Explicit Construction of q-ary 2-deletion Correcting Codes with Low Redundancy

Shu Liu, Ivan Tjuawinata, Chaoping Xing

***Abstract*—We consider the problem of efficient construction of q-ary 2-deletion correcting codes with low redundancy. We show that our construction requires less redundancy than any existing efficiently encodable q-ary 2-deletion correcting codes. Precisely speaking, we present an explicit construction of a q-ary 2-deletion correcting code with redundancy $5\log n+10\log\log n+3\log q+O(1)$ where q is assumed to be a constant with respect to n. Using a minor modification to the original construction, we obtain an efficiently encodable q-ary 2-deletion code that is efficiently list-decodable. Similarly, we show that our construction of list-decodable code requires a smaller redundancy compared to any existing list-decodable codes.**

To obtain our sketches, we transform a q-ary codeword to a binary string which can then be used as an input to the underlying base binary sketch. This is then complemented with additional q-ary sketches that the original q-ary codeword is required to satisfy. In other words, we build our codes via a binary 2-deletion code as a black-box. Finally we utilize the binary 2-deletion code proposed by Guruswami and Håstad to our construction to obtain the main result of this paper.

***Index Terms*—Insertion and Deletion, Efficient Construction, Error Correction**

Shu Liu is with the National Key Laboratory of Wireless Communications, University of Electronic Science and Technology of China, Chengdu 611731, China (email: shuliu@uestc.edu.cn).

The work of Shu Liu was supported in part by the National Key R&D Program of China under Grant 2023YFE0123900, 2022YFA1004900, in part by the National Natural Science Foundation of China under Grant 12271084, 12361141818, in part by Young Elite Scientists Sponsorship Program by CAST under Grant 2023QNRC001, in part by the National Key Laboratory of Wireless Communications Foundation under Grant IFN20230107.

Ivan Tjuawinata is with the Strategic Centre for Research on Privacy-Preserving Technologies and Systems, Nanyang Technological University, Singapore 637553 (email: ivan.tjuawinata@ntu.edu.sg).

This research is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

Chaoping Xing is with the School of Electronic Information and Electric Engineering, Shanghai Jiao Tong University, Shanghai, China (email: xingcp@sjtu.edu.cn).

The work of Chaoping Xing was supported in part by the National Natural Science Foundation of China under Grants 12031011.

I. Introduction

We consider the construction of a q-ary 2-deletion correcting code. A q-ary 2-deletion correcting code is a set of strings of a fixed length n over an alphabet Σ_q of size $q, \mathcal{C} \subseteq \Sigma_q^n$ such that for any two distinct codewords $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$, they do not have any common subsequence of length at least $n-2$. A natural question is whether we can efficiently construct such q-ary 2-deletion correcting code with as large size as possible. Equivalently, this can be formulated as the problem of designing efficient construction of q-ary 2-deletion correcting code with minimum amount of redundancy. Here we define the redundancy of the code to be the additional bits required during the encoding to facilitate the error correction. More specifically, we define the redundancy of a q-ary code $\mathcal{C} \subset \Sigma_q^n$ of size M to be $n\log_2 q - \log_2 M$. Denote by $R_D(q,n,2)$ (and $R_S(q,n,2)$, respectively) to be the smallest redundancy of q-ary 2-deletion (and substitution, respectively) error correcting codes of length n. Throughout this work, for simplicity, the notation $\log(\cdot)$ is used to denote the logarithm base

2. It was shown in [1] that

$$\begin{aligned} & 2\log n + 2\log q + o(\log q \log n) \\ \leqslant\ & R_D(q,n,2) \\ \leqslant\ & 4\log n + 2\log q + o(\log q \log n). \end{aligned}$$

It is then interesting to see if an efficient construction of q-ary 2-deletion correcting code with redundancy approaching the optimal redundancy provided above can be designed.

A. Related Work

There have been numerous studies on low redundancy Hamming metric codes with fixed Hamming distance (see [2] for instance). In particular, we have $R_S(q,n,2) = 2\log n + o(\log n)$ for $q = 2,3,4$ and $2\log n + o(\log n) \leqslant R_S(q,n,2) \leqslant \frac{7}{3}\log n + o(\log n)$ for $q > 4$.

There have been various works on the construction of t-deletion codes. The study of t-deletion code construction was first considered in the construction of the binary Varshamov-Tenengolts codes (VT codes for short) [3], [4] which was extended to q-ary codes in [5]. VT codes, which can correct one deletion error, was then further extended to the binary Helberg code [6], which can correct multiple deletions. Helberg code was further extended to be defined over alphabets of size q in [7]. All these constructions are based on the concept of sketch where codes are constructed by collecting all solutions to a predefined system of equations. Asymptotically, the redundancy of the resulting code equals the size of the corresponding sketch.

Although Hamming metric codes with logarithmic redundancy have been studied for many years, study on deletion correcting codes with logarithmic redundancy was initiated in [8] recently, which proposed binary codes capable of correcting multiple deletions. More specifically, they construct binary k-deletion codes with redundancy $O(k^2 \log k \log n)$. There have been some improvements on this construction, especially in the case when $k = 2$. In [9], a binary 2-deletion codes with sketch of size $7\log n + 6$ bits was proposed while a binary $8\log n + O(\log\log n)$ was proposed in [10]. A more general improvement on binary k-deletion codes was proposed with redundancy $8k\log n + o(\log N)$ in [11] as long as $k = o(\sqrt{\log\log n})$. A binary 2-deletion codes was more recently constructed with the help of both sketch and regularity assumption in [12] to produce a binary code of redundancy $4\log n + O(\log\log n)$. In contrast to the previous constructions, the work of [12] provides a smaller sketch size by requiring that the codewords satisfy a regularity assumption. In order to enforce such assumption while maintaining the rate of the code, an explicit encoding process to regular strings is proposed. Recently, there have also been some works on the construction of codes capable of correcting both deletion and substitution [13], [14]. In their work [14], a binary t-deletion correcting code has also been proposed with redundancy $(4t-1)\log n + o(\log n)$.

There have also been some works on the construction of q-ary 2-deletion codes following the line of work previously discussed. In [15], a q-ary 2-deletion codes was constructed with existing binary 2-deletion codes as the underlying base. In the proposed construction, an additional $3\log n + O_q(\log\log n)$ bits of sketch is required. The work has also considered a simpler construction which produces a code with redundancy $6\log n + O_q(\log\log n)$ that can list decode any 2 deletions with list size 2. In [16], they consider the case when $q > 2$ is even and also constant with respect to n. In such case, they proposed a q-ary 2-deletion codes with redundancy $5\log n + O(\log q \log\log n)$. In [14], a q-ary t-deletion s-substitution correcting systematic code was proposed where q is at most logarithmic with respect to the code length. In such construction, when s is set to be 0, the construction provides a q-ary t-deletion correcting code of redundancy $4t\log_q n + o(\log_q n)$.

B. Our Result

In this work, we consider a general construction of q-ary sketch which is based on sketches of binary 2-deletion codes. Based on this construction, we show that the resulting construction can be used to con-

struct q-ary 2-deletion codes with lower redundancy compared to the previously best known result. We further note that the encoding and decoding algorithm of such codes are linear with respect to the code length and the square of the logarithm of the alphabet size (Theorem IV.2). Such construction is also used to construct q-ary deletion codes that can list decode against any 2 deletions with smaller redundancy compared to the previously best known result. Again, we show that the encoding and decoding complexities of such code is linear with respect to the code length and the square of the logarithm of the alphabet size (Theorem IV.7). These two results are summarized in the following.

Theorem I.1. *For any positive integer $q > 2$, there exists an explicit and efficiently encodable q-ary 2-deletion correcting code $\mathcal{C} \subseteq \Sigma_q^N$ of size q^n with $N \leq n + 8\log n + 3\log q + O(\log\log n)$ with redundancy $8\log n + 3\log q + O(\log\log n)$ with encoding complexity $O(n)$ and decoding complexity $O(n + \log^2 q)$.*

Furthermore, there is an explicit and efficiently encodable q-ary 2-deletion correcting code $\mathcal{C} \subseteq \Sigma_q^N$ of size q^n with $N \leq n + 4\log n + 3\log q + O(\log\log n)$ and redundancy $4\log n + 3\log q + O(\log\log n)$ with encoding complexity $O(n)$ which is list decodable against any 2-deletions with list size 2 and decoding complexity $O(n + \log^2 q)$.

Lastly, there is an explicit and efficiently encodable q-ary 2-deletion correcting code $\mathcal{C} \subseteq \Sigma_q^N$ of size q^n with $N \leq n + 3\log n + 2\log q + O(\log\log n)$ with redundancy $3\log n + 2\log q + O(\log\log n)$ with encoding complexity $O(n)$ which is list decodable against any 2-deletions with list size 4 and decoding complexity $O(n + \log^2 q)$.

In order to take advantage of the construction of the binary 2-deletion codes proposed in [12], we further modify our construction to allow for the regularity assumption to again be satisfied. This results in q-ary deletion codes with better redundancy compared to the general construction we have previously discussed (Theorem V.4). Here we note that although the redundancy of the constructed codes is reduced, it comes with the increase of encoding and decoding complexity. More specifically, the resulting codes have encoding complexity of $\mathrm{poly}(n)$ while the efficiency of the decoding complexity relies on the efficiency of the inversion of the encoding process. The main result of this paper is given below.

Theorem I.2. *For any positive integer $q > 2$, there exists an explicit and efficiently encodable q-ary 2 deletion correcting code of length n with redundancy $5\log n + 10\log\log n + 3\log q + O(1)$.*

Furthermore, there exists an explicit and efficiently encodable q-ary deletion code of length n with redundancy $4\log n + 10\log\log n + 2\log q + O(1)$ that can be list decoded against 2 deletions with list size of at most 2.

In both cases, the encoding complexity is $\mathrm{poly}(n)$. Furthermore, assuming that the complexity of the inverse of the message encoding algorithm is $T_{\Pi_q,D}(n)$, then the decoding complexity is $2T_{\Pi_q,D}(n) + O(\log^2 q + n)$.

Remark I.3. As we mentioned in the earlier subsection, the lowest redundancy of q-ary 2 deletion correcting codes of length n with explicit construction in literature is given by $5\log n + O(\log q \log\log n)$ in [16]. It is easy to see that our redundancy is smaller than that of [16].

C. Our Technique

In order to construct our sketch, we define a function that transforms our q-ary codewords to a binary string which can then be used as an input to the underlying base binary sketch. This is then complemented with additional q-ary sketches that the original q-ary codeword is required to satisfy. Having these two steps of sketches, the decoding algorithm can also then be designed in two separate steps. First, based on a received q-ary string, the same transformation function can be applied to generate the corresponding binary string, which can then be shown to be obtainable by deleting 2 elements from the binary string obtained from the sent codeword.

This, together with the binary sketches, allows us to recover the original binary string corresponding to the sent codeword. We can then utilize such binary string as well as the q-ary sketches to recover the original codeword. An observation to be made in such decoding algorithm is that the last step of the decoding algorithm, which requires $2\log q$-bits of sketches can be removed and this results in a list decoding algorithm with list size of at most 2. This provides us with the list-decoding variant of our construction with smaller sketch size.

In order to utilize the binary 2-deletion code proposed in [12], we require that our message is encoded to a q-ary string with some regularity assumption. Similar to [12], we show that such encoding algorithm can be designed explicitly and the encoding process has a sufficiently large domain size to maintain the redundancy size.

D. Organization

The remainder of the paper is organized as follows. In Section II, we review some basic definitions and existing results that will be useful in the discussion in the following sections. Section III focuses on a general construction of q-ary 2-deletion correcting code from existing binary 2-deletion correcting code with specific form. Such construction is then realized using previously proposed binary 2 deletion correcting codes, the resulting redundancy is considered and compared in Section IV. In the same section, we also consider a variant of such construction to allow for list-decodable 2-deletion codes with smaller redundancy. Lastly, a specific construction based on the binary 2-deletion correcting code proposed in [12] is considered and analyzed in Section V.

II. Preliminary

Let q be a prime number, n be positive integers such that q is constant with respect to n and for any positive integer m, let $\Sigma_m = \{0, 1, \cdots, m-1\} \subseteq \mathbb{Z}$. Given a string $\mathbf{x} = (x_1, \cdots, x_n)$ and a positive integer $m \leqslant n$, a string $\mathbf{y} = (y_1, \cdots, y_m)$ is said to be a subsequence of $\mathbf{x}$ if there exists $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$ such that $y_j = x_{i_j}$ for $j = 1, \cdots, m$. Furthermore, $\mathbf{y}$ is said to be a substring of $\mathbf{x}$ if it is a subsequence of $\mathbf{x}$ and for any $j = 2, \cdots, m, i_j = i_{j-1} + 1$.

Definition 1. 1) *Let $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n) \in \Sigma_q^n$. Define $\mathbf{a} = (a_1, \cdots, a_n) \in \Sigma_2^n$ such that for $i = 1, \cdots, n$,*

$$a_i = \begin{cases} 1, & \text{if } i = 1 \text{ or } \alpha_i \geqslant \alpha_{i-1} \\ 0, & \text{otherwise} \end{cases}.$$

We denote such map by $\varphi_n : \Sigma_q^n \to \Sigma_2^n$ where it applies to any length n. We observe that φ_n can be calculated in $O(n)$ time. Here when n is clear in the context or the discussion applies to any length n, we write φ instead of φ_n.

2) *Suppose that $\boldsymbol{\alpha} \in \Sigma_q^n$ is associated to $\mathbf{a} = (a_1, \cdots, a_n) \in \Sigma_2^n$ by the definition above. Recall that a run in $\mathbf{a}$ is defined to be a substring of $\mathbf{a}$ with the same symbol. More specifically, $\mathbf{a}$ has a run from position i to position j for some $1 \leqslant i \leqslant j \leqslant n$ if $a_i = a_{i+1} = \cdots = a_j \in \Sigma_2$, either $i = 1$ or $a_{i-1} = 1 - a_i$ and either $j = n$ or $a_{j+1} = 1 - a_j$. We can define $\mathbf{a}$ as a sequence of runs. More specifically, assuming that $\mathbf{a}$ has r runs, there exists $b \in \Sigma_2$ such that $\mathbf{a} = (1^{\ell_1} \| 0^{\ell_2} \| \cdots \| b^{\ell_r})$ where $\ell_i > 0$ and $\sum_{i=1}^r \ell_i = n$ and*

$$b = \begin{cases} 1, & \text{if } r \text{ is odd} \\ 0, & \text{otherwise} \end{cases}.$$

For non-negative i, we define

$$j_i = \begin{cases} 0, & \text{if } i = 0 \\ \ell_i + j_{i-1}, & \text{otherwise} \end{cases}.$$

For $i = 1, \cdots, r$, we define the i-th run of $\boldsymbol{\alpha}$ associated to $\mathbf{a}$ by the vector $(\alpha_{j_{i-1}+1}, \cdots, \alpha_{j_i})$. It is easy to see that for any $i = 1, \cdots, r$, the i-th run of $\boldsymbol{\alpha}$ is either decreasing or non-decreasing. Lastly, we define $\beta_i = \alpha_{j_{i-1}+1} + \alpha_{j_{i-1}+2} + \cdots + \alpha_{j_i}$. In the following, given $\boldsymbol{\alpha} \in \Sigma_q^n$ and $\mathbf{a} = \varphi_n(\boldsymbol{\alpha})$, we denote by $r_{\boldsymbol{\alpha}}$ the number of runs of $\mathbf{a}$. Furthermore, for $i = 1, \cdots, r$, we call β_i by the i-th run sums *of $\boldsymbol{\alpha}$. We again note that such*

β_i can be computed in $O(n)$ given the value of $\varphi_n(\boldsymbol{\alpha})$.

Next, we provide an observation on φ with respect to a deletion operation.

Claim II.1 (See for example [5])**.** *Let m be a positive integer, $\boldsymbol{\alpha} \in \Sigma_q^m$ and $\mathbf{a} = \varphi_m(\boldsymbol{\alpha}) \in \Sigma_2^m$. Suppose that $\boldsymbol{\alpha}' \in \Sigma_q^{m-1}$ is a subsequence of $\boldsymbol{\alpha}$ by deleting one of its entries and suppose that $\mathbf{a}' = \varphi_{m-1}(\boldsymbol{\alpha}') \in \Sigma_2^{m-1}$. Then $\mathbf{a}'$ is a subsequence of $\mathbf{a}$.*

Note that by applying Claim II.1 multiple times, we obtain the following corollary.

Corollary II.2. *Let $m < n$ be a positive integer, $\boldsymbol{\alpha} \in \Sigma_q^n$ and $\mathbf{a} = \varphi_n(\boldsymbol{\alpha})$. Suppose that $\boldsymbol{\alpha}' \in \Sigma_q^m$ is a subsequence of $\boldsymbol{\alpha}$ by deleting $n-m$ of its entries and suppose that $\mathbf{a}' = \varphi_m(\boldsymbol{\alpha}')$. Then $\mathbf{a}'$ is a subsequence of $\mathbf{a}$.*

We will briefly discuss some results in [12] that will be useful in our discussion. First, we discuss the definition of regular binary string.

Definition 2. *Let d be a positive integer. We say that a string $\mathbf{x} \in \Sigma_2^n$ is d-regular if each substring of $\mathbf{x}$ of length at least $d\log_2 n$ contains both 00 and 11 as substrings.*

We can further extend this definition to a q-ary string. More specifically, we say that $\mathbf{x} \in \Sigma_q^n$ is d-regular if its associated binary string $\mathbf{a} \in \Sigma_2^n$ is d-regular. This corresponds to the requirement that any substring $\mathbf{x}'$ of $\mathbf{x}$ of length at least $d\log_2 n$ must contain a substring of length three (y_1, y_2, y_3) such that $y_1 > y_2 > y_3$. Furthermore, we also require that it contains a substring (z_1, z_2, z_3) such that $z_1 \leqslant z_2 \leqslant z_3$ unless $\mathbf{x}'$ contains the first entry of $\mathbf{x}$. In the case that $\mathbf{x}'$ contains the first entry of $\mathbf{x}$ and it does not contain any substring (z_1, z_2, z_3) such that $z_1 \leqslant z_2 \leqslant z_3$, then we require that $x_1 \leqslant x_2$ to allow for the first two entries of $\mathbf{a}$ to all be 1.

In this work, we consider constructions of codes with the use of sketch functions. We note that by the Pigeonhole Principle, the construction of a q-ary sketch function with output size of ℓ-bits that can be used to correct up to 2 deletions implies the existence of a q-ary 2-deletion codes with redundancy of at most ℓ. When explicitness of the code is required, the redundancy required is larger. More specifically, similar to the discussion in [12, Lemma 2.1] and [15, Lemma 1], we can obtain the following relation.

Lemma II.3. *Fix an integer $k \geq 1$. Let $s : \Sigma_q^n \to \Sigma_2^{c_1 \log n + c_2 \log q + O(1)}$ be an efficiently computable function such that for any $\mathbf{x} \in \Sigma_q^n$, given $s(\mathbf{x})$ and any $\mathbf{y} \in \Sigma_q^{n-2}$, a subsequence of $\mathbf{x}$, the original $\mathbf{x}$ can be efficiently and uniquely recovered. Then there exists an efficiently encodable and decodable code $\mathcal{C} \subseteq \Sigma_q^N$ of size q^n with length $N \leq n + c_1 \log n + c_2 \log q + O(\log\log n)$ such that $\mathcal{C}$ is a 2-deletion correcting codes.*

Intuitively, this can be achieved by the use of the given q-ary sketch s as well as a binary sketch $\bar{s}$ with output length $O(\log(n'))$ given any input length n', which can be satisfied by sketches proposed in [8], [10], [9]. The encoding is then done by appending the message $\mathbf{m}$ by the sketch $s(\mathbf{m})$ followed by repeating the sketch $\bar{s}(s(\mathbf{m}))$ three times. The decoding can then be done by first recovering $\bar{s}(s(\mathbf{m}))$, followed by using $\bar{s}(s(\mathbf{m}))$ to recover $s(\mathbf{m})$. Lastly, having $s(\mathbf{m}), \mathbf{m}$ can then be recovered.

We can then see that the encoding complexity equals the complexity to calculate s given an input of length n and to calculate $\bar{s}$ with input of length $c_1 \log n + c_2 \log q + O(1)$. Note that since $\bar{s}$ is assumed to be efficiently computable, its complexity is polynomial in the input length. Hence in our case, the complexity is polynomial in $O(\log n + \log q)$. Hence the encoding complexity asymptotically equals the complexity of calculating s with input length n, which is again assumed to be efficient.

Similarly, the decoding process contains three steps. The first step includes the decoding process of repetition code with length $O(\log(\log n + \log q))$ which is then sub-linear with respect to n and q. The second step is the decoding process of $\bar{s}$ with input length $O(\log n + \log q)$. Since the decoding process of $\bar{s}$ is polynomial in its length by assumption, its

complexity in our application is hence $\mathrm{poly}(\log n + \log q)$. The last step is the decoding process of s with input length n. Hence, the dominating factor of the decoding complexity is the complexity of the decoding process of s with input length n.

Due to the relation described in Lemma II.3, in the remainder of this work, we focus on the construction of such sketch function and its output length, which can be directly seen as the asymptotic redundancy of the resulting explicit q-ary 2-deletion correcting code.

III. General Construction

Here we construct a q-ary sketch function that can correct up to 2 deletions from an existing binary sketch function capable of correcting up to 2 deletions. Suppose that there exists a sketch function $\bar{s}$ such that for any $\mathbf{x} \in \Sigma_2^n$, given $\bar{s}(\mathbf{x})$ and any subsequence $\mathbf{y} \in \Sigma_2^{n-2}$ of $\mathbf{x}$, the original $\mathbf{x}$ can be uniquely determined.

Let $\tilde{s}$ be a function that takes a q-ary string $\mathbf{x}$ as an input, computes $\bar{\mathbf{x}} = \varphi(\mathbf{x})$ and outputs $\bar{s}(\bar{\mathbf{x}})$. We further define the following three sketches $s^{(1)}, s^{(2)}, s^{(3)} : \Sigma_q^n \to \mathbb{Z}$ such that for $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n) \in \mathbb{Z}_q^n$ with $r_{\boldsymbol{\alpha}}$ runs and run sums $\beta_1, \cdots, \beta_{r_{\boldsymbol{\alpha}}}$,

1) $s^{(1)}(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i$,
2) $s^{(2)}(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i^2$, and
3) $s^{(3)}(\boldsymbol{\alpha}) = \sum_{i=1}^{r_{\boldsymbol{\alpha}}} i\beta_i$.

It is easy to see that for the three sketches defined above, they can be computed in $O(n)$.

Lemma III.1. *For any $\boldsymbol{\alpha} \in \Sigma_q^n$, given the values of $s^{(1)}(\boldsymbol{\alpha}) \pmod q, s^{(2)}(\boldsymbol{\alpha}) \pmod q$, $s^{(3)}(\boldsymbol{\alpha}) \pmod{nq}$, $\tilde{s}(\boldsymbol{\alpha})$, and any subsequence $\boldsymbol{\alpha}' = (\alpha'_1, \cdots, \alpha'_{n-2}) \in \Sigma_q^{n-2}$ of $\boldsymbol{\alpha}$, the original $\boldsymbol{\alpha}$ can be uniquely recovered.*

Proof. Let $\mathbf{a}' = (a'_1, \cdots, a'_{n-2}) = \varphi(\boldsymbol{\alpha}')$. By Corollary II.2, we have that $\mathbf{a}'$ is a subsequence of $\mathbf{a}$. So we have $\mathbf{a}' \in \Sigma_2^{n-2}$, a subsequence of $\mathbf{a}$ obtained by deleting two entries. Furthermore, we also have $\tilde{s}(\boldsymbol{\alpha}) = \bar{s}(\mathbf{a})$. Hence by assumption, given these, we can uniquely determine $\mathbf{a} = (1^{\ell_1} \| 0^{\ell_2} \| \cdots \| 1^{\ell_{r_{\boldsymbol{\alpha}}}})$ where $b = 1$ if $r_{\boldsymbol{\alpha}}$ is odd and $b = 0$ otherwise. This means that we also obtain the runs of $\mathbf{a}$ as well as their lengths ℓ_j where $\ell_i > 0$ and $\sum_{i=1}^{r_{\boldsymbol{\alpha}}} \ell_i = n$. In particular, we also obtain the values of $1 \leqslant k_1 \leqslant k_2 \leqslant r_{\boldsymbol{\alpha}}$ that determines the runs where the two entries of $\boldsymbol{\alpha}$ are deleted from.

Suppose that the deleted symbols from $\boldsymbol{\alpha}$ in the k_1-th and k_2-th runs are $v_1, v_2 \in \Sigma_q$ respectively. By assumption, we know the values of $s^{(i)} \equiv s^{(i)}(\boldsymbol{\alpha}) \pmod q$ for $i = 1, 2$. Given $\boldsymbol{\alpha}'$, we can also compute $\hat{s}^{(i)} \equiv s^{(i)}(\boldsymbol{\alpha}') \pmod q$. Then for $i = 1, 2$, we have $v_1^i + v_2^i \equiv \Delta_i \pmod q$ where $\Delta_i \triangleq s^{(i)} - \hat{s}^{(i)} \pmod q$. Hence we have the following system of equations

$$\begin{cases} v_1 + v_2 \equiv \Delta_1 \pmod q \\ v_1^2 + v_2^2 \equiv \Delta_2 \pmod q \end{cases} \tag{1}$$

Squaring the first equivalence and subtracting the second from it yields $2v_1v_2 \equiv \Delta_1^2 - \Delta_2 \pmod q$. Since q is an odd prime, $2^{-1} \pmod q$ exists and define $\Delta_0 \in \mathbb{Z}_q$ such that $\Delta_0 \equiv 2^{-1}\left(\Delta_1^2 - \Delta_2\right) \pmod q$. Hence treating the equations over $\mathbb{Z}_q$, we have $v_1 + v_2 = \Delta_1$ and $v_1v_2 = \Delta_0$. This implies that v_1 and v_2 are roots of the quadratic equation $x^2 - \Delta_1 x + \Delta_0 = 0$. Define $S = \{a \in \mathbb{Z}_q : \exists b \in \mathbb{Z}_q, b^2 \equiv a \pmod q\}$ be the set of quadratic residues modulo q. Note that by definition, given that $b^2 \equiv a \pmod q$, we must also have $(q-b)^2 \equiv a \pmod q$. Hence for any $a \in S$, there exists a unique $b \in \left\{0, 1, \cdots, \frac{q-1}{2}\right\} \subseteq \mathbb{Z}_q$ such that $b^2 \equiv a \pmod q$. We denote such b, if it exists, by $\sqrt{a}$. Now note that under the condition that $b \triangleq \sqrt{\Delta_1^2 - 4\Delta_0}$ exists, we have that v_1, v_2 are different elements of the set $\left\{2^{-1}(\Delta_1 + b) \pmod q, 2^{-1}(\Delta - b) \pmod q\right\}$. However, note that by definitions of Δ_0 and Δ_1, we have $\Delta_1^2 - 4\Delta_0 = (v_1 - v_2)^2$. Hence b is the unique element in $\{(v_1 - v_2) \pmod q, (v_2 - v_1) \pmod q\} \cap \left\{0, 1, \cdots, \frac{q-1}{2}\right\}$. This shows that provided that the square root function $\sqrt{\cdot}$ can be efficiently computed, we can compute $\{v_1, v_2\}$ efficiently.

Note that although we have learned the set $\{v_1, v_2\}$, which we denote by θ_1 and θ_2, unless

$\theta_1 = \theta_2$, we need to determine whether $(v_1, v_2) = (\theta_1, \theta_2)$ or $(v_1, v_2) = (\theta_2, \theta_1)$. Here with the help of $s^{(3)}(\boldsymbol{\alpha})$ we prove that given $\{\theta_1, \theta_2\}$ and k_1, k_2, the run numbers where the two deleted entries are from, we can uniquely recover $\boldsymbol{\alpha}$. Note that if $k_1 = k_2$, since $\theta_1 \neq \theta_2$, there is a unique way to place θ_1 and θ_2 in the k_1-th run to satisfy the run definition. Hence, it remains to show that with the possible help of $s^{(3)}(\boldsymbol{\alpha})$, we can uniquely recover $\boldsymbol{\alpha}$ from $\{\theta_1, \theta_2\}$ and k_1, k_2 given that $k_1 \neq k_2$ and $\theta_1 \neq \theta_2$. Without loss of generality, when seen as positive integers, assume that $\theta_1 < \theta_2$ and $k_1 < k_2$. Note that here $\theta_2 - \theta_1 < q$ and $k_2 - k_1 < n$. Hence $0 < (\theta_2 - \theta_1)(k_2 - k_1) < nq$. We divide the case depending whether the number of runs of $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ are different.

- Case 1 : Both the k_1-th run and k_2-th run also exist in $\boldsymbol{\alpha}'$. This implies that after the re-insertion, both k_1-th and k_2-th run contain elements from $\boldsymbol{\alpha}'$. For $i = 1, 2$, let $\boldsymbol{\alpha}^{(i)}$ be the word obtained from $\boldsymbol{\alpha}'$ by adding θ_1 to the k_i-th run of $\boldsymbol{\alpha}'$ and θ to the k_{3-i}-th run of $\boldsymbol{\alpha}'$. Then $s^{(3)}(\boldsymbol{\alpha}^{(1)}) \equiv B + k_1\theta_1 + k_2\theta_2 \pmod{nq}$ while $s^{(3)}(\boldsymbol{\alpha}^{(2)}) \equiv B + k_1\theta_2 + k_1\theta_1 \pmod{nq}$. Hence $s^{(3)}(\boldsymbol{\alpha}^{(1)}) \equiv s^{(3)}(\boldsymbol{\alpha}^{(2)}) \pmod{nq}$ if and only if $(k_2 - k_1)(\theta_2 - \theta_1) \equiv 0 \pmod{nq}$. However, this is impossible since $0 < (k_2 - k_1)(\theta_2 - \theta_1) < nq$. Hence in this case, there can only be one of the possibilities that can have $s^{(3)}(\boldsymbol{\alpha}^{(i)}) = s^{(3)}(\boldsymbol{\alpha})$.
- Case 2 : Both the k_1-th run and k_2-th run did not exist in $\boldsymbol{\alpha}'$. Note that in this case, after the re-insertion, both k_1-th and k_2-th run are of length 1. Note that because of this, the actual position of the deleted elements are determined, say at $i_1 < i_2$. Hence we have that the first symbol is inserted between α'_{i_1-1} and α'_{i_1} while the second symbol is inserted between α'_{i_2-2} and α'_{i_2-1}. Note that if at least one of $\boldsymbol{\alpha}^{(1)}$ or $\boldsymbol{\alpha}^{(2)}$ does not satisfy the required value of $\mathbf{a}$, we can directly eliminate such value from the possible value of $\boldsymbol{\alpha}$. So we assume that both $\mathbf{a} = \varphi(\boldsymbol{\alpha}^{(1)}) = \varphi(\boldsymbol{\alpha}^{(2)})$. Then $s^{(3)}(\boldsymbol{\alpha}^{(i)}) \equiv B + k_1\theta_1 + 2\sum_{j=i_1}^{i_2-2} \alpha'_i + k_2\theta_{3-i} + 4\sum_{j=i_2-1}^{n-2} \alpha'_i \pmod{nq}$ for $i = 1, 2$. So we again have that $s^{(3)}(\boldsymbol{\alpha}^{(1)}) \equiv s^{(3)}(\boldsymbol{\alpha}^{(2)}) \pmod{nq}$ if and only if $(k_2 - k_1)(\theta_2 - \theta_1) \equiv 0 \pmod{nq}$ which is again impossible since $0 < (k_2 - k_1)(\theta_2 - \theta_1) < nq$.
- Case 3 : Suppose that the k_1-th run did not exist in $\boldsymbol{\alpha}'$ while the k_2-th run did. We again note that if there exists i such that $\varphi(\boldsymbol{\alpha}^{(i)}) \neq \mathbf{a}$, we can directly eliminate such possibility for $\boldsymbol{\alpha}$. So we consider the case when $\mathbf{a} = \varphi(\boldsymbol{\alpha}^{(1)}) = \varphi(\boldsymbol{\alpha}^{(2)})$. Using a similar argument as before, $s^{(3)}(\boldsymbol{\alpha}^{(i)}) \equiv B + k_1\theta_1 + 2\sum_{j=i_1}^{n-2} \alpha'_j + k_2\theta_{3-i} \pmod{nq}$. So $s^{(3)}(\boldsymbol{\alpha}^{(1)}) \equiv s^{(3)}(\boldsymbol{\alpha}^{(2)}) \pmod{nq}$ if and only if $(k_2 - k_1)(\theta_2 - \theta_1) \equiv 0 \pmod{nq}$ which cannot happen by our assumption of k_1, k_2, θ_1 and θ_2.
- Case 4 : Lastly, suppose that k_1-th run existed in $\boldsymbol{\alpha}'$ while the k_2-th run did not. Using a similar argument as before, $s^{(3)}(\boldsymbol{\alpha}^{(i)}) \equiv B + k_1\theta_1 + k_2\theta_{3-i} + 2\sum_{j=i_2-1}^{n-2} \alpha'_j \pmod{nq}$. So $s^{(3)}(\boldsymbol{\alpha}^{(1)}) \equiv s^{(3)}(\boldsymbol{\alpha}^{(2)}) \pmod{nq}$ if and only if $(k_2 - k_1)(\theta_2 - \theta_1) \equiv 0 \pmod{nq}$ which cannot happen by our assumption of k_1, k_2, θ_1 and θ_2 which concludes our proof.

□

It is easy to see that by definition, the sketch we use, which consists of $\bar{s}, s^{(1)}(\boldsymbol{\alpha}) \pmod{q}, s^{(2)}(\boldsymbol{\alpha}) \pmod{q}$ and $s^{(3)}(\boldsymbol{\alpha}) \pmod{nq}$ can be computed in $T_{\bar{s},E}(n) + O(n)$ where $T_{\bar{s},E}(n)$ is the complexity of calculating $\bar{s}$. Hence our sketch is efficiently computable provided that $\bar{s}$ is efficiently computable.

Next, we consider the efficiency of the decoding algorithm implied by our proof above. After the decoding of $\mathbf{a}$ from $\mathbf{a}'$, which uses the decoding algorithm of the underlying binary sketch, the values of v_1 and v_2 are determined by solving the System (1), whose complexity is asymptotically equal to the complexity of calculating the square root function over $\mathbb{Z}_q$. Hence, if a square root algorithm, for example, the Tonelli-Shanks algorithm is used [17], [18], its complexity is $O(\log^2 q)$. Having k_1, k_2, v_1

and v_2, the last step of the decoding algorithm is to identify whether to insert v_1 or v_2 at the k_1-th or k_2-th run. This can be done by considering both possibilities and calculate the value of $s^{(3)}$ for both possibilities. Hence, it is easy to see that such calculation can be done in time linear with respect to n. So the overall decoding algorithm takes time $T_{\bar{s},D}(n) + O(\log^2 q + n)$ where $T_{\bar{s},D}(n)$ is the decoding complexity of the underlying binary sketch.

This results in our the following efficiently computable q-ary sketch function.

Theorem III.2. *Let $\bar{s} : \Sigma_2^n \to \Sigma_2^{\ell(n)}$ be an efficiently computable sketch function such that for any $\mathbf{a} \in \Sigma_2^n$, given $\bar{s}(\mathbf{a})$ and any subsequence $\mathbf{a}' \in \Sigma_2^{n-2}$ of $\mathbf{a}$, the original $\mathbf{a}$ can be uniquely and efficiently recovered. We further assume that $\bar{s}$ can be computed in time $T_{\bar{s},E}(n)$ while the error correction takes $T_{\bar{s},D}(n)$. Then for any positive integer $q > 2$, there exists an explicit sketch $s : \Sigma_q^n \to \Sigma_2^{\log n + 3\log q}$ that can be efficiently computed consisting of $s^{(1)}, s^{(2)}$ and $s^{(3)}$ such that for any $\boldsymbol{\alpha} \in \Sigma_q^n$, given $\bar{s}(\varphi(\boldsymbol{\alpha})), s(\boldsymbol{\alpha})$ and any subsequence $\boldsymbol{\alpha}' \in \Sigma_q^{n-2}$ of $\boldsymbol{\alpha}$, the original $\boldsymbol{\alpha}$ can be uniquely and efficiently recovered. In other words, the total sketch size is $\ell(n) + \log n + 3\log q$-bits. Furthermore, the sketch calculation and the error correction complexity are $T_{\bar{s},E}(n) + O(n)$ and $T_{\bar{s},D}(n) + O(\log^2 q + n)$ respectively.*

Remark III.3. By Lemma II.3, such sketch implies the existence of a q-ary 2-deletion code of length $N \leqslant n + \ell(n) + \log n + 3\log(q) + O(\log\log n)$ with encoding complexity $T_{\bar{s},E}(n) + O(n)$ and decoding complexity $T_{\bar{s},D}(n) + O(\log^2 q + n)$.

IV. Construction Realization and Comparison with Existing Construction

In this section, we use the construction shown in Theorem III.2 with some existing binary sketch functions to obtain the actual output length of the resulting construction. Then, we compare such construction with existing q-ary sketch functions. Furthermore, we also consider the use of our construction and its variant to construct q-ary sketch functions capable of list decode up to two deletions.

A. Instantiation using Existing Binary Two-Deletion Sketch Function and its Comparison

In the following we consider the length of the resulting sketch function based on various existing two-deletion codes proposed in [10], [9], [14]. We note that the code and sketch described in [9] has $O(n)$ encoding and decoding algorithms, the ones described in [14] has $O(n^5)$ and $O(n^3)$ encoding and decoding algorithms while the one proposed in [10] does not come with the encoding and decoding complexities. Hence, we no longer have the efficiency assumption for the construction based on [10].

Corollary IV.1.

- *Let $\mathcal{C}_0^{(1)} \subseteq \Sigma_2^n$ be the binary two-deletion correcting code proposed in [10] which has a sketch of size $8\log n + O(\log\log n)$ bits. Then we can compute a sketch function $s_0^{(1)}$ with total output size $9\log n + O(\log\log n) + 3\log q$ bits that can be used to correct up to 2 deletions.*
- *Let $\mathcal{C}_0^{(2)} \subseteq \Sigma_2^n$ be the binary two-deletion correcting codes proposed in [14, Construction 2] which has a sketch of size $7\log n + o(\log n)$ bits.*
  Then we can efficiently compute a sketch function $s_0^{(2)}$ with total output size $8\log n + 3\log q + o(\log n)$ bits that can be used to efficiently correct up to 2 deletions. Moreover, the sketch can be computed in $O(n^5)$ while the error correction can be done in $O(\log^2 q + n^3)$.
- *Let $\mathcal{C}_0^{(3)} \subseteq \Sigma_2^n$ be the binary two-deletion correcting codes proposed in [9] which has a sketch of size $7\log n + 6$ bits.*
  Then we can efficiently compute a sketch function $s_0^{(3)}$ with total output size $8\log n + 3\log q + 6$ bits that can be used to efficiently correct up to 2 deletions. Moreover, the sketch can be computed in $O(n)$ while the error correction can be done in $O(\log^2 q + n)$.

Note that the second point of Corollary IV.1 yields an efficiently encodable and decodable q-ary 2-deletion code, which will be summarized in the following theorem.

Theorem IV.2. *There is an explicit q-ary 2-deletion correcting code $\mathcal{C} \subseteq \Sigma_q^N$ of size q^n with $N \leq n + 8\log n + 3\log q + O(\log\log n)$ with redundancy $8\log n + 3\log q + O(\log\log n)$ with encoding complexity $O(n)$ and decoding complexity $O(n+\log^2 q)$.*

Remark IV.3. We note that the construction discussed above cannot be directly used with the binary sketch proposed in [12]. This is because such construction requires the binary string to be regular, which is not a requirement that can be expressed as a sketch. So based on Corollary IV.1, the smallest sketch from our initial construction requires output length of $8\log n + 3\log q + 6$ bits.

Note that when $q > 2$ is even and constant with respect to n, a construction of a q-ary sketch function that can be used to correct up to 2 deletion with output $5\log n + (16\log q + 10)\log\log n$ bits was proposed in [16]. It is easy to see that the regime of q considered in our work is different where we require q to be an odd prime. Furthermore, in general, our best sketch requires larger output compared to the one provided in [16]. However, our construction is still efficient even when q is no longer constant with respect to n. Furthermore, when q is sufficiently large, the output length of our proposed sketch function is smaller compared to the extension of the construction provided in [16].

A similar construction of q-ary 2-deletion sketch function was proposed in [15]. Given an underlying binary sketch capable to correct 2 deletions with output length ℓ bits, their proposed construction provides a q-ary sketch capable to correct 2 deletions with output length $\ell + 3\log n + O_q(\log\log n)$ bits. It is easy to see that our resulting sketch has a smaller output length of $\ell + \log n + 3\log q$ bits.

Compared to the q-ary 2-deletion correcting code implied by [14, Construction 3], we note that it has the same asymptotic redundancy. However, our resulting construction has a better encoding and decoding complexity, which is $O(n)$ encoding and $O(n)$ decoding complexities instead of $O(n^5)$ encoding and $O(n^3)$ decoding complexities.

B. Construction of Sketch Function for List Decoding from Lemma III.1 and its Simplification

Note that a q-ary sketch function that can be used to list decode against 2-deletion can be constructed using a similar approach as Lemma III.1. More specifically, if the underlying binary sketch function $\bar{s}$ is list-decodable against 2-deletion, our resulting sketch can also be used to list decode against 2-deletion with the same list size. Alternatively, we note that without the sketch $s^{(3)}(\boldsymbol{\alpha})$, the decoding algorithm discussed in Lemma III.1 yields a list of size at most 2. Hence, if the underlying binary sketch function $\bar{s}$ can be used to correct up to 2 deletions, this gives a q-ary sketch that can list decode against 2-deletion with list size of at most 2. On the other hand, if the underlying binary sketch $\bar{s}$ can be used to list-decode against 2-deletion errors with list size $\mathcal{L}$, our construction produces a q-ary sketch that can list decode against 2-deletion with list size of at most $2\mathcal{L}$. We summarize this discussion in the following Corollary, which is stated without proof due to the similarity of the proof to that of Lemma III.1.

Corollary IV.4. *Let $\bar{s}_U : \Sigma_2^n \to \Sigma_2^{\ell(n)}$ be a sketch function that can be used to decode up to 2-deletion errors. Furthermore, let $\bar{s}_L : \Sigma_2^n \to \Sigma_2^{\ell'(n)}$ be a binary sketch function that can be used to list decode up to 2-deletions with list size $\mathcal{L}$ for some positive integer $\mathcal{L}$. We assume that the complexity of computing $\bar{s}_U, \bar{s}_L$, the error correction using $\bar{s}_U$ and the list decoding using $\bar{s}_L$ are $T_{\bar{s}_U,E}(n), T_{\bar{s}_L,E}(n), T_{\bar{s}_U,D}(n)$ and $T_{\bar{s}_L,D}(n)$ respectively. Then for any $q > 2$, there exists:*

1) *A q-ary sketch function s with output length $\ell(n) + 2\log q$ bits that can be used to list-decode 2 deletions with list size of at most $2\mathcal{L}$. Furthermore, the complexities of the sketch computation and the list decoding algorithm*

are $T_{\bar{s}_U,E}(n)+O(n)$ *and* $T_{\bar{s}_U,D}(n)+O(\log^2 q)$ *respectively.*

2) *A* q*-ary sketch function* s *with output length* $\ell'(n)+3\log q+\log n$ *bits that can be used to list-decode* 2 *deletions with list size of at most* $2\mathcal{L}$*. Furthermore, the complexities of the sketch computation and the list decoding algorithm are* $T_{\bar{s}_L,E}(n)+O(n)$ *and* $T_{\bar{s}_L,D}(n)+O(\log^2 q+n)$ *respectively.*
3) *A* q*-ary sketch function* s *with output length* $\ell'(n)+2\log q$ *bits that can be used to list-decode* 2 *deletions with list size of at most* $4\mathcal{L}$*. Furthermore, the complexities of the sketch computation and the list decoding algorithm are* $T_{\bar{s}_L,E}(n)+O(n)$ *and* $T_{\bar{s}_L,D}(n)+O(\log^2 q)$ *respectively.*

Here we provide instantiations based on existing constructions of binary deletion sketches in [9], [12]. We note that the complexities of computing the list-decodable binary sketch proposed in [12] and the list-decoding algorithm are both $O(n)$.

Corollary IV.5.
- *Let* s_U *be the binary sketch function proposed in [9] with output length* $7\log n+6$ *bits. Then the* q*-ary sketch function* $s^{(I)}$ *generated following the construction discussed in the first point of Corollary IV.4 can be used to efficiently list decode against any* 2 *deletions with list size of at most* 2 *and sketch length* $7\log n+2\log q+6$*. Furthermore, the complexities of the sketch computation and the list decoding algorithm are* $O(n)$ *and* $O(\log^2 q+n)$ *respectively.*
- *Let* s_L *be a binary sketch function proposed in [12] that can be used to efficiently list decode any two deletions with list of size at most* 2 *and sketch length* $3\log n+O(\log\log n)$*. Then the sketch* $s^{(II)}$ *obtained by following the construction discussed in the second point of Corollary IV.4 can be used to efficiently list decode any* 2 *deletions with list size of at most* 2 *and sketch length* $4\log n+3\log q+O(\log\log n)$*. The complexities of the sketch computation and the list decoding algorithm are* $O(n)$ *and* $O(\log^2 q+n)$ *respectively.*

  Furthermore, the sketch $s^{(III)}$ *obtained by following the construction discussed in the last point of Corollary IV.4 can be used to efficiently list decode any* 2 *deletions with list size of at most* 4 *and output length* $3\log n+2\log q+O(\log\log n)$*. The complexities of the sketch computation and the list decoding algorithm are* $O(n)$ *and* $O(\log^2 q+n)$ *respectively.*

Remark IV.6. The work in [15] also proposed a q-ary sketch function that can be used to list decode any 2 deletions with list size of at most 2 and sketch length $6\log n+O_q(\log\log n)$ bits. Recall that our construction provides a q-ary sketch that can be used to list decode against 2 deletions with list size of at most 2 with sketch length $4\log n+3\log q+O(\log\log n)$ bits. It is then easy to see that as long as $q^2<n^3$, our construction provides a smaller sketch.

The sketches constructed in Corollary IV.5 yields a q-ary deletion code that can be efficiently encoded and list decoded against any 2-deletions. Such result is summarized in the following theorem.

Theorem IV.7. *There is an explicit* q*-ary deletion correcting code* $\mathcal{C}\subseteq\Sigma_q^N$ *of size* q^n *with* $N\leq n+4\log n+3\log q+O(\log\log n)$ *with redundancy* $4\log n+3\log q+O(\log\log n)$ *with encoding complexity* $O(n)$ *which is list decodable against any* 2*-deletions with list size* 2 *and decoding complexity* $O(n+\log^2 q)$.

Furthermore, there is an explicit q*-ary deletion correcting code* $\mathcal{C}\subseteq\Sigma_q^N$ *of size* q^n *with* $N\leq n+3\log n+2\log q+O(\log\log n)$ *with redundancy* $3\log n+2\log q+O(\log\log n)$ *with encoding complexity* $O(n)$ *which is list decodable against any* 2*-deletions with list size* 4 *and decoding complexity* $O(n+\log^2 q)$.

V. Construction of q-ary 2-Deletion Correcting Code from [12]

We first note that the general construction discussed in Lemma III.1 cannot be directly used if we

use the binary 2-deletion correcting sketch function $s^{(0)}$ proposed in [12] as the underlying binary sketch. This is because the proposed construction requires the binary string considered to be d-regular for some constant d. Note that in the construction in [12], this is done by providing a specific efficient encoding Π of messages to regular strings, which is shown to have sufficiently large domain. Note that in our case, we cannot encode $\varphi(\boldsymbol{\alpha})$ using Π to obtain $\mathbf{a}$ since we can no longer guarantee that the two deletions in $\boldsymbol{\alpha}$ correspond to two deletions in $\mathbf{a}$. Hence, a modification to the construction in Lemma III.1 is required where we further require that $\boldsymbol{\alpha}$ itself is d-regular. This yields the following lemma

Lemma V.1. *Let $\bar{s}$ be the sketch function used in [12, Theorem 5.9]. Then for any d-regular $\boldsymbol{\alpha} \in \Sigma_q^n$, given $s^{(1)}(\boldsymbol{\alpha}) \pmod q, s^{(2)}(\boldsymbol{\alpha}) \pmod q, s^{(3)}(\boldsymbol{\alpha}) \pmod{nq}, \tilde{s}(\boldsymbol{\alpha})$ and any subsequence $\boldsymbol{\alpha}' \in \Sigma_q^{n-2}$ of $\boldsymbol{\alpha}$, the original $\boldsymbol{\alpha}$ can be uniquely and efficiently recovered.*

This implies that if an efficient encoding function to regular q-ary string exists, the existence of an efficiently encodable q-ary 2-deletion correcting code can then be guaranteed. In the following, we consider such encoding function. For any positive integer m, define the set $S_m = \{\boldsymbol{\alpha} \in \Sigma_q^m : \boldsymbol{\alpha} \text{ is } d-\text{regular}\}$. We further define $S_m^{(0)} = \{\boldsymbol{\alpha} \in \Sigma_q^m : \nexists i \in \{1, \cdots, m-2\} : \alpha_i > \alpha_{i+1} > \alpha_{i+2}\}, F_m^{(0)} = |S_m^{(0)}|, S_m^{(1)} = \{\boldsymbol{\alpha} \in \Sigma_q^m : \nexists i \in \{1, \cdots, m-2\} : \alpha_i \leqslant \alpha_{i+1} \leqslant \alpha_{i+2}\}$ and $F_m^{(1)} = |S_m^{(1)}|$.

Lemma V.2. *For any $m \geqslant 3$ and $q > 2, F_m^{(1)} < F_m^{(0)} < (0.99q)^m$.*

Proof. Note that the first inequality is clear since $S_m^{(1)} \subsetneq T_m^{(1)} \triangleq \{\boldsymbol{\alpha} \in \Sigma_q^m : \nexists i \in \{1, \cdots, m-2\} : \alpha_i < \alpha_{i+1} < \alpha_{i+2}\}$ and $|T_m^{(1)}| = F_m^{(0)}$.

Now, we note that $F_3^{(0)} = q^3 - \binom{q}{3}$ and it can be verified that $F_3^{(0)} < (0.99q)^3$ for any $q > 2$. Next we consider $F_4^{(0)}$. Note that for a q-ary string of length 4 not to be considered in $S_4^{(0)}$, it must have a strictly decreasing substring of length 3 either in its first 3 or last 3 elements. Hence, there are $2q\binom{q}{3} - \binom{q}{4}$ of such strings and $F_4^{(0)} = q^4 - 2q\binom{q}{3} + \binom{q}{4}$ which can again be verified to be bounded from above by $(0.99q)^4$ for any $q > 2$. Lastly, we consider $F_5^{(0)}$. It is easy to see that if q-ary string of length 5 is not in $F_5^{(0)}$, then it must have a strictly decreasing substring of length 3, which can start at either the first, second or third entry. Noting that the case that a string has two strictly decreasing substrings of length 3 starting from both the first and third entry must also have a strictly decreasing substring of length 3 from the second entry, by the Inclusion and Exclusion principle, we have

$$\begin{aligned} F_5^{(0)} &= q^5 - 3q^2\binom{q}{3} + 2q\binom{q}{4} \\ &= \frac{7}{12}q^5 + q^4 - \frac{13}{12}q^3 - \frac{q^2}{2} \leqslant (0.99q)^5. \end{aligned}$$

Now we claim that for any $m \geqslant 6$, if $F_i^{(0)} < (0.99q)^i$ for $i = m-3, m-2, m-1$, then we must also have $F_m^{(0)} < (0.99q)^m$.

Note that for $\mathbf{v} = (v_1, \cdots, v_m) \in S_m^{(0)}$, it must satisfy one of the following conditions:

- $v_1 \in \{0, 1\}$ and $(v_2, \cdots, v_m) \in S_{m-1}^{(0)}$,
- $v_1 \geqslant 2, v_2 \in \{0, v_1, v_1 + 1, \cdots, q-1\}$ and $(v_3, \cdots, v_m) \in S_{m-2}^{(0)}$ or
- $v_1 \geqslant 2, v_2 \in \{1, \cdots, v_1 - 1\}, v_3 \geqslant v_2$ and $(v_4, \cdots, v_m) \in S_{m-3}^{(0)}$.

Hence,

$$\begin{aligned} F_m^{(0)} &\leqslant 2F_{m-1}^{(0)} + \sum_{j=2}^{q-1}(q-j+1)F_{m-2}^{(0)} \\ &\quad + \sum_{j=2}^{q-1}\sum_{i=1}^{j-1}(q-i)F_{m-3}^{(0)} \\ &= 2F_{m-1}^{(0)} + \frac{(q+1)(q-2)}{2}F_{m-2}^{(0)} \\ &\quad + \frac{1}{3}q(q-1)(q-2)F_{m-3}^{(0)} \\ &< 2(0.99q)^m + \frac{(q+1)(q-2)}{2}(0.99q)^{m-2} \\ &\quad + \frac{1}{3}q(q-1)(q-2)(0.99q)^{m-3} - 1 \\ &= 0.99^{m-3}q^{m-2}\left(q^2\left(\frac{0.99}{2} + \frac{1}{3}\right)\right. \\ &\quad + q\left(2(0.99)^2 - \frac{0.99}{2} - 1\right) \\ &\quad \left. + \left(\frac{2}{3} - 0.99\right)\right) \end{aligned}$$

which can be easily verified to be at most $(0.99)^m q^m$, as claimed. □

Let U_m be the set of all q-ary strings $\mathbf{v} = (v_1, \cdots, v_m)$ of length m such that it contains $1 \leqslant i, j \leqslant m-2$ where $v_i > v_{i+1} > v_{i+2}$ and $v_j \leqslant v_{j+1} \leqslant v_{j+2}$ and G_m be its size. Hence we have $G_m \geqslant q^m - F_m^{(0)} - F_m^{(1)} > q^m(1 - 2(0.99)^m)$. Then we have the following result.

Lemma V.3. *For $d \geq 134$ and a sufficiently large n, there exists a positive integer $M > q^{n-1}$ and a one-to-one map $\Pi_q : \Sigma_q^{n-1} \to S_n$ which can be efficiently computed.*

Proof. Set $m = \left\lfloor \frac{d}{2} \log n \right\rfloor - 1$ and $\Delta = \left\lfloor \frac{n}{m} \right\rfloor$. It is then easy to see that concatenating Δ q-ary strings taken from U_m and padding it with an arbitrary $(n - m\Delta)$ q-ary string produces an element of S_n, which is a d-regular q-ary string of length n. Hence the number of d-regular q-ary strings of length n that can be constructed in this manner, which we denote by M is

$$\begin{aligned} M = \; & G_m^{\Delta} q^{n-m\Delta} > \left(q^m(1 - 2(0.99)^m)\right)^{\Delta} q^{n-m\Delta} \\ = \; & q^n \left(1 - 2(0.99)^m\right)^{\Delta}. \end{aligned}$$

Note that $2(0.99)^m \leq \left(\frac{1}{2}\right)^{\frac{3d}{400} \log n - \frac{203}{200}}$. Hence for a sufficiently large $n, 2(0.99)^m \leq \left(\frac{1}{2}\right)^{c \log n}$ where $c > 1$ if $d \geq 134$. So we have $2(0.99)^m = O\left(\frac{1}{n^c}\right)$ for some $c > 1$. Furthermore, since $\Delta = O\left(\frac{n}{\log n}\right)$, we have $2\Delta(0.99)^m = O\left(\frac{1}{n^{c-1} \log n}\right)$. Hence for sufficiently large $n, 2\Delta(0.99)^m < 1$ and we have

$$\begin{aligned} (1 - 2(0.99)^m)^{\Delta} &= 1 + \sum_{j=1}^{\Delta} \binom{\Delta}{j} (-2(0.99)^m)^j \\ &> 1 - \sum_{j=1}^{\Delta} \binom{\Delta}{j} (2(0.99)^m)^j \\ &> 1 - \sum_{j=1}^{\Delta} \frac{(2\Delta(0.99)^m)^j}{j!} \\ &> 1 - \sum_{j=1}^{\Delta} \frac{2\Delta(0.99)^m}{j!} \\ &= 1 - 2\Delta(0.99)^m \sum_{j=1}^{\Delta} \frac{1}{j!} \\ &> 1 - 2\Delta(0.99)^m (e-1). \end{aligned}$$

We again note that $2(e-1)\Delta(0.99)^m = O\left(\frac{1}{n^{c-1} \log n}\right)$. Hence for a sufficiently large n, we have $1 - 2(e-1)\Delta(0.99)^m \geqslant \frac{1}{q}$. This shows that for a sufficiently large $n, M > q^{n-1}$. Hence Π_q can be designed and efficiently computed in the same way as the one proposed in [12, Lemma 5.12] with computation complexity polynomial in n. □

We conclude by summarizing our result in the following theorem.

Theorem V.4. *Let $\mathcal{C}_0 \subseteq \Sigma_2^n$ be the binary two-deletion correcting code constructed in [12, Theorem 5.9]. Then for any positive integer $q > 2$, there exists an explicit and efficiently encodable 2 deletion correcting code $C_1 \subseteq \Sigma_q^n$ with redundancy $5 \log n + 10 \log \log n + 3 \log q + O(1)$.*

Furthermore, utilizing our list-decodable code construction described in Corollary IV.4 with $\mathcal{C}_0$ as the underlying binary 2-deletion correcting code, there exists an explicit and efficiently encodable q-ary deletion code $C_2 \subseteq \Sigma_q^n$ with redundancy $4 \log n + 10 \log \log n + 2 \log q + O(1)$ that can be list decoded against 2 deletions with list size of at most 2.

In both cases, the encoding complexity is $\mathrm{poly}(n)$. Furthermore, if the complexity of computing Π_q^{-1} is $T_{\Pi_q, D}(n)$, then the decoding complexity is $2T_{\Pi_q, D}(n) + O(\log^2 q + n)$.

Remark V.5. Note that the q-ary 2-deletion correcting code C_1 defined in Theorem V.4 outperforms any of the existing construction of 2-deletion codes for any q. This includes the code constructed in [16].

Similarly, the q-ary 2-deletion code C_2 defined in Theorem V.4 with list size at most 2 outperforms our best construction from Corollary IV.5.

VI. Acknowledgement

This research is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] V. I. Levenshtein. Bounds for Deletion/Insertion Correcting Codes. In *Proceedings IEEE International Symposium on Information Theory*, pp. 370. doi: 10.1109/ISIT.2002.1023642, 2002.

[2] S. Yekhanin and I. Dumer, *Long Nonbinary Codes Exceeding the Gilbert-Varshamov Bound for any Fixed Distance,* IEEE Trans. on Information Theory, vol.10(2004), 2357–2362.

[3] R. R. Varshamov and G. M. Tenengolts. Codes which Correct Single Asymmetric Errors. In *Automatika i Telemkhanika,* vol. 161, no. 3, pp. 288 – 292. 1965 (in Russian).

[4] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet Physics-Doklady,* vol. 10, no. 8, pp. 707 – 710. 1966. Translated from *Doklady Akademii Nauk SSSR,* vol. 163, no. 4, pp. 845 – 848. 1965 (in Russian).

[5] G. Tenengolts. Nonbinary Codes, Correcting Single Deletion or Insertion. In *IEEE Transactions on Information Theory,* vol. 30, no. 5, pp. 766 – 769. 1984.

[6] A. Helberg and H. Ferreira. On Multiple Insertion/Deletion Correcting Codes. In *IEEE Transactions on Information Theory,* vol. 48, no. 1, pp. 305 – 308. 2002.

[7] T. Le and H. Nguyen. New Multiple Insertion/Deletion Correcting Codes for Non-binary Alphabets. In *IEEE Transactions on Information Theory,* vol. 62, no. 5, pp. 2682 – 2693. 2016.

[8] J. Brakensiek, V. Guruswami and S. Zbarsky. Efficient Low-Redundancy Codes for Correcting Multiple Deletions. In *IEEE Transactions on Information Theory,* vol. 64, no. 5, pp. 3403 – 3410. 2018.

[9] J. Sima, N. Raviv and J. Bruck. Two Deletion Correcting Codes from Indicator Vectors. In *IEEE Transactions on Information Theory*, vol. 66 no. 4 pp. 2375–2391, doi: 10.1109/ISIT.2018.8437868, 2019.

[10] R. Gabrys and F. Sala. Codes Correcting Two Deletions. In *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 965–974, doi: 10.1109/TIT.2018.2876281, 2019.

[11] J. Sima and J. Bruck, Optimal k-Deletion Correcting Codes. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 847–851, doi: 10.1109/ISIT.2019.8849750, 2019.

[12] V. Guruswami and J. Håstad. Explicit Two-Deletion Codes with Redundancy Matching the Existential Bound. In *IEEE Transactions on Information Theory*, vol. 67, no. 10, pp. 6384–6394. doi: 10.1109/TIT.2021.3069446, 2021.

[13] I. Smagloy, L. Welter, A. Wachter-Zeh and E. Yaakobi, Single-Deletion Single-Substitution Correcting Codes, In *IEEE International Symposium on Information Theory (ISIT)*, pp. 775–780, doi: 10.1109/ISIT44484.2020.9174213, 2020.

[14] W. Song, N. Polyanskii, K. Cai and X. He. Systematic Codes Correcting Multiple-Deletion and Multiple-Substitution Errors, in *IEEE Transactions on Information Theory*, vol. 68, no. 10, pp. 6402–6416. doi: 10.1109/TIT.2022.3177169, 2022.

[15] Z. Zhou. 2-Deletion Codes: Beyond Binary. Master Thesis, Carnegie Mellon University, 2020. [Online] Available at: http://reports-archive.adm.cs.cmu.edu/anon/2020/CMU-CS-20-103.pdf

[16] W. Song and K. Cai. Non-binary Two-Deletion Correcting Codes and Burst-Deletion Correcting Codes. *arXiv:2210.14006v1*, 2022. [Online] Available at: http://arxiv.org/abs/2210.14006v1

[17] A. Tonelli, Bemerkung Über die Auflösung Quadratischer Congruenzen. In *Göttinger Nachrichten,* pp. 344–346, 1891.

[18] D. Shanks. Five Number-Theoretic Algorithms. In *Proceedings of the Second Manitoba Conference on Numerical Mathematics, Congressus Numerantium, Utilitas Mathematica,* no. 7, pp 51–70, 1973

Shu Liu (Senior Member, IEEE) received the Ph.D. degree from Nanyang Technological University, Singapore, in 2018. She is currently an Associate Professor with the National Key Laboratory on Wireless Communications, University of Electronic Science and Technology of China, China. Her research interests include coding theory and its applications.

Ivan Tjuawinata (Member, IEEE) received the Ph.D. degree from Nanyang Technological University, Singapore, in 2017. He is currently a Research Fellow with the Strategic Centre for Research in Privacy-Preserving Technologies and Systems (SCRiPTS), Nanyang Technological University. His research interests include secure multiparty computation, cryptanalysis, and coding theory.

Chaoping Xing (Senior Member, IEEE) received the Ph.D. degree from the University of Science and Technology of China, China, in 1990. From 1990 to 1993, he was a Lecturer and an Associate Professor with the University of Science and Technology of China. From 1993 to 1995, he was with the University of Essen, Germany, as an Alexander von Humboldt Fellow. After this, he spent most time with the Institute of Information Processing, Austrian Academy of Sciences, until 1998. From March 1998 to November 2007, he was with the National University of Singapore, Singapore, as an Assistant Professor/an Associate Professor/a Full Professor. From December 2007 to October 2019, he was with Nanyang Technological University, Singapore, as a Full Professor. Since November 2019, he has been with Shanghai Jiaotong University as a Chair Professor. He has been working on the areas of coding theory, cryptography, algebraic curves over finite fields, and quasi-Monte Carlo methods.