\boldsymbol{v}'$ can be recovered from $\boldsymbol{y}_{[m_1,m_2-t']}$ and $\bar{f}(\boldsymbol{v}')$, and so $\bar{f}(\boldsymbol{v})$ can be recovered from $\boldsymbol{v}' = \mathcal{E}_{\mathrm{Den}}(\mathcal{Q}(\bar{f}(\boldsymbol{v})))$. Finally, by Theorem 5 again, $\boldsymbol{v}$ can be recovered from $\boldsymbol{y}_{[1,m_1-t']}$ and $\bar{f}(\boldsymbol{v})$. Thus, $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{v}', \boldsymbol{v}'')$ can be recovered from any $\boldsymbol{y} \in \mathcal{B}_{\leq t}(\boldsymbol{x})$, which proves that $\bar{\mathcal{C}}$ is capable of correcting a burst of at most t deletions.

Since $\boldsymbol{u} \in \mathbb{Z}_q^{n-1}$ and $\boldsymbol{v} = \mathcal{E}_{\mathrm{Den}}(\boldsymbol{u}) \in \mathcal{S}_n \subseteq \mathbb{Z}_q^n$, so $\boldsymbol{v}$ has $\log q$ bits redundancy. Moreover, by Theorem 5, the length of $\boldsymbol{v}'$ is

$$|\boldsymbol{v}'| = \log n + 8(\log q + 1)\log\log n + o(\log q \log\log n) + \gamma_t$$

bits and the length of $\boldsymbol{v}''$ is

$$
\begin{aligned}
|\boldsymbol{v}''| &= \log|\boldsymbol{v}'| + 8(\log q + 1)\log\log|\boldsymbol{v}'| + o(\log q \log\log|\boldsymbol{b}'|) \\
&\quad + \gamma_t \\
&= \log\log n + \gamma_t + o(\log q \log\log n)
\end{aligned}
$$

in bits. So the total redundancy of $\boldsymbol{c} = \bar{\mathcal{E}}^{\mathrm{b}}(\boldsymbol{a})$ is

$$
\begin{aligned}
\text{redundancy of } \bar{\mathcal{C}} &= \log q + |\boldsymbol{v}'| + |\boldsymbol{v}''| \\
&= \log n + (8\log q + 9)\log\log n \\
&\quad + o(\log q \log\log n) + \gamma_t
\end{aligned}
$$

bits. ∎

VI. CONCLUSIONS

We constructed systematic q-ary two-deletion correcting codes and q-ary burst-deletion correcting codes, where $q \geq 2$ is an even integer. For q-ary two-deletion codes, the redundancy of our construction is $\log n$ higher than the best known explicit binary codes and is lower than all existing explicit q-ary codes. For q-ary burst-deletion codes, our construction is scaling-optimal in redundancy.

It is also an interesting problem to generalize our constructions to odd q. Another interesting problem is to construct explicit q-ary t-deletion correcting codes that improves upon the state of the art constructions in redundancy.

APPENDIX A
PROOF OF LEMMA 9

In this appendix, we prove Lemma 9. We first need to prove the following lemma.

Lemma 11: Suppose $\boldsymbol{c} \in \{0,1\}^n$ and $\{j_1, j_2\}, \{j_1', j_2'\} \subseteq [n]$ such that $j_1 < j_2, j_1' < j_2'$ and $j_1 \leq j_1'$. If $c_{[n]\setminus\{j_1,j_2\}} = c_{[n]\setminus\{j_1',j_2'\}}$, then one of the following holds:

1) $c_{j_1}, c_{j_1'}$ are in the same run of $\boldsymbol{c}$, and $c_{j_2}, c_{j_2'}$ are in the same run of $\boldsymbol{c}$.
2) There is an alternative substring $c_{[s_1,s_2]}$ of $\boldsymbol{c}$ of length ≥ 3 such that c_{j_1} and c_{s_1} are in the same run of $\boldsymbol{c}$, $j_2 = s_1 + 1$, $j_1' = s_2 - 1$ and $c_{j_2'}$ and c_{s_2} are in the same run of $\boldsymbol{c}$.

Proof: If $\{j_1, j_2\} = \{j_1', j_2'\}$, then 1) holds. In the following, we assume that $\{j_1, j_2\} \neq \{j_1', j_2'\}$. Since $c_{[n]\setminus\{j_1,j_2\}} = c_{[n]\setminus\{j_1',j_2'\}}$, we can denote $\boldsymbol{b} = c_{[n]\setminus\{j_1,j_2\}} = c_{[n]\setminus\{j_1',j_2'\}}$. From $\boldsymbol{b} = c_{[n]\setminus\{j_1,j_2\}}$ we can obtain

$$
b_i = \begin{cases} c_i, & \text{for } i \in [1, j_1 - 1], \\ c_{i+1}, & \text{for } i \in [j_1, j_2 - 2], \\ c_{i+2}, & \text{for } i \in [j_2 - 1, n - 2]. \end{cases} \tag{17}
$$

Similarly, from $\boldsymbol{b} = c_{[n]\setminus\{j_1',j_2'\}}$ we can obtain

$$
b_i = \begin{cases} c_i, & \text{for } i \in [1, j_1' - 1], \\ c_{i+1}, & \text{for } i \in [j_1', j_2' - 2], \\ c_{i+2}, & \text{for } i \in [j_2' - 1, n - 2]. \end{cases} \tag{18}
$$

Since $j_1 < j_2, j_1' < j_2'$ and $j_1 \leq j_1'$, then we can divide our discussions into the following three cases:

Case 1: $j_1 < j_2 \leq j_1' < j_2'$. Combining (17) and (18), we can obtain

$$
\begin{aligned}
\boldsymbol{b} &= c_1 \cdots c_{j_1-1} c_{j_1+1} c_{j_1+2} \cdots c_{j_2-1} c_{j_2+1} c_{j_2+2} c_{j_2+3} c_{j_2+4} \cdots c_{j_1'-2} c_{j_1'-1} \ c_{j_1'} \ c_{j_1'+1} c_{j_1'+2} c_{j_1'+3} \cdots c_{j_2'-1} \ c_{j_2'} \ c_{j_2'+1} \cdots c_n \\
&= c_1 \cdots c_{j_1-1} \ c_{j_1} \ c_{j_1+1} \cdots c_{j_2-2} c_{j_2-1} \ c_{j_2} \ c_{j_2+1} c_{j_2+2} \cdots c_{j_1'-4} c_{j_1'-3} c_{j_1'-2} c_{j_1'-1} c_{j_1'+1} c_{j_1'+2} \cdots c_{j_2'-2} c_{j_2'-1} c_{j_2'+1} \cdots c_n.
\end{aligned} \tag{19}
$$

We further need to divide this case into the following two subcases.

Case 1.1: $j_1' - j_2$ is odd. By comparing the symbols of the corresponding positions in $c_{[n]\setminus\{j_1,j_2\}}$ and $c_{[n]\setminus\{j_1',j_2'\}}$, we can obtain

$$
\begin{aligned}
c_{j_1} = c_{j_1+1} &= \cdots = c_{j_2-2} = c_{j_2-1} = c_{j_2+1} = c_{j_2+3} = \cdots \\
&= c_{j_1'-4} = c_{j_1'-2} = c_{j_1'}
\end{aligned}
$$

and

$$c_{j_2} = c_{j_2+2} = c_{j_2+4} = \cdots = c_{j_1'-3} = c_{j_1'-1} = c_{j_1'+1}$$
$$= c_{j_1'+2} = c_{j_1'+3} = \cdots = c_{j_2'}.$$

Case 1.2: $j_1' - j_2$ is even. By comparing the symbols of the corresponding positions in $c_{[n]\setminus\{j_1,j_2\}}$ and $c_{[n]\setminus\{j_1',j_2'\}}$, we can obtain

$$c_{j_1} = c_{j_1+1} = c_{j_1+2} = \cdots = c_{j_2-2} = c_{j_2-1} = c_{j_2+1}$$
$$= c_{j_2+3} = \cdots = c_{j_1'-3} = c_{j_1'-1} = c_{j_1'+1} = c_{j_1'+2}$$
$$= c_{j_1'+3} = \cdots = c_{j_2'-1} = c_{j_2'}$$

and

$$c_{j_2} = c_{j_2+2} = c_{j_2+4} = \cdots = c_{j_1'-4} = c_{j_1'-2} = c_{j_1'}$$

For the both subcases, we can see that

- If $c_{j_1} = c_{j_2}$, then we have $c_{j_1} = c_{j_1+1} = \cdots = c_{j_2'}$, so $c_{j_1}, c_{j_2}, c_{j_1'}$ and $c_{j_2'}$ are in the same run of c, which implies that 1) of Lemma 11 holds.
- If $c_{j_1} \neq c_{j_2}$, then $c_{[j_2-1,j_1'+1]}$ is an alternative substring of c of length ≥ 3. By letting $s_1 = j_2 - 1$ and $s_2 = j_1' + 1$, we obtain 2) of Lemma 11.

Example 1) is an illustration of this case.

Case 2: $j_1 \leq j_1' < j_2 \leq j_2'$. In this case, combining (17) and (18), we can obtain

$$\boldsymbol{b} = c_1 \cdots c_{j_1-1} c_{j_1+1} c_{j_1+2} \cdots c_{j_1'-1} \; c_{j_1'} \quad c_{j_1'+1} c_{j_1'+2} \cdots c_{j_2-1} c_{j_2+1} c_{j_2+2} c_{j_2+3} \cdots c_{j_2'-1} \; c_{j_2'} \quad c_{j_2'+1} \cdots c_n$$
$$= c_1 \cdots c_{j_1-1} \; c_{j_1} \quad c_{j_1+1} \cdots c_{j_1'-2} c_{j_1'-1} c_{j_1'+1} c_{j_1'+2} \cdots c_{j_2-1} \; c_{j_2} \quad c_{j_2+1} c_{j_2+2} \cdots c_{j_2'-2} c_{j_2'-1} c_{j_2'+1} \cdots c_n, \tag{20}$$

from which we see that

$$c_{j_1} = c_{j_1+1} = \cdots = c_{j_1'}$$

and

$$c_{j_2} = c_{j_2+1} = \cdots = c_{j_2'}.$$

Hence, 1) of Lemma 11 holds.

Case 3: $j_1 \leq j_1' < j_2' < j_2$. In this case, combining (17) and (18), we can obtain

$$\boldsymbol{b} = c_1 \cdots c_{j_1-1} c_{j_1+1} c_{j_1+2} \cdots c_{j_1'-1} \; c_{j_1'} \quad c_{j_1'+1} c_{j_1'+2} \cdots c_{j_2'-1} \; c_{j_2'} \quad c_{j_2'+1} c_{j_2'+2} \cdots c_{j_2-1} c_{j_2+1} c_{j_2+2} \cdots c_n$$
$$= c_1 \cdots c_{j_1-1} \; c_{j_1} \quad c_{j_1+1} \cdots c_{j_1'-2} c_{j_1'-1} c_{j_1'+1} c_{j_1'+2} \cdots c_{j_2'-1} c_{j_2'+1} c_{j_2'+2} c_{j_2'+3} \cdots \; c_{j_2} \quad c_{j_2+1} c_{j_2+2} \cdots c_n, \tag{21}$$

from which we can see that

$$c_{j_1} = c_{j_1+1} = \cdots = c_{j_1'}$$

and

$$c_{j_2'} = c_{j_2'+1} = \cdots = c_{j_2}.$$

Hence, 1) of Lemma 11 holds. ∎

Example 1: To illustrate the Case 1 in the proof of Lemma 11, let's consider the following two examples.

- Consider $\boldsymbol{c} = 01000101011110$. Let $j_1 = 3$, $j_2 = 6$, $j_1' = 9$ and $j_2' = 12$. Then $c_{[n]\setminus\{j_1,j_2\}} = c_{[n]\setminus\{j_1',j_2'\}} = 010001011110$ and $j_1' - j_2 = 9 - 6 = 3$ is odd. We can find that $c_{j_1} = \cdots = c_{j_2-1} = c_{j_2+1} = c_{j_2+3} = \cdots = c_{j_1'} = 0$ and $c_{j_2} = c_{j_2+2} = \cdots = c_{j_1'-1} = c_{j_1'+1} = \cdots = c_{j_2'} = 1$.
- Consider $\boldsymbol{c} = 01000101010001$. Let $j_1 = 3$, $j_2 = 6$, $j_1' = 10$ and $j_2' = 12$. Then $c_{[n]\setminus\{j_1,j_2\}} = c_{[n]\setminus\{j_1',j_2'\}} = 010001010001$ and $j_1' - j_2 = 10 - 6 = 4$ is even. We can find that $c_{j_1} = \cdots = c_{j_2-1} = c_{j_2+1} = c_{j_2+3} = \cdots = c_{j_1'-1} = c_{j_1'+1} = c_{j_1'+2} = \cdots = c_{j_2'} = 0$ and $c_{j_2} = c_{j_2+2} = \cdots = c_{j_1'} = 1$.

Now, we can prove Lemma 9.

Proof of Lemma 9: Suppose $\boldsymbol{c} \in \{0,1\}^n$ is regular and $\boldsymbol{b} \in \{0,1\}^{n-2}$ such that $\boldsymbol{b}$ can be obtained from $\boldsymbol{c}$ by deleting two symbols of $\boldsymbol{c}$. Then we can always find two symbols of $\boldsymbol{c}$, say c_{j_1} and c_{j_2} ($j_1 < j_2$), such that, $\boldsymbol{b} = c_{[n]\setminus\{j_1,j_2\}}$.

First, suppose c_{j_1} and c_{j_2} are in the same run of $\boldsymbol{c}$. Then for any $\{j_1', j_2'\} \subseteq [n]$ such that $\boldsymbol{b} = c_{[n]\setminus\{j_1',j_2'\}}$, it is easy to see that 2) of Lemma 11 can't hold. (Otherwise, there is an alternative substring $c_{[s_1,s_2]}$ of $\boldsymbol{c}$ of length ≥ 3 such that c_{j_1}, c_{s_1} are in the same run of $\boldsymbol{c}$ and $j_2 = s_1 + 1$, which implies that $c_{j_1} = c_{s_1} \neq c_{s_1+1} = c_{j_2}$, which contradicts to the assumption that c_{j_1} and c_{j_2} are in the same run of $\boldsymbol{c}$.) Therefore, 1) of Lemma 11 must hold, which implies that there is a run c_J of $\boldsymbol{c}$, where

$J \subseteq [n]$ is an interval, such that $\boldsymbol{b}$ is obtained from $\boldsymbol{c}$ by deleting two symbols in c_J. Since $\boldsymbol{c} \in \{0,1\}^n$ is regular, by Remark 1, the length of c_J is at most $d \log n < \rho = 3d \log n$. Thus, 2) of Lemma 9 holds.

In the following, we suppose that c_{j_1} and c_{j_2} ($j_1 < j_2$) are in two different runs of $\boldsymbol{c}$. Specifically, suppose $j_1 \in J = [i_1, i_2] \subseteq [n]$ and $j_2 \in J' = [i'_1, i'_2] \subseteq [n]$ such that c_J and $c_{J'}$ are two different runs of $\boldsymbol{c}$. Since $j_1 < j_2$, then

$$i_1 \leq i_2 < i'_1 \leq i'_2.$$

We need to consider the following two cases.

Case 1: $i'_1 > i_2 + 1$. Since $j_2 \in J' = [i'_1, i'_2]$, then $j_2 \geq i'_1 > i_2 + 1$. For any $\{j'_1, j'_2\} \subseteq [n]$ such that $\boldsymbol{b} = c_{[n] \setminus \{j'_1, j'_2\}}$, it is easy to see that 2) of Lemma 11 can't hold. (Otherwise, there is an alternative substring $c_{[s_1, s_2]}$ of $\boldsymbol{c}$ of length ≥ 3 such that c_{j_1}, c_{s_1} are in the same run of $\boldsymbol{c}$ and $j_2 = s_1 + 1$, which implies that $s_1 = i_2$ and $j_2 = s_1 + 1 = i_2 + 1$, which contradicts to the fact that $j_2 \geq i'_1 > i_2 + 1$.) Therefore, 1) of Lemma 11 must hold, which implies that $j'_1 \in J = [i_1, i_2]$ and $j'_2 \in J' = [i'_1, i'_2]'$. Thus, 1) of Lemma 9 holds.

Case 2: $i'_1 = i_2 + 1$. We need to consider the following two subcases.

Case 2.1: $|J| \geq 2$ and $|J'| \geq 2$. Then for any $\{j'_1, j'_2\} \subseteq [n]$ such that $\boldsymbol{b} = c_{[n] \setminus \{j'_1, j'_2\}}$, it is easy to see that 2) of Lemma 11 can't hold because no such alternative substring $c_{[s_1, s_2]}$ of $\boldsymbol{c}$ can be found. Therefore, 1) of Lemma 11 must hold, which implies that $j'_1 \in J = [i_1, i_2]$ and $j'_2 \in J' = [i'_1, i'_2]'$. Thus, 1) of Lemma 9 holds.

Case 2.2: $|J| = 1$ or $|J'| = 1$. Without loss of generality, assume $|J| = 1$. Then $i_1 = i_2$ and $c_{i_1-1} c_{i_1} c_{i_1+1}$ is an alternative substring of $\boldsymbol{c}$. Let $c_{[\lambda_1, \lambda_2]}$ be the maximal alternative substring of $\boldsymbol{c}$ that contains $c_{i_1-1} c_{i_1} c_{i_1+1}$, where $[\lambda_1, \lambda_2] \subseteq [n]$ is an interval. Let $c_{[\lambda_0, \lambda_1]}$ (if $\lambda_1 > 1$) and $c_{[\lambda_2, \lambda_3]}$ (if $\lambda_2 < n$) be two runs of $\boldsymbol{c}$. For any $\{j'_1, j'_2\} \subseteq [n]$ such that $\boldsymbol{b} = c_{[n] \setminus \{j'_1, j'_2\}}$, by Lemma 11, we have $\{j'_1, j'_2\} \subseteq [\lambda_0, \lambda_3]$. Since $\boldsymbol{c} \in \{0,1\}^n$ is regular, by Remark 1, the length of the alternative substring $c_{[\lambda_1, \lambda_2]}$ of $\boldsymbol{c}$ is at most $d \log n$, and the lengths of the runs $c_{[\lambda_0, \lambda_1]}$, $c_{[\lambda_2, \lambda_3]}$ of $\boldsymbol{c}$ are both at most $d \log n$. Hence, the length of $c_{[\lambda_0, \lambda_3]}$ is at most $\rho = 3d \log n$. Thus, 2) of Lemma 9 holds.

By the above discussions, we proved that exact one of the two claims of Lemma 9 holds. ∎

As an example, suppose $\boldsymbol{c} = 011000101011110100$. We consider the following cases.

- If $\boldsymbol{b} = 0110101011110100$, then $\boldsymbol{b}$ can be obtained from $\boldsymbol{c}$ by deleting two symbols in the run $c_{[4,6]} = 000$.
- If $\boldsymbol{b} = 0110010011110100$, then $\boldsymbol{b}$ can be obtained from $\boldsymbol{c}$ by deleting one symbol in the run $c_{[4,6]} = 000$ and one symbol in the run $c_{[9,9]} = 1$. This case is an example of Case 1 in the proof of Lemma 9.
- If $\boldsymbol{b} = 0100101011110100$, then $\boldsymbol{b}$ can be obtained from $\boldsymbol{c}$ by deleting one symbol in the run $c_{[2,3]} = 11$ and one symbol in the run $c_{[4,6]} = 000$. This case is an example of Case 2.1 in the proof of Lemma 9.
- If $\boldsymbol{b} = 0110001011110100$, then $\boldsymbol{b}$ can be obtained from $\boldsymbol{c}$ by deleting two symbols in the substring $c_{[4,14]} = 00010101111$, which may be any of the following cases: i) one symbol in the run $c_{[4,6]} = 000$ and the symbol $c_7 = 1$; ii) the symbols c_i, c_{i+1} for $i \in \{7, 8, 9\}$; iii) one symbol in the run $c_{[11,14]} = 1111$ and the symbol $c_{10} = 0$. This case is an example of Case 2.2 in the proof of Lemma 9.

REFERENCES

[1] R. Heckel, G. Mikutis, and R. N. Grass, "A Characterization of the DNA Data Storage Channel," *Scientific Reports*, vol. 9, no. 1, pp. 9663, 2019. Available online at: https://doi.org/10.1038/s41598-019-45832-6

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

[3] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors (in Russian)," *Automatika i Telemkhanika*, vol. 161, no. 3, pp. 288-292, 1965.

[4] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," *IEEE Trans. Inform. Theory*, vol. 64, no. 5, pp. 3403-3410, 2018.

[5] R. Gabrys and F. Sala, "Codes correcting two deletions," *IEEE Trans. Inform. Theory*, vol. 65, no. 2, pp. 965-974, Feb 2019.

[6] J. Sima, N. Raviv, and J. Bruck, "Two deletion correcting codes from indicator vectors," *IEEE Trans. Inform. Theory*, vol. 66, no. 4, pp. 2375-2391, April 2020.

[7] V. Guruswami and J. Håstad, "Explicit two-deletion codes with redundancy matching the existential bound," *IEEE Trans. Inform. Theory*, vol. 67, no. 10, pp. 6384-6393, October 2021.

[8] J. Sima and J. Bruck, "Optimal k-Deletion Correcting Codes," in *Proc. ISIT*, 2019.

[9] J. Sima and J. Bruck, "On Optimal k-Deletion Correcting Codes," *IEEE Trans. Inform. Theory*, vol. 67, no. 6, pp. 3360-3375, June 2021.

[10] J. Sima, R. Gabrys, and J. Bruck, "Syndrome Compression for Optimal Redundancy Codes," in *Proc. ISIT*, 2020.

[11] J. Sima, R. Gabrys, and J. Bruck, "Optimal Systematic t-Deletion Correcting Codes," in *Proc. ISIT*, 2020.

[12] W. Song, N. Polyanskii, K. Cai, and X. He, "Systematic Codes Correcting Multiple-Deletion and Multiple-Substitution Errors," *IEEE Trans. Inform. Theory*, vol. 68, no. 10, pp. 6402-6416, October 2022.

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

[14] V. Levenshtein, "Asymptotically optimum binary code with correction for losses of one or two adjacent bits," *Problemy Kibernetiki*, vol. 19, pp. 293-298, 1967.

[15] A. Lenz, N. Polyanskii, "Optimal Codes Correcting a Burst of Deletions of Variable Length," in *Proc. ISIT*, 2021.

[16] V.I. Levenshtein, "Bounds for deletion/insertion correcting codes," in *Proc. ISIT*, 2002.

[17] G. M. Tenengolts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Inform. Theory*, vol. 30, no. 5, pp. 766-769, Sept. 1984.

[18] J. Sima, R. Gabrys, and J. Bruck, "Optimal codes for the q-ary deletion channel," in *Proc. ISIT*, 2020.

[19] K. Cai, Y. M. Chee, R. Gabrys, H. M. Kiah, and T. T. Nguyen, "Optimal codes correcting a single indel/edit for DNA-based data storage," 2019, Available online at: https://arxiv.org/abs/1910.06501

[20] S. Wang, J. Sima, and F. Farnoud, "Non-binary Codes for Correcting a Burst of at Most 2 Deletions," in *Proc. ISIT*, 2021.