A New Version of q-ary Varshamov-Tenengolts Codes with more Efficient Encoders: The Differential VT Codes and The Differential Shifted VT Codes

Tuan Thanh Nguyen, *Member, IEEE,* Kui Cai, *Senior Member, IEEE,* and Paul H. Siegel, *Life Fellow, IEEE*

***Abstract*—The problem of correcting deletions and insertions has recently received significantly increased attention due to the DNA-based data storage technology, which suffers from deletions and insertions with extremely high probability. In this work, we study the problem of constructing non-binary burst-deletion/insertion correcting codes. Particularly, for the quaternary alphabet, our designed codes are suited for correcting a burst of deletions/insertions in DNA storage.**

Non-binary codes correcting a single deletion or insertion were introduced by Tenengolts [1984], and the results were extended to correct a fixed-length burst of deletions or insertions by Schoeny *et al.* [2017]. Recently, Wang *et al.* [2021] proposed constructions of non-binary codes of length n, correcting a burst of length at most two for q-ary alphabets with redundancy $\log n+O(\log q \log\log n)$ bits, for arbitrary even q. The common idea in those constructions is to convert non-binary sequences into binary sequences, and the error decoding algorithms for the q-ary sequences are mainly based on the success of recovering the corresponding binary sequences, respectively.

In this work, we look at a natural solution that the error detection and correction algorithms are performed directly over q-ary sequences, and for certain cases, our codes provide a more efficient encoder with lower redundancy than the best-known encoder in the literature. Particularly,

- **(Single-error correction codes) We first present a new version of non-binary VT codes that are capable of correcting a single deletion or single insertion, providing an alternative simpler and more efficient encoder of the construction by Tenengolts [1984]. Our construction is based on the *differential vector*, and the codes are referred to as the *differential VT codes*. In addition, we provide linear-time algorithms that encode user messages into these codes of length n over the q-ary alphabet for $q \geqslant 2$ with at most $\lceil \log_q n \rceil + 1$ redundant symbols, while the optimal redundancy required is at least $\log_q n + \log_q(q-1)$ symbols. Our designed encoder reduces the redundancy of the best-known encoder of Tenengolts [1984] by at least 2 redundant symbols or equivalently $2\log_2 q$ bits.**
- **(Burst-error correction codes) We use the idea of the *binary shifted VT codes* to define the *q-ary differential shifted VT codes*, and propose non-binary codes correcting a burst of up to two deletions (or two insertions) with redundancy $\log n + 3\log\log n + O(\log q)$ bits, which improves a recent result of Wang *et al.* [2021] with redundancy $\log n + O(\log q \log\log n)$ bits for all $q \geqslant 8$. We then extend the construction to design non-binary codes correcting a burst of either exactly or at most t deletions (or insertions) for arbitrary $t \geqslant 2$.**

This work was presented in part at the IEEE 2023 IEEE International Conference on Communications: SAC Cloud Computing, Networking and Storage Track (IEEE ICC) [1]. The work of Tuan Thanh Nguyen and Kui Cai was supported by the SUTD Kickstarter Initiative (SKI) Grant 2021_04_05 and the Singapore Ministry of Education Academic Research Fund Tier 2 T2EP50221-0036. The work of Paul Siegel is supported in part by NSF Grant CCF-2212437.

Tuan Thanh Nguyen and Kui Cai are with the Science, Mathematics, and Technology Cluster, Singapore University of Technology and Design, Singapore 487372 (email: {tuanthanh_nguyen, cai_kui}@sutd.edu.sg).

Paul H. Siegel is with the University of California, San Diego, La Jolla, CA 92093, USA (email: psiegel@ucsd.edu).

I. Introduction

Codes correcting deletions and insertions are important for many data storage systems such as the bit-patterned media magnetic recording systems [2] and racetrack memory devices [3]. Insertions and deletions may also occur due to the synchronization errors in communication systems [4] and mobile data [5]. Furthermore, the problem of correcting such errors has recently received significantly increased attention due to the DNA-based data storage technology, which suffers from deletions and insertions with extremely high probability [6]–[10]. Designing codes for correcting deletions and/or insertions is well-known to be a challenging problem, even in the most fundamental settings with only a single error. One of the challenges that make deletions or insertions more destructive than substitutions is that only a small number of errors can cause the original data sequences and the received sequences to be vastly different under the Hamming metric.

In this work, we focus on the design of non-binary codes that are capable of correcting a burst of deletions (or insertions), where a burst refers to a block of errors that occur in consecutive symbols. This has been pointed out as a typical type of error that arises in DNA-based data storage technology that uses nanopore sequencing technologies [11], [12]. In addition, in wireless communications, burst errors also occur with high frequency due to multi-path fading [13], [14]. In this work, not only are we interested in constructing large error-correction codes, we desire efficient encoders and decoders that map arbitrary user data into these codes and vice versa. In general, code design takes into account the lowest redundancy required to correct such errors with fast encoding and decoding procedures. In this work, we define $\mathcal{B}_t(\boldsymbol{x})$ to be the set of sequences that can be obtained from $\boldsymbol{x}$ via a burst of either t deletions or t insertions. Similarly, $\mathcal{B}_{\leqslant t}(\boldsymbol{x})$ is the set of sequences that can

be obtained from $\boldsymbol{x}$ via a burst of at most t deletions or at most t insertions.

Over the q-ary alphabet, $q \geqslant 2$, consider a channel model with codewords of length n and a given error ball function $\mathcal{B}$. Suppose that the optimal redundancy required to correct such errors is $\mathrm{r}_{n,q,\mathcal{B}}$; then two crucial coding theory problems are:

P1: Code Construction. Can one construct the largest-size code $\mathcal{C}$ of length n, whose redundancy, denoted by $\mathrm{r}_{\mathcal{C}}$, should satisfy that $\lim_{n\to\infty}(\mathrm{r}_{\mathcal{C}} - \mathrm{r}_{n,q,\mathcal{B}}) = 0$?

P2: Encoder/Decoder Design. Can one design an efficient encoder ENC (and a corresponding decoder DEC) that encodes arbitrary user messages into codewords of length n in $\mathcal{C}$ such that the redundancy of the encoder ENC, denoted by r_{ENC}, should satisfy that $\lim_{n\to\infty}(\mathrm{r}_{\text{ENC}} - \mathrm{r}_{\mathcal{C}}) = 0$?

In the literature, the problems of constructing codes (problem P1) correcting a burst of exactly t deletions (or exactly t insertions), also known as *fixed-length burst*, and a burst of at most t deletions (or at most t insertions), also known as *variable-length burst* have both been studied, with the latter being the more complex problem [15]–[25]. On the other hand, designing efficient encoders (problem P2) is crucial for practical applications, however, in many settings, it remains an open challenge, even in the most fundamental settings with only a single error.

Non-binary single-error correction codes. The first challenge comes from extending the coding solutions in binary codes to non-binary codes. Particularly, while the problems of giving nearly-optimal explicit constructions of single-deletion (or single-insertion) correction codes (P1) and designing nearly-optimal encoders for such codes (P2) over the binary alphabet have been settled for more than 50 years, the approach fails to be extended to the case of q-ary alphabet for any fixed $q > 2$. In particular, to correct a single deletion or single insertion, we have the celebrated class of Varshamov-Tenengolts (VT) codes. In 1965, Varshamov and Tenengolts introduced the binary VT codes to correct asymmetric errors [19], and Levenshtein subsequently showed that such codes can be used for correcting a deletion or insertion with a simple linear-time decoding algorithm [20]. For codewords of length n, the binary VT codes incur $\log(n+1)$ redundant bits [1] , while the optimal redundancy, provided in [20], is at least $\log n$ bits. Curiously, even though the binary VT codes and efficient decoding algorithm were known since 1965, a linear-time encoder for such codes was only proposed by Abdel-Ghaffar and Ferriera in 1998 [21], which used $\lceil \log(n+1) \rceil$ redundant bits. We observe that, over the binary alphabet, (P1) and (P2) are solved asymptotically optimal:

$$\mathrm{r}_{n,2,\mathcal{B}_1} \geqslant \log n, \ \mathrm{r}_{\mathcal{C}} = \log(n+1), \text{ and } \mathrm{r}_{\text{ENC}} = \lceil \log(n+1) \rceil.$$

Here, we have

$$\lim_{n\to\infty}(\mathrm{r}_{\mathcal{C}} - \mathrm{r}_{n,2,\mathcal{B}_1}) = \lim_{n\to\infty}(\mathrm{r}_{\text{ENC}} - \mathrm{r}_{\mathcal{C}}) = 0.$$

[1]In this work, for simplicity, we use the notation "log" without the base to refer to the logarithm of base two.

For the non-binary alphabet, in 1984, a non-binary version of the VT codes was proposed by Tenengolts [22], and the constructed codes can correct a single deleted or inserted symbol in the q-ary alphabet with a linear-time decoder for any $q > 2$. The construction of Tenengolts retains the attractive properties of the binary VT codes, such as the simple decoding algorithm. For codewords of length n, such codes incur at most $\log_q n + 1$ redundant symbols. In the same paper, Tenengolts also provided an upper bound for the cardinality of any q-ary codes of length n correcting a deletion or insertion, which is at most $q^n/((q-1)n)$, and hence, the minimum redundancy required is at least $\log_q n + \log_q(q-1)$ symbols. Unlike the binary case, designing an efficient encoder that encodes arbitrary user messages into Tenengolts' code is a challenging task (refer to Section III-A for detailed discussion). To overcome the challenge, several attempts have been made in three variations:

- *Targeting a specific value of q.* When $q = 4$, Chee *et al.* [26] presented a linear-time quaternary encoder that corrects a single deletion or insertion with $\lceil \log_4 n \rceil + 1$ redundant symbols. The redundancy is asymptotically optimal. Unfortunately, the approach fails to be extended to the case of q-ary alphabet for arbitrary $q > 2$.
- *Using more redundancy.* Abroshan *et al.* [27] presented a systematic encoder that maps user messages into a single q-ary VT code as constructed in [22] with complexity that is linear in the code length. Unfortunately, the redundancy of this encoder is more than $\log_q n + \log n$ symbols (see Section II).
- *Relaxing the condition for output codewords*. In [22], Tenengolts provided a systematic encoder that requires at least $\lceil \log_q n \rceil + 3$ symbols, which is the best-known encoder for codes that correct a single deletion or insertion. In terms of redundancy, a natural question is: can one construct a linear-time encoder with at most r redundant symbols, where $\log_q n + \log_q(q-1) \leqslant r < \lceil \log_q n \rceil + 3$? In addition, The drawback of the encoder in [22] is that the codewords obtained from this encoder are not contained in a single q-ary VT code. Note that to correct a single deletion or insertion, it is not necessary that all the codewords must belong to the same coset of q-ary VT codes. Nevertheless, when the words share the same parameters, Abroshan *et al.* [27] demonstrated that these codes can be adapted to correct multiple insertion/deletion errors, in the context of *segmented edits* [28]–[30].

Our contribution for single-error correction codes. Motivated by the code design problem above, we present a new version of non-binary VT codes that give asymptotically optimal solutions for (P1) and (P2), as follows:

$$\mathrm{r}_{n,q,\mathcal{B}_1} \geqslant \log_q n + \log_q(q-1), \text{ while}$$
$$\mathrm{r}_{\mathcal{C}} = \log_q n + 1, \text{and } \mathrm{r}_{\text{ENC}} = \lceil \log_q n \rceil + 1.$$

We observe that (P1) and (P2) are now solved asymptotically optimal since

$$\lim_{n\to\infty}(\mathrm{r}_{\mathcal{C}} - \mathrm{r}_{n,q,\mathcal{B}_1}) = \lim_{n\to\infty}(\mathrm{r}_{\text{ENC}} - \mathrm{r}_{\mathcal{C}}) = 0.$$

Our construction is based on the *differential vector*, and the codes are referred to as the *differential VT codes*. Our constructed codes have the same cardinality and redundancy, as compared to the best known q-ary single deletion/insertion codes constructed by Tenengolts [22]. On the other hand, our proposed code construction method supports more efficient encoding and decoding procedures (in other words, it enables an easier method to solve (P2)). Consequently, our best encoder uses at most $\lceil \log_q n \rceil + 1$ redundant symbols, and hence, it reduces the redundancy of the best known encoder of Tenengolts [22] by at least 2 redundant symbols, or equivalently $2 \log q$ redundant bits.

Non-binary burst-error correction codes. The earliest work on the subject, proposed by Levenshtein in 1967 [15], provided an efficient construction of binary codes capable of correcting a burst at most two deletions (or two insertions) that had redundancy $\log n + 1$ for codewords of length n. Binary codes correcting a burst of deletions (or insertions) were later proposed in [16], [18]. Particularly, for an arbitrary constant $t > 1$, Schoeny *et al.* [16] proposed binary codes correcting a burst of length exactly t, while the work of Lenz and Polyanskii in [18] can correct a burst of variable length up to t. Note that, there is a significant difference between codes that can correct a burst of length at most t and a burst of length exactly t, as a code of the earlier type can correct errors of the latter, but the converse is not true in general. Over the general q-ary alphabet, recently, Wang *et al.* [24] proposed constructions of codes of length n, correcting a burst of length at most two with redundancy $\log n + O(\log q \log \log n)$ bits, for arbitrary even q. The results were later extended to construct non-binary codes correcting a burst of up to t deletions (or insertions) in [25]. However, designing efficient encoders (problem P2) for such constructed codes remains an open challenge, even in the case of $t = 2$. Particularly, to correct a burst of at most 2 errors, the authors [25] provided a systematic construction of encoder, however, the redundancy is roughly $\log q \log n + O(\log q)$, which is much larger than the constructed codes whose redundancy was only $\log n + O(\log q \log \log n)$ bits.

Our contribution for burst-error correction codes. We use the idea of the *binary shifted VT codes* to define the *q-ary differential shifted VT codes*, which is crucial to the construction of q-ary codes correcting a burst of errors. Given $t > 0$, we propose non-binary codes correcting a burst of either exactly or at most t deletions/insertions. Particularly, for $t = 2$ and a given q-ary alphabet, we construct non-binary codes of length n that can correct a burst of at most two deletions or two insertions with redundancy $\log n + 3 \log \log n + O(\log q)$ bits, which improves a recent result of Wang *et al.* [2021] with redundancy $\log n + O(\log q \log \log n)$ bits for all $q \geqslant 8$. In addition, we present a linear-time encoder that encodes arbitrary user messages into non-binary codes correcting a burst of at most two deletions with redundancy $\log n + 3 \log \log n + O(\log q)$ bits, which improves the redundancy of the encoder in [25]. We then extend the coding method to correct a burst of exactly t errors (or at most t errors) for an arbitrary value of t.

The remainder of this paper is organized as follows. We first go through notations and some preliminary results in Section II. In Section III-A, we focus on the single error correction code, i.e. $t = 1$, and present a new version of non-binary VT codes, which are referred to as the *differential VT codes*. In addition, in Section III-B, we present a linear-time encoder that encodes user messages into the codes, and for codewords of length n over the q-ary alphabet, our designed encoder uses at most $\lceil \log_q n \rceil + 1$ redundant symbols. The efficiency of our proposed encoders, compared to previous works on single error correction codes, is illustrated in Table II. In Section IV, we introduce the *differential shifted VT codes* and propose non-binary codes correcting a burst of exactly t errors with redundancy $\log n + (t-1) \log \log n + O(t \log q)$ bits, and design linear-time encoders for such codes. We then extend the coding method to correct at most t deletions in Section V. Finally, Section VI concludes the paper. A summary of our contributions is illustrated in Table I.

II. Preliminary

Let Σ_q denote an *alphabet* of size q, where $\Sigma_q = \{0, 1, 2, \ldots, q-1\}$. For any positive integer $m < n$, we let $[m, n]$ denote the set $\{m, m+1, \ldots, n\}$ and $[n] = [1, n]$.

Given two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$, we let $\boldsymbol{xy}$ denote the *concatenation* of the two sequences. In the special case where $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^n$, we use $\boldsymbol{x} || \boldsymbol{y}$ to denote their *interleaved sequence* $x_1 y_1 x_2 y_2 \ldots x_n y_n$. For a subset $I = \{i_1, i_2, \ldots, i_j\}$ of coordinates, we use $\boldsymbol{x}|_I$ to denote the vector $x_{i_1} x_{i_2} \ldots x_{i_j}$. A sequence $\boldsymbol{y}$ is said to be a *subsequence* of $\boldsymbol{x}$, if there exists a subset of coordinates I such that $\boldsymbol{y} = \boldsymbol{x}|_I$. We now introduce the definition of a burst of deletions or insertions.

Definition 1. Given $\boldsymbol{x} = (x_1, x_2 \ldots, x_n) \in \Sigma_q^n$. We say that $\boldsymbol{x}$ suffers a burst of t deletions if exactly t consecutive symbols have been deleted from $\boldsymbol{x}$, resulting a subsequence $\boldsymbol{x}' = (x_1, x_2, \ldots, x_i, x_{i+t+1}, x_{i+t+2}, \ldots, x_n) \in \Sigma_q^{n-t}$ for some $i \in [n-t]$. On the other hand, we say that $\boldsymbol{x}$ suffers a burst of t insertions if exactly t consecutive insertions have occurred from $\boldsymbol{x}$, resulting a subsequence $\boldsymbol{x}'' = (x_1, x_2, \ldots, x_j, y_1, y_2, \ldots, y_t, x_{i+1}, x_{i+2}, \ldots, x_n) \in \Sigma_q^{n+t}$ for some $i \in [n]$. Similarly, we say $\boldsymbol{x}$ suffers a burst of up to t deletions if s_1 consecutive symbols have been deleted for some $s_1 \leqslant t$, or $\boldsymbol{x}$ suffers a burst of up to t insertions if s_2 consecutive insertions have occurred for some $s_2 \leqslant t$.

In this work, we define $\mathcal{B}_t(\boldsymbol{x})$ to be the set of sequences that can be obtained from $\boldsymbol{x}$ via a burst of either t deletions or t insertions. Similarly, $\mathcal{B}_{\leqslant t}(\boldsymbol{x})$ is the set of sequences that can be obtained from $\boldsymbol{x}$ via a burst of at most t errors.

Definition 2. Let $\mathcal{C} \subseteq \Sigma_q^n$. We say that $\mathcal{C}$ corrects a burst of t deletions or t insertions if and only if $\mathcal{B}_t(\boldsymbol{x}) \cap \mathcal{B}_t(\boldsymbol{y}) = \varnothing$ for all distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$. Similarly, we say that $\mathcal{C}$ can correct a burst of up to t deletions or up to t insertions if and only if $\mathcal{B}_{\leqslant t}(\boldsymbol{x}) \cap \mathcal{B}_{\leqslant t}(\boldsymbol{y}) = \varnothing$ for all distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$.

	Size of burst	(P1) Redundancy of the constructed code $\mathcal{C}$	(P2) Redundancy of the encoder for $\mathcal{C}$
Tenengolts [22]	$=1$	$\log_q n+1$ (symbols)	$\lceil \log_q n \rceil + 3$ (symbols)
This work	$=1$	$\log_q n+1$ (symbols)	$\lceil \log_q n \rceil + 1$ (symbols)
Wang *et al.* [24]	$\leqslant 2$	$\log n + O(\log q \log\log n)$ (bits)	$\log q \log n + O(\log q)$ (bits)
This work	$\leqslant 2$	$\log n + 3\log\log n + O(\log q)$ (bits)	$\log n + 3\log\log n + O(\log q)$ (bits)
Schoeny *et al.* [12]	$=t$	$\log n + (t-1)\log\log n + O(t\log q)$ (bits)	NA
This work	$=t$	$\log n + (t-1)\log\log n + O(t\log q)$ (bits)	$\log n + (t-1)\log\log n + O(t\log q)$ (bits)
Wang *et al.* [25]	$\leqslant t$	$\log n + O(\log q \log\log n)$ (bits)	NA
This work	$\leqslant t$	$\log n + O(t^2\log\log n) + O(t\log q)$ (bits)	NA

TABLE I: Related works for non-binary codes in the literature and the main contributions of this work.

For a code $\mathcal{C} \subseteq \Sigma_q^n$, the redundancy is measured by the value $\mathrm{r}_{\mathcal{C}} = n - \log_q |\mathcal{C}|$ (in symbols) or $n\log q - \log|\mathcal{C}|$ (in bits). In this work, not only are we interested in constructing large error-correction codes (problem P1), we desire an efficient encoder that maps arbitrary user data into these codes (problem P2).

Definition 3. The map $\textsc{Enc} : \Sigma_q^k \to \Sigma_q^n$ is a *t-burst-encoder* if there exists a *decoder* map $\textsc{Dec} : \Sigma_q^{n+t} \cup \Sigma_q^n \cup \Sigma_q^{n-t} \to \Sigma_q^n$ such that the following conditions hold:

- For all $\boldsymbol{x} \in \Sigma_q^k$, we have $\textsc{Dec} \circ \textsc{Enc}(\boldsymbol{x}) = \boldsymbol{x}$,
- If $\boldsymbol{c} = \textsc{Enc}(\boldsymbol{x})$ and $\boldsymbol{c}' \in \mathcal{B}_t(\boldsymbol{c})$, then $\textsc{Dec}(\boldsymbol{c}') = \boldsymbol{x}$.

Hence, we have that the code $\mathcal{C} = \{\boldsymbol{c} : \boldsymbol{c} = \textsc{Enc}(\boldsymbol{x}), \boldsymbol{x} \in \Sigma_q^k\}$ and $|\mathcal{C}| = q^k$. The *message length* is k while the *codeword length* is n. The *redundancy of the encoder* is measured by the value $n-k$ (in symbols) or $(n-k)\log q$ (in bits). A *$\leqslant t$-burst-encoder* can be defined similarly.

Definition 4. For $q \geqslant 2$, the *VT syndrome* of a q-ary sequence $\boldsymbol{x} \in \Sigma_q^n$ is defined to be $\mathrm{Syn}(\boldsymbol{x}) = \sum_{i=1}^n i x_i$.

To correct a single deletion or single insertion, we have the celebrated class of Varshamov-Tenengolts (VT) codes.

Construction 1 (Binary VT codes [19])**.** *Given $n > 0$ and $q = 2$. For $a \in \mathbb{Z}_{n+1}$, let*

$$\mathrm{VT}_a(n) = \Big\{\boldsymbol{x} \in \{0,1\}^n : \mathrm{Syn}(\boldsymbol{x}) = a \pmod{(n+1)}\Big\}.$$

Theorem 1 (Levenshtein, 1965 [20])**.** *For $a \in \mathbb{Z}_{n+1}$, $\mathrm{VT}_a(n)$ can correct a single deletion or a single insertion. There exists $a \in \mathbb{Z}_{n+1}$ such that $\mathrm{VT}_a(n)$ has at least $2^n/(n+1)$ codewords, and the redundancy of the code is at most $\log(n+1)$ bits.*

Over the nonbinary alphabet, in 1984, Tenengolts [22] generalized the binary VT codes to q-ary VT codes for any fixed q-ary alphabet. Crucial to the construction of Tenengolts in [22] was the concept of the *signature vector* defined as follows.

Definition 5. The *signature vector* of a q-ary vector $\boldsymbol{x}$ of length n is a binary vector $\alpha(\boldsymbol{x})$ of length $n-1$, where $\alpha(x)_i = 1$ if $x_{i+1} \geq x_i$, and 0 otherwise, for $i \in [n-1]$.

Construction 2 (q-ary VT codes as proposed in [22])**.** *Given $n, q > 0$, for $a \in \mathbb{Z}_n$ and $b \in \mathbb{Z}_q$, set*

$$\mathrm{T}_{a,b}(n;q) \triangleq \Big\{\boldsymbol{x} \in \mathbb{Z}_q^n : \alpha(\boldsymbol{x}) \in \mathrm{VT}_a(n-1), \text{ and } \sum_{i=1}^n x_i = b \pmod q\Big\}.$$

Theorem 2 (Tenengolts, 1984 [22])**.** *The set $\mathrm{T}_{a,b}(n;q)$ forms a q-ary single deletion/insertion correction code and there exists a and b such that the size of $\mathrm{T}_{a,b}(n;q)$ is at least $q^n/(qn)$. There exists a systematic encoder $\textsc{Enc}_{\mathrm{T}}$ with redundancy $\lceil\log n\rceil + 3\lceil\log q\rceil$ (bits) or $\lceil\log_q n\rceil + 3$ (symbols).*

On the other hand, the codewords obtained from the encoder $\textsc{Enc}_{\mathrm{T}}$ are not contained in a single q-ary VT code $\mathrm{T}_{a,b}(n;q)$. Recently, Abroshan *et al.* [27] presented a systematic encoder that maps binary messages into $\mathrm{T}_{a,b}(n;q)$. Unfortunately, the redundancy of the encoder is as large as $\log n(\log q + 1) + 2(\log q - 1)$ bits, and hence, more than $\log n + \log_q n$ symbols.

III. Correcting a Single Deletion or Insertion: A New Version of q-ary VT Codes

A Natural Idea from Binary VT Codes. Recall the design of the binary VT codes $\mathrm{VT}_a(n)$ from Construction 1 to correct a single deletion or insertion. A natural question is whether there exists a simple VT syndrome over q-ary codewords to correct single deletion or insertion for arbitrary $q > 2$. Observe that, in the construction of Tenengolts [22] (refer to Construction 2), the VT syndrome is enforced over the signature of each codeword, which is a binary sequence. That is a drawback leading to the difficulty of designing an efficient encoder as in the binary case. Consequently, to encode arbitrary messages into $\mathrm{T}_{a,b}(n;q)$ by enforcing the VT syndrome over the binary signature sequences, Abroshan *et al.* [27] required more than $\log_q n + \log n$ redundant symbols. A natural solution should be obtained by enforcing a single VT syndrome over all q-ary sequences, and consequently, the design of a corresponding encoder would be simple as in the binary case. On the other

hand, we observe that imposing VT syndrome directly over every q-ary codeword is not sufficient to correct a deletion or insertion. For example, it is easy to verify that the following two sequences $\boldsymbol{z}_1 = \boldsymbol{x}213\boldsymbol{y}$ and $\boldsymbol{z}_2 = \boldsymbol{x}132\boldsymbol{y}$, where $\boldsymbol{x}, \boldsymbol{y}$ are arbitrary sequences, have the same VT syndrome, however, share a common sequence in the single error ball as $\boldsymbol{z}' = \boldsymbol{x}13\boldsymbol{y}$. The first contribution of our work is to show that imposing the VT syndrome over the *differential vector* of every q-ary codeword allows us to correct a single error.

A. The Differential VT Codes

Definition 6. Given $\boldsymbol{x} \in \Sigma_q^n$. The *differential vector* of $\boldsymbol{x}$, denoted by $\mathrm{Diff}(\boldsymbol{x})$, is a sequence $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x}) \in \Sigma_q^n$ where:

$$\begin{cases} y_i &= x_i - x_{i+1} \pmod q, \text{ for } 1 \leqslant i \leqslant n-1, \\ y_n &= x_n. \end{cases}$$

Clearly, $\mathrm{Diff}(\boldsymbol{x})$ is a one-to-one function. From $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x})$, we can obtain $\boldsymbol{x} = \mathrm{Diff}^{-1}(\boldsymbol{y})$ as follows.

$$\begin{cases} x_n &= y_n, \text{ and} \\ x_i &= \sum_{j=i}^{n} y_j \pmod q, \text{ for } n-1 \geqslant i \geqslant 1. \end{cases}$$

Construction 3 (The q-ary Differential VT codes). *Given $n > 0$. For $q \geqslant 2, a \in \mathbb{Z}_{qn}$, set*

$$\mathrm{Diff\_VT}_a(n;q) \triangleq \left\{\boldsymbol{x} \in \Sigma_q^n : \mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x})) = a \pmod{qn}\right\}.$$

Our main contribution in this section is as follows.

Theorem 3. *The code $\mathrm{Diff\_VT}_a(n;q)$ can correct a single deletion or single insertion in linear time. In other words, there exists a linear-time decoder $\mathrm{DEC}_{\mathrm{error}} : \Sigma_q^{n-1} \cup \Sigma_q^{n+1} \to \Sigma_q^n$ such that if $\boldsymbol{x}'$ is obtained from $\boldsymbol{x} \in \mathrm{Diff\_VT}_a(n;q)$ after a deletion or an insertion, we can recover $\boldsymbol{x} = \mathrm{DEC}_{\mathrm{error}}(\boldsymbol{x}')$. There exists $a \in \mathbb{Z}_{qn}$, such that $\left|\mathrm{Diff\_VT}_a(n;q)\right| \geqslant q^n/(qn)$.*

The following lemmas are crucial to prove Theorem 3.

Lemma 1. *Given $\boldsymbol{x} \in \Sigma_q^n$ and let $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x}) \in \Sigma_q^n$. Suppose that $\boldsymbol{x}'$ is obtained via $\boldsymbol{x}$ by a deletion at symbol x_i for some $1 \leqslant i \leqslant n$. We then have $\boldsymbol{y}' = \mathrm{Diff}(\boldsymbol{x}') \in \Sigma_q^{n-1}$ and:*

(i) If $2 \leqslant i \leqslant n$, then $y_{i-1}y_i$ in $\boldsymbol{y}$ are replaced by one symbol $(y_{i-1} + y_i) \pmod q$.

(ii) If $i = 1$, then y_1 is deleted in $\boldsymbol{y}$. In other words, we have $\boldsymbol{y}' = y_2 y_3 \dots y_n$.

Proof. We have $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x})$, where $y_i = x_i - x_{i+1} \pmod q$ for $1 \leqslant i \leqslant n-1$ and $y_n = x_n$.

If $i = 1$, we have $\boldsymbol{x}' = x_2 x_3 \dots x_n$. Clearly, $\mathrm{Diff}(\boldsymbol{x}') = y_2 y_3 \dots y_n$, or y_1 is deleted in $\boldsymbol{y}$.

If $2 \leqslant i \leqslant n$, a deletion at x_i affects y_{i-1}, y_i in $\mathrm{Diff}(\boldsymbol{x})$, as $y_{i-1} = x_{i-1} - x_i \pmod q$ and $y_i = x_i - x_{i+1} \pmod q$. We observe that the change in $\mathrm{Diff}(\boldsymbol{x}')$ is then

$$\begin{aligned} \mathrm{Diff}(\boldsymbol{x}')_{i-1} &= x_{i-1} - x_{i+1} \pmod q \\ &= (x_{i-1} - x_i) + (x_i - x_{i+1}) \pmod q \\ &= y_{i-1} + y_i \pmod q. \end{aligned}$$

We conclude that $y_{i-1}y_i$ is replaced by $y_{i-1} + y_i \pmod q$. ■

Example 1. Consider $\Sigma_4 = \{0, 1, 2, 3\}$, and $\boldsymbol{x} = 0211301$. We then have $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x}) = 2102331$. Suppose that the symbol 2 is deleted in $\boldsymbol{x}$, resulting $\boldsymbol{x}' = 011301$, and $\mathrm{Diff}(\boldsymbol{x}') = 302331$. In this example, we observe that, x_2 is deleted in $\boldsymbol{x}$, and the resulting $y_1y_2 = 21$ in $\mathrm{Diff}(\boldsymbol{x})$ is replaced by $3 = y_1 + y_2$.

Lemma 2 (Parity check lemma)**.** *Given $n > 0$, $q \geqslant 2$, and $a \in \mathbb{Z}_{qn}$. Consider $\boldsymbol{x} \in \Sigma_q^n$ such that $\mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x})) = a \pmod{qn}$. We then have $\sum_{i=1}^{n} x_i \equiv a \pmod q$.*

Proof. Let $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x})$, where $y_i = x_i - x_{i+1} \pmod q$ for $1 \leqslant i \leqslant n-1$ and $y_n = x_n$. Suppose that $\mathrm{Syn}(\boldsymbol{y}) = a + kqn$ for some positive integer k. We have

$$\begin{aligned} \mathrm{Syn}(\boldsymbol{y}) &= \sum_{i=1}^{n-1} i y_i + n y_n \\ &\equiv \sum_{i=1}^{n} i(x_i - x_{i+1}) + n x_n \pmod q \\ &\equiv \sum_{i=1}^{n} x_i \pmod q. \end{aligned}$$

Since $\mathrm{Syn}(\boldsymbol{y}) = a + kqn$, it implies $\sum_{i=1}^{n} x_i \equiv a \pmod q$. ■

We are now ready to show the correctness of Theorem 3. Note that any code that corrects k deletions if and only if it can correct k insertions, as established by Levenshtein [23]. Also, a code $\mathcal{C}$ can correct a deletion burst of size exactly (or at most) k if and only if it can correct an insertion burst of size exactly (or at most, respectively) k (refer to Theorem 2, Theorem 3 in [16]). Therefore, for simplicity, throughout this paper, we present the decoding algorithm to correct deletion errors only.

Proof of Theorem 3. Observe that the lower bound is verified by using the pigeonhole principle. It remains to show that the code $\mathrm{Diff\_VT}_a(n;q)$ can correct a single deletion.

For a codeword $\boldsymbol{x} \in \mathrm{Diff\_VT}_a(n;q)$, let $\boldsymbol{x}'$ be obtained from $\boldsymbol{x}$ after a deletion of symbol γ at index i, i.e. $x_i = \gamma$. According to Lemma 2, we can obtain the value of the deleted symbol as follows: $\gamma = a - \sum_{j=1}^{n-1} x'_j \pmod q$. It remains to determine the value of i, i.e. the location of the deleted symbol. Let $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x})$ and $\boldsymbol{y}' = \mathrm{Diff}(\boldsymbol{x}')$. We then compute:

$$\Delta = \mathrm{Syn}(\boldsymbol{y}) - \mathrm{Syn}(\boldsymbol{y}') = a - \mathrm{Syn}(\boldsymbol{y}') \pmod{qn}, \text{ and}$$

$$s = \sum_{j=1}^{n-1} y'_j, \text{ i.e. the sum of symbols in } \boldsymbol{y}'.$$

Observe that the code's parameters such as a, q, n are known, and the received sequence $\boldsymbol{x}'$ and its differential vector $\boldsymbol{y}'$ are known, hence, the values of Δ and s can be determined. Let $s_R = \sum_{j=i}^{n} y_j$. We show how $\boldsymbol{y}$ can be recovered from $\boldsymbol{y}'$ and thus $\boldsymbol{x}$ can be recovered based on Δ and s, which are computable at the decoder. We now have the following cases.

Case 1. If $i = 1$, we consider a non-trivial case that $y_1 > 0$. Indeed, if $y_1 = 0$, it implies $x_1 = x_2$, and such a deletion in x_1 is equivalent to a deletion in x_2, which is considered in Case 2. Thus, we obtain $\Delta = y_1 + \sum_{j=1}^{n-1} y'_j = y_1 + s > s$ and $\Delta < q + s$.

Case 2. If $2 \leqslant i \leqslant n$, according to Lemma 1, $y_{i-1}y_i$ is replaced by $y_{i-1} + y_i \pmod q$. In other words, in the sequence $\boldsymbol{y}' = \mathrm{Diff}(\boldsymbol{x}')$, we have $y'_{i-1} = y_{i-1} + y_i \pmod q$.

- (2a) If $y_{i-1} + y_i \leqslant q-1$, then $y'_{i-1} = y_{i-1} + y_i$, and

$$\begin{aligned}\Delta &= \mathrm{Syn}(\boldsymbol{y}) - \mathrm{Syn}(\boldsymbol{y}') \pmod{qn} \\ &= \Big((i-1)y_{i-1} + iy_i - (i-1)y'_{i-1}\Big) + \sum_{j=i}^{n-1} y'_j \\ &= \Big((i-1)y_{i-1} + iy_i - (i-1)y_{i-1} - (i-1)y_i\Big) + \sum_{j=i}^{n-1} y'_j \\ &= y_i + \sum_{j=i}^{n-1} y'_j = s_R \leqslant s.\end{aligned}$$

- (2b) If $q \leqslant y_{i-1} + y_i \leqslant 2(q-1)$, from the constraint $y'_{i-1} = y_{i-1} + y_i \pmod q$, it implies $y'_{i-1} + q = y_{i-1} + y_i$. We then have

$$\begin{aligned}\Delta &= \mathrm{Syn}(\boldsymbol{y}) - \mathrm{Syn}(\boldsymbol{y}') \pmod{qn} \\ &= \Big((i-1)y_{i-1} + iy_i - (i-1)y'_{i-1}\Big) + \sum_{j=i}^{n-1} y'_j \\ &= (i-1)q + y_i + \sum_{j=i}^{n-1} y'_j \\ &= (i-1)q + (q - y_{i-1}) + y'_{i-1} + \sum_{j=i}^{n-1} y'_j \\ &= (i-1)q + (q - y_{i-1}) + s - \sum_{j=1}^{i-2} y'_j \\ &= q + s + (q - y_{i-1}) + \Big((i-2)q - \sum_{j=1}^{i-2} y'_j\Big) \\ &> q + s.\end{aligned}$$

Therefore, given the computed values Δ and s, we can distinguish all three cases: case 1, case (2a) and case (2b). Moreover, observe that both s_R and $iq + s_R$ are monotonic functions in the index i. Particularly, it is easy to verify that s_R is decreasing in the index i while $iq + s_R$ is increasing function in the index i. For example, if the case (2a) happens, we can show that if a sequence $\boldsymbol{x}_1$ is obtained from $\boldsymbol{x}$ via a deletion at $x_{i_1} = \gamma$ and a sequence $\boldsymbol{x}_2$ is obtained from $\boldsymbol{x}$ via a deletion at $x_{i_2} = \gamma$ for some $i_1 < i_2$ and $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ then we must have $\Delta_1 > \Delta_2$, where $\Delta_1 = \mathrm{Syn}(\boldsymbol{y}) - \mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x}_1)) \pmod{qn}$ and $\Delta_2 = \mathrm{Syn}(\boldsymbol{y}) - \mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x}_2)) \pmod{qn}$. As shown earlier, we observe that $\Delta_1 = \sum_{j=i_1}^{n} y_j$ while $\Delta_2 = \sum_{j=i_2}^{n} y_j$. Thus, $\Delta_1 - \Delta_2 = \sum_{j=i_1}^{i_2-1} y_j \geqslant 0$, or $\Delta_1 \geqslant \Delta_2$. Note that $\Delta_1 = \Delta_2$ if and only if $y_j = 0$ for all $i_1 \leqslant j \leqslant i_2 - 1$. In other words, we have $x_{i_1} = x_{i_1+1} = \ldots = x_{i_2-1} = x_{i_2}$, or $\boldsymbol{x}_1 \equiv \boldsymbol{x}_2$. We have a contradiction. Therefore, given the value of $x_i = \gamma$, there is a unique value of i according to the value of Δ. Similarly, we can show that $iq + s_R$ is an increasing function in the index i for case (2b). Now, to locate the error in $\boldsymbol{y}$, for (2a), the decoder scans $\boldsymbol{y}'$ and simply searches for the largest index h where $\sum_{j=h}^{n-1} y'_j > \Delta$, while for (2b), the decoder scans $\boldsymbol{y}'$ and simply searches for the largest index h where $qh + \sum_{j=h}^{n-1} y'_j < \Delta$. The error location in $\boldsymbol{x}$ is then $i = h + 1$.

In conclusion, the code $\mathrm{Diff\_VT}_a(n; q)$ can correct a single deletion (or equivalently, a single insertion). ■

One may modify the decoding algorithm in the proof of Theorem 3 to obtain a similar decoding algorithm to correct an insertion. Particularly, from the received sequence, we can identify the value of the inserted symbol by using Lemma 2 (the parity check lemma). One may develop a similar result as in Lemma 1 to analyze the error behavior in the differential vector after one insertion. On the other hand, since Theorem 3 is proven and given the knowledge that $\mathrm{Diff\_VT}_a(n; q)$ can correct a single deletion or single insertion, there is an alternative linear-time decoding algorithm that can be applied to correct either a deletion or an insertion as follows. After using Lemma 2, the value of the deleted symbol (or inserted symbol) is known, and suppose that it is γ. The decoder then simply inserts (or removes) a symbol γ into (or from) an arbitrary position in the received sequence. Note that there is a unique codeword that can be obtained so that the Syndrome constraint is satisfied.

Remark 1. Recall the construction of a binary VT code from Construction 1 for given $n > 0$ and an arbitrary integer $a \in \mathbb{Z}_{n+1}$, we have

$$\mathrm{VT}_a(n) = \Big\{\boldsymbol{x} \in \{0,1\}^n : \mathrm{Syn}(\boldsymbol{x}) = a \pmod{(n+1)}\Big\}.$$

In [20], Levenshtein showed that the construction can be extended over an arbitrary modulo $N \geqslant n+1$ as follows. Given $n > 0, N \geqslant n+1$, and an arbitrary integer $a \in \mathbb{Z}_N$, we have the set

$$\mathrm{VT}^*_a(n, N) = \Big\{\boldsymbol{x} \in \{0,1\}^n : \mathrm{Syn}(\boldsymbol{x}) = a \pmod{N}\Big\},$$

that can correct a single deletion or single insertion (refer to Theorem 1, [20]). Similarly, it is easy to show that our construction can also be extended to over modulo qN for arbitrary $N \geqslant n$. We formally state the result in Corollary 1.

Corollary 1 (The modified q-ary Differential VT codes). *Given n, q. For an arbitrary $N \geqslant n, a \in \mathbb{Z}_{qN}$, set*

$$\mathrm{Diff\_VT}^*_a(n, N; q) \triangleq \Big\{\boldsymbol{x} \in \Sigma_q^n : \mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x})) = a \pmod{qN}\Big\}.$$

We then have $\mathrm{Diff\_VT}^*_a(n, N; q)$ is a single deletion/insertion correcting code.

Remark 2. One may construct a code $\mathrm{Diff\_VT}_a(n; q)$ using different variations of the differential function $\mathrm{Diff}(\boldsymbol{x})$ as follows. For all values p, $1 \leqslant p \leqslant q-1$ and $\gcd(p, q) = 1$, this coding method works for all *p-transformation vector* $\Gamma_p(\boldsymbol{x})$, defined as $y_i = p(x_i - x_{i+1}) \pmod q$ for $1 \leqslant i \leqslant n-1$, and $y_n = px_n$. Another variation of the differential vector was used in [15], [24] for binary codes to correct a burst of at most two deletions.

We now illustrate the error-decoding procedure through the following examples.

Example 2. Given $n = 10, q = 4, a = 0$, $\Sigma_4 = \{0, 1, 2, 3\}$. Consider a codeword $\boldsymbol{x} = 0103112013 \in \text{Diff\_VT}_0(10; 4)$. We obtain $\boldsymbol{y} = \text{Diff}(\boldsymbol{x}) = 3112032323$. It is easy to verify that $\text{Syn}(\boldsymbol{y}) = 120 \equiv 0 \pmod{40}$ and $\sum_{i=1}^{10} x_i \equiv 0 \pmod 4$.

Suppose that we receive $\boldsymbol{x}' = 013112013$, i.e. a deletion occurs at $x_3 = 0$. We then obtain $\boldsymbol{y}' = \text{Diff}(\boldsymbol{x}') = 322032323$. Now, to correct $\boldsymbol{x}$ and find out the value of i, we follow the decoding procedure in Theorem 3 as follows.

- From $\boldsymbol{x}'$, the decoder finds the value of the deleted symbol, which is $a - \sum_{i=1}^{n-1} x'_i = 0 \pmod 4$.
- From $\boldsymbol{y}' = \text{Diff}(\boldsymbol{x}') = 322032323$, the decoder computes:

$$\Delta = a - \text{Syn}(\boldsymbol{y}') = 0 - 104 = 16 \pmod{40},$$
$$s = \sum_{i=1}^{n-1} y'_i = 3 + 2 + 2 + 3 + 2 + 3 + 2 + 3 = 20.$$

- Since $\Delta < s$, the decoder concludes that it belongs to the case (2a) where the deletion is not at the first position, i.e. $i \neq 1$, and $y_{i-1} + y_i < q = 4$.
- Find the error location in $\boldsymbol{y}$. It can be observed that $\sum_{h=2}^{9} y'_i = 17 > \Delta = 16$ while $\sum_{h=3}^{9} y'_i = 15 < \Delta$. The decoder then concludes that the error in $\boldsymbol{y}$ is at the $h = 2$ position, and hence, the error in $\boldsymbol{x}$ is at $i = h + 1 = 3$.
- To correct $\boldsymbol{x}$, it inserts the symbol 0 to the third position.

We now consider another case, where we receive a sequence $\boldsymbol{x}' = 010311213$, i.e. a deletion occurs at $x_8 = 0$. We then obtain $\boldsymbol{y}' = \text{Diff}(\boldsymbol{x}') = 311203123$. We verify that $y_7 y_8 = 23$ has been replaced to $y'_7 = y_2 + y_3 = 1$ in $\boldsymbol{y}'$. Now, to correct $\boldsymbol{x}$ and find out the value of i, we follow the decoding procedure in Theorem 3 as follows.

- From $\boldsymbol{x}'$, the decoder finds the value of the deleted symbol, which is $a - \sum_{i=1}^{n-1} x'_i = 0 - (0 + 1 + 0 + 3 + 1 + 1 + 2 + 1 + 3) = 0 \pmod 4$.
- From $\boldsymbol{y}' = \text{Diff}(\boldsymbol{x}') = 311203123$, the decoder computes:

$$\Delta = a - \text{Syn}(\boldsymbol{y}') = 0 - 84 = 36 \pmod{40},$$
$$s = \sum_{i=1}^{n-1} y'_i = 3 + 1 + 1 + 2 + 3 + 1 + 2 + 3 = 16.$$

- Since $\Delta > s + q$, the decoder concludes that it belongs to the case (2b) where the deletion is not at the first position, i.e. $i \neq 1$, and $y_{i-1} + y_i > q = 4$.
- Find the error location in $\boldsymbol{y}$. It can be observed that $7 \times 4 + \sum_{h=7}^{9} y'_i = 34 < \Delta = 36$ while $8 \times 4 \sum_{h=8}^{9} y'_i = 37 > \Delta$. The decoder then concludes that the error in $\boldsymbol{y}$ is at the $h = 7$ position, and hence, the error in $\boldsymbol{x}$ is at $i = h + 1 = 8$.
- To correct $\boldsymbol{x}$, it inserts the symbol 0 to the 8th position.

Example 3. We now consider a special case when the deleted symbol belongs to a run of identical symbols. Given $n = 10, q = 3, a = 7$, and a codeword $\boldsymbol{x} = 0102122200 \in \text{Diff\_VT}_7(10; 3)$. Suppose that we receive $\boldsymbol{x}' = 010212200$, i.e. one can consider a deletion occurs at either x_6, or x_7, or x_8. We observe that $\boldsymbol{y} = \text{Diff}(\boldsymbol{x}) = 2111200200$ and $\boldsymbol{y}' = \text{Diff}(\boldsymbol{x}') = 211120200$.

The decoding procedure is as follows.

- The decoder computes $\Delta = a - \text{Syn}(\boldsymbol{y}') = 2 \pmod{30}$ and $s = \sum_{j=1}^{9} y'_j = 9$. Since $\Delta < s$, the decoder concludes that it belongs to the case (2a).
- Observe that $\sum_{h=5}^{9} y'_i = 4 > \Delta = 2$ while $\sum_{h=8}^{9} y'_i = 0 < \Delta = 2$ (*). The first index where $\sum_{j=h}^{n-1} y'_j > \Delta$ is then $h = 5$, i.e. the error in $\boldsymbol{x}$ is at $i = h + 1 = 6$. On the other hand, one may also select $h = 6, 7$ according to (*), i.e. the error is at x_7 or x_8, respectively. Nevertheless, we obtain the same codeword $\boldsymbol{x} = 0102122200$.

Remark 3. It is easy to show that our constructed codes $\text{Diff\_VT}_a(n; q)$, from Construction 3, also support a systematic linear-time encoder. The design is similar to the construction of the systematic encoder proposed by Tenengolts [22]. For message $\boldsymbol{x} \in \Sigma_q^k$, the encoder appends the information of the VT syndrome of the differential vector of $\boldsymbol{x}$ (of length $m + 1 = \lceil \log_q n \rceil + 1$) into its suffix. In addition, there is a marker of length two, which serves as a separator between the data part and the redundancy part (refer to [22]). We illustrate the main idea of the encoder in Figure 1a.

B. More Efficient Encoder and Decoder of The Differential VT codes

In this section, we present a linear-time encoder that encodes user data into the constructed differential VT codes $\text{Diff\_VT}_a(n; q)$ with only $\lceil \log_q n \rceil + 1$ redundant symbols.

The differential VT encoder $\text{ENC}_{\text{Diff\_VT}}$

INPUT: n, q, and $a \in \mathbb{Z}_{qn}$, a sequence $\boldsymbol{x} \in \Sigma_q^k$, where $k \triangleq n - \lceil \log_q n \rceil - 1$

OUTPUT: $\boldsymbol{c} \triangleq \text{ENC}_{\text{Diff\_VT}}(\boldsymbol{x}) \in \text{Diff\_VT}_a(n; q)$

(I) Set $m \triangleq \lceil \log_q n \rceil$ and $S \triangleq \{q^{j-1} : j \in [m]\} \cup \{n\}$ and $I \triangleq [n] \setminus S$. In other words, the set S includes the nth index and all the indices that are powers of q.

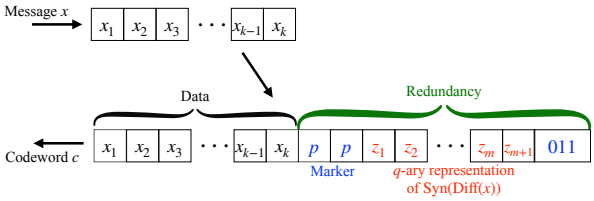
(II) Set $\boldsymbol{y} = y_1 y_2 \dots y_n \in \Sigma_q^n$, where $\boldsymbol{y}|_I = \boldsymbol{x}$ and $\boldsymbol{y}|_S = 0$. In other words, the symbols in $\boldsymbol{x}$ are filled into $\boldsymbol{y}$ excluding indices in S (refer to Figure 1 (b)) and $y_j = 0$ for $j \in S$.

(III) Compute the difference $a' \triangleq a - \text{Syn}(\boldsymbol{y}) \pmod{qn}$.
In the next step, we modify $\boldsymbol{y}$, by setting suitable values for y_j where $j \in S$, to obtain $\text{Syn}(\boldsymbol{y}) = a \pmod{qn}$. Since $0 \leqslant a' \leqslant qn - 1$, we find β, $0 \leqslant \beta \leqslant q - 1$, to be the number such that $\beta n \leqslant a' < (\beta + 1)n$.

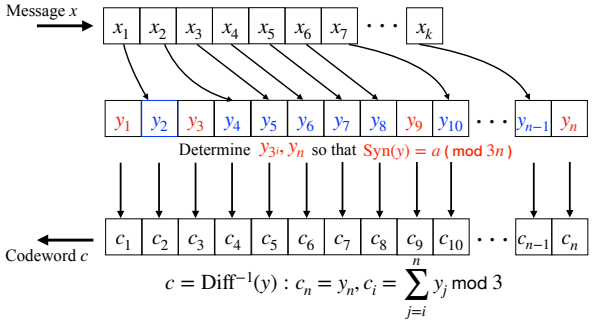
(IV) The values for y_j where $j \in S$ are set as follows.
- Set $y_n = \beta$, and $a'' = a' - \beta n < n$.
- Let $z_{t-1} \dots z_1 z_0$ be the q-ary representation of a''. Clearly, since $a'' < n$, the q-ary representation of a'' is of length at most $m = \lceil \log_q n \rceil$. We then have $a'' = \sum_{i=0}^{m-1} z_i q^i$.
- Set $y_{q^{j-1}} = z_{j-1}$ for $j \in [m]$.

(V) Set $\boldsymbol{c} = \text{Diff}^{-1}(\boldsymbol{y})$. In other words, we set $c_n = y_n$ and $c_i = \sum_{j=i}^{n} y_j \pmod q$ for $1 \leqslant i \leqslant n$.

(VI) Output $\boldsymbol{c}$.



(a) A systematic encoder for non-binary codes correcting a single deletion using the differential VT codes $\text{Diff\_VT}_a(n;q)$. Here $m = \lceil \log_q n \rceil$. The combination 011 at the end of the code sequence plays the role of the comma between transmitted sequences. The marker pp, where $p = x_k + 1 \pmod q$ serves as separators between the data part and the redundancy part. Here, the output codewords do not belong to the same coset of the differential VT codes.



(b) An example of our designed linear-time encoder to encode arbitrary messages into the differential VT codes $\text{Diff\_VT}_a(n;q)$ when $q = 3$. In general, the VT syndrome $\text{Syn}(\boldsymbol{y})$ is computed in modulo qn while each symbol is computed in modulo q. The set S includes index n and all powers of q. The message is of length $k = n - \lceil \log_q n \rceil - 1$. Here, the output codewords belong to the same coset of the differential VT codes, i.e. the information of a is known to the decoder.

Fig. 1: Our proposed encoders for non-binary codes correcting a single deletion using the differential VT codes $\text{Diff\_VT}_a(n;q)$. The construction of a systematic encoder is similar to the work proposed by Tenengoltz [22], both incur $\lceil \log_q n \rceil + 3$ redundant symbols, while our best encoder (in Figure (b)) uses only $\lceil \log_q n \rceil + 1$ redundant symbols.

Theorem 4. *Our constructed encoder* $\text{ENC}_{\text{Diff\_VT}}$ *is correct and has redundancy* $\lceil \log_q n \rceil + 1$ *symbols. In other words,* $\text{ENC}_{\text{Diff\_VT}}(\boldsymbol{x}) \in \text{VT}_a(n;q)$ *for all* $\boldsymbol{x} \in \Sigma_q^{n-\lceil \log_q n \rceil - 1}$.

Proof. We observe that the user message is of length $k = n - \lceil \log_q n \rceil - 1$, and hence, the redundancy of the encoder is $\lceil \log_q n \rceil + 1$ symbols. It remains to show that $\text{ENC}_{\text{Diff\_VT}}(\boldsymbol{x}) \in \text{VT}_a(n;q)$ for all $\boldsymbol{x} \in \Sigma_q^k$.

Suppose that $\boldsymbol{c} = \text{ENC}_{\text{Diff\_VT}}(\boldsymbol{x})$ for some $\boldsymbol{x} \in \Sigma_q^k$. It suffices to show that $\text{Syn}(\text{Diff}(\boldsymbol{c})) = a \pmod{qn}$. From Step (V) of the encoder $\text{ENC}_{\text{Diff\_VT}}$, $\boldsymbol{c} = \text{Diff}^{-1}(\boldsymbol{y})$, in other words, $\boldsymbol{y} = \text{Diff}(\boldsymbol{c})$. It remains to show that $\text{Syn}(\boldsymbol{y}) = a \pmod{qn}$.

Recall that from Step (I) of the encoder $\text{ENC}_{\text{Diff\_VT}}$, $S \triangleq \{q^{j-1} : j \in [m]\} \cup \{n\}$ and $I \triangleq [n] \setminus S$. Therefore,

$$\begin{aligned}
\text{Syn}(\boldsymbol{y}) &= \sum_{j \in S} j y_j + \sum_{j \in I} j y_j \pmod{qn} \\
&= \sum_{j \in [m]} q^{j-1} y_j + n y_n + \sum_{j \in I} j y_j \pmod{qn} \\
&= a'' + n\beta + (a - a') \pmod{qn} \\
&= (a' - \beta n) + n\beta + a - a' \pmod{qn} \\
&= a \pmod{qn}.
\end{aligned}$$

■

We illustrate the encoder $\text{ENC}_{\text{Diff\_VT}}$ via an example.

Example 4. Consider $n = 10, q = 3$ and $a = 0$. Then $m = \lceil \log_3 10 \rceil = 3$ and $k = 10 - 3 - 1 = 6$. Suppose that the message is $\boldsymbol{x} = 220011$ and we compute $\boldsymbol{c} = \text{ENC}_{\text{Diff\_VT}}(\boldsymbol{x}) \in \text{Diff\_VT}_0(10;3)$.

(I) Set $S = \{1, 3, 9, 10\}$ and $I = \{2, 4, 5, 6, 7, 8\}$.

(II) The encoder first sets $\boldsymbol{y} = y_1 2 y_3 20011 y_9 y_{10}$. It then sets $y_1 = y_3 = y_9 = y_{10} = 0$ to obtain $\boldsymbol{y} = 0202001100$ and computes $a' = a - \text{Syn}(\boldsymbol{y}) = 0 - 27 = 3 \pmod{30}$.

(III) Since $0 < a' = 3 < 10$, the encoder sets $\beta = 0$ and $a'' = a' = 3$. It then sets $y_{10} = \beta = 0$.

(IV) The 3-ary representation of 3 is then 010. Therefore, the encoder sets $y_1 = 0$, $y_3 = 1$, and $y_9 = 0$ to obtain $\boldsymbol{y} = 0212001100$. We can verify that $\text{Syn}(\boldsymbol{y}) = 0 \pmod{30}$.

(V) The encoder outputs $\boldsymbol{c} = \text{Diff}^{-1}(\boldsymbol{y}) = 1121222100$.

For completeness, we state the corresponding decoder.

The differential VT decoder $\text{DEC}_{\text{Diff\_VT}}$. Given n, q, and $a \in \mathbb{Z}_{qn}$, $m \triangleq \lceil \log_q n \rceil$ and $k \triangleq n - m - 1$. Given $\boldsymbol{c} = \text{ENC}_{\text{Diff\_VT}}(\boldsymbol{x})$ for some message $\boldsymbol{x} \in \Sigma_q^k$, and suppose the decoder receives a sequence $\boldsymbol{c}'$.

INPUT: $\boldsymbol{c}' \in \Sigma_q^{n-1} \cup \Sigma_q^n \cup \Sigma_q^{n+1}$
OUTPUT: $\boldsymbol{x} = \text{DEC}_{\text{Diff\_VT}}(\boldsymbol{c}') \in \Sigma_q^k$

(I) The decoder follows the error-decoding procedure in Theorem 3 to obtain $\boldsymbol{c} \triangleq \text{DEC}_{\text{error}}(\boldsymbol{c}') \in \Sigma_q^n$.

(II) Set $\boldsymbol{y} = \text{Diff}(\boldsymbol{c}) \in \Sigma_q^n$, $y_i = c_i - c_{i+1} \pmod q$ for $1 \leqslant i \leqslant n-1$ and $y_n = c_n$.

(III) Set $S \triangleq \{q^{j-1} : j \in [m]\} \cup \{n\}$ and $I \triangleq [n] \setminus S$.

(IV) Output $\boldsymbol{x} = \boldsymbol{y}|_I \in \Sigma_q^k$.

To conclude this section, the efficiency of our proposed encoders, compared to previous works, is illustrated in Table II.

IV. Correcting a Burst of Fixed Length: The Differential Shifted VT Codes

For arbitrary fixed $t > 1$, binary codes correcting a burst of exactly t deletions were proposed in [16], [17]. Recently, Schoeny *et al.* [12] extended the construction of binary codes in [16] to the non-binary regime. To correct a burst of exactly t deletions, for both the binary and non-binary cases, a common idea is to represent the codewords of length n as a $t \times n/t$ codeword array, where t divides n. Thus, for a codeword $\boldsymbol{x}$, the codeword array $A_t(\boldsymbol{x})$ is formed by t rows and n/t columns. When n/t is not an integer, one can append a sufficient number of bits/symbols 0 into the suffix of each codeword (see [31]).

Encoder	Redundancy (in symbols)	Encoding/Decoding Complexity	Receiver Information on Code's Parameters	Encoder Output	Remark
Encoder proposed by Tenengolts [22] using $\mathrm{T}_{a,b}(n;q)$	$\lceil \log_q n \rceil + 3 + \lceil \log_q 3 \rceil$	$O(n)$	not available	not in $\mathrm{T}_{a,b}(n;q)$	systematic
Encoder proposed by Abroshan *et al.* [27] using $\mathrm{T}_{a,b}(n;q)$	$> \log_q n + \log_2 n$	$O(n)$	VT Syndrome and parity check	in $\mathrm{T}_{a,b}(n;q)$	systematic
Systematic encoder proposed in this work using $\mathrm{Diff\_VT}_a(n;q)$ (see Figure 1a)	$\lceil \log_q n \rceil + 3 + \lceil \log_q 3 \rceil$	$O(n)$	not available	not in $\mathrm{Diff\_VT}_a(n;q)$	systematic
Encoder $\mathrm{ENC}_{\mathrm{Diff\_VT}}$ proposed in this work using $\mathrm{Diff\_VT}_a(n;q)$ (see Theorem 4 and Figure 1b)	$\lceil \log_q n \rceil + 1$	$O(n)$	VT Syndrome and parity check	in $\mathrm{Diff\_VT}_a(n;q)$	non-systematic

TABLE II: Efficient encoders for q-ary codes correcting single deletion or insertion proposed in this work and and those in literature. For each design category, the most desirable option is highlighted in blue. Particularly, our proposed encoder ENC_2 incurs the least redundancy of $\lceil \log_q n \rceil + 1$ symbols. Here, the receiver information on code's parameters plays an important role in error-detecting and error-correcting procedure. For example, it may provide more efficient basis for the design of segmented deletion/insertion correcting codes (see [28]–[30]).

In this work, for simplicity, we assume that t divides n. Observe that a burst of t deletions deletes in $\boldsymbol{x}$ exactly one bit (in binary case) or one symbol (in a non-binary alphabet) from each row of the array $A_t(\boldsymbol{x})$.

$$A_t(\boldsymbol{x}) = \begin{bmatrix} x_1 & x_{t+1} & \cdots & x_{(j-1)t+1} & \cdots & x_{(n/t-1)t+1} \\ x_2 & x_{t+2} & \cdots & x_{(j-1)t+2} & \cdots & x_{(n/t-1)t+2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_t & x_{2t} & \cdots & x_{jt} & \cdots & x_n \end{bmatrix}.$$

Here, the ith row of the array is denoted by $A_t(\boldsymbol{x})_{\mathrm{i}}$, and the jth column of the array is denoted by $A_t(\boldsymbol{x})^{\mathrm{j}}$. We now briefly describe the coding methods in [12] to correct a burst of exactly t deletions in the general q-ary alphabet, $q \geqslant 2$. The overall coding strategy in [12] is split into two main parts.

- The first row in the array belongs to a q-ary VT code $\mathrm{T}_{a,b}(n;q)$ (refer to Construction 2, Section II) that can correct a single error. In addition, such a code has an additional *run-length-limited (RLL)* property, that restricts the longest run of identical symbols to be at most $\ell = \lceil \log_q n \rceil + O(1)$. The authors also showed that for sufficiently large n, there exists a *runlength-limited encoder* which uses only one redundancy symbol to enforce such an RLL property. A similar design of such an encoder for binary codes was proposed in [16], that enforces binary codewords of maximum run length at most $\lceil \log n \rceil + 3$ with only one redundant bit (see [16, Appendix B]). The method is based on the *sequence replacement technique*. The idea can be extended to non-binary codes whose maximum runlength is at most $\lceil \log_q n \rceil + 3$ with only one redundant symbol (for example, see [10]).
- Each of the remaining $(t-1)$ rows in the array is then encoded using a modified version of the VT code, which they refer as *shifted VT (SVT) code*. This code corrects a single deletion in each row provided the location of the error is known to be within P consecutive positions. To obtain the desired redundancy, Schoeny *et al.* also set $P = \ell + 1 = \lceil \log_q n \rceil + O(1)$.

Lemma 3 (Nguyen *et al.* [10])**.** *Given n, q, $\ell = \lceil \log_q n \rceil + 3$. There exist a linear-time encoder $\mathrm{ENC}_{\ell\_\mathrm{RLL}} : \Sigma_q^{n-1} \to \Sigma_q^n$ and a corresponding decoder $\mathrm{DEC}_{\ell\_\mathrm{RLL}} : \Sigma_q^n \to \Sigma_q^{n-1}$ such that the following conditions hold:*

- *For all $\boldsymbol{x} \in \Sigma_q^{n-1}$, we have $\mathrm{DEC}_{\ell\_\mathrm{RLL}} \circ \mathrm{ENC}_{\ell\_\mathrm{RLL}}(\boldsymbol{x}) = \boldsymbol{x}$,*
- *If $\boldsymbol{c} = \mathrm{ENC}_{\ell\_\mathrm{RLL}}(\boldsymbol{x})$ then the maximum run of identical symbols in $\boldsymbol{c}$ is at most ℓ.*

The redundancy of $\mathrm{ENC}_{\ell\_\mathrm{RLL}}$ is one redundant symbol.

Definition 7 (Refer to [12], [16])**.** A *P-bounded single-deletion-correcting code* is a code in which the decoder can correct a single deletion given knowledge of the location of the deleted symbol to within P consecutive positions, i.e. prior knowledge that the index of the deletion error lies in an interval $[i, 1+1, ..., i+P-1]$, for a specified $i \in [1, n-P+1]$.

Formally, the following results were provided by Schoeny *et al.* [12]. Recall that the signature vector of a q-ary vector $\boldsymbol{x}$ of length n is a binary vector $\alpha(\boldsymbol{x})$ of length $n-1$, where $\alpha(x)_i = 1$ if $x_{i+1} \geq x_i$, and 0 otherwise, for $i \in [n-1]$.

Construction 4 (q-ary Shifted VT Codes [12])**.** *For $0 \leq a \leqslant P$ and $0 \leqslant b < q$, $c \in \{0,1\}$, the q-ary shifted VT code $\mathrm{SVT}_{a,b,c}(n,P,q)$ is defined as:*

$$\mathrm{SVT}_{a,b,c}(n,P,q) \triangleq \Big\{ \boldsymbol{x} \in \Sigma_q^n : \mathrm{Syn}(\alpha(\boldsymbol{x})) = a \pmod{(P+1)},\ \textit{and} \sum_{i=1}^{n} x_i = b \pmod{q},\ \textit{and} \sum_{i=1}^{n-1} \alpha(x)_i = c \pmod{2} \Big\}.$$

Lemma 4 (Schoeny *et al.* [12])**.** *The code $\mathrm{SVT}_{a,b,c}(n,P,q)$ is a P-bounded single deletion correcting code.*

Theorem 5 (Schoeny *et al.* [12])**.** *There exists a q-ary code correcting a burst of exactly t deletions whose number of redundancy symbols is at most*

$$\log_q(n/t) + (t-1)\log_q(2(\log_q(n/t)+6)+t+1.$$

In term of bits, the redundancy is $\log n + (t-1)\log\log n + O(t\log q)$ bits.

As discussed in Section III, a drawback is the difficulty of enforcing VT syndrome over the signature vectors of the codewords. In this section, we extend the idea of the q-ary differential VT codes to construct the *q-ary differential shifted VT codes*, which are P-bounded single deletion correcting codes, but more importantly, they support more efficient encoding and decoding procedures.

A. The Differential Shifted VT Codes

A new version of the q-ary shifted VT codes is as follows.

Construction 5 (**q-ary Differential Shifted VT Codes**)**.** *For $0 \le a < q(P+1)$ and $0 \leqslant b \leqslant q$, the q-ary differential shifted VT code* $\mathrm{Diff\_SVT}_{a,b}(n;q,P)$ *is defined as:*

$$\mathrm{Diff\_SVT}_{a,b}(n;q,P) \triangleq \Big\{\boldsymbol{x}\in\Sigma_q^n :\ \text{if } \boldsymbol{y}=\mathrm{Diff}(\boldsymbol{x})\in\Sigma_q^n \text{ then } \mathrm{Syn}(\boldsymbol{y}) = a \pmod{q(P+1)} \text{ and } \sum_{i=1}^{n} y_i = b \pmod{(q+1)}\Big\}.$$

Lemma 5. *The code* $\mathrm{Diff\_SVT}_{a,b}(n;q,P)$ *is a P-bounded single deletion correcting code.*

Proof. Similar to the proof of Lemma 2, we have that if $\mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x})) = a \pmod{q(P+1)}$ then we also have the parity check property, which consequently gives us the information of the deleted symbol:

$$x_1 + x_2 + \ldots + x_{n-1} + x_n = a \pmod{q}.$$

Suppose that we receive the sequence $\boldsymbol{x}' \in \mathcal{B}(\boldsymbol{x})$ of length $n-1$, the deleted symbol is γ, which can be determined from the parity check property, and the location of the error is within $L = [i, i+1, \ldots, i+P-1]$ for some $i \geqslant 1$ and $i+P-1 \leqslant n-1$. Recall that according to Lemma 1, for some $j > 1$, a deletion at symbol x_j replaces two symbols y_{j-1}, y_j with $y_{j-1}+y_j \pmod{q}$. Consequently, given $\boldsymbol{y}' = \mathrm{Diff}(\boldsymbol{x}')$ and the values of q and b, we can verify if $y_{j-1}+y_j \leqslant q-1$ or $y_{j-1}+y_j \geqslant q$ as follows:

- If $y_{j-1}+y_j \leqslant q-1$ then when $y_{j-1}y_j$ is replaced by $y_{j-1}+y_j \pmod{q}$, there is no change in the sum of symbols in the differential vector. We must have
$$\sum_{h=1}^{n-1} y'_h = b \pmod{(q+1)}. \tag{1}$$
- On the other hand, if $y_{j-1}+y_j \geqslant q$ we observe that $y_{j-1}y_j$ is replaced by the new symbol $y_{j-1}+y_j-q$, and hence,
$$\sum_{h=1}^{n-1} y'_h = b-q \pmod{(q+1)} \neq b \pmod{(q+1)}. \tag{2}$$

Now, assume that there are at least two locations in L to insert the deleted symbol γ, i.e we obtain two different sequences $\boldsymbol{x}_1$ (by inserting γ at index j_1) and $\boldsymbol{x}_2$ (by inserting γ at index j_2) for some $i \leqslant j_1 < j_2 \leqslant i+P-1$ so that all the code's constraints are satisfied, i.e.

$$\boldsymbol{x}_1 = (x'_1,\ldots,x'_{j_1-1},\gamma,x'_{j_1},\ldots x'_{j_2-1},x'_{j_2},\ldots,x'_{n-1}), \text{ and}$$
$$\boldsymbol{x}_2 = (x'_1,\ldots,x'_{j_1-1},x'_{j_1},\ldots x'_{j_2-1},\gamma,x'_{j_2},\ldots,x'_{n-1}).$$

Let $\boldsymbol{u} = \mathrm{Diff}(\boldsymbol{x}_1)$ and $\boldsymbol{v} = \mathrm{Diff}(\boldsymbol{x}_2)$. It is easy to see that $u_j = v_j$ for $j \leqslant j_1-2$ or $j \geqslant j_2+1$. We consider two cases.

Case 1. If $j_1 > 1$. As shown in (1) and (2), if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ share the same code's constraints, we must have $u_{j_1-1}+u_{j_1} = v_{j_2-1}+v_{j_2}$ since the values of $u_{j_1-1}+u_{j_1}$ and $v_{j_2-1}+v_{j_2}$ can be determined given the knowledge of $\boldsymbol{y}'$, q and b. Consequently, it implies that $\sum_{j=j_1-1}^{j_2} u_j = \sum_{j=j_1-1}^{j_2} v_j$. Next, from $\mathrm{Syn}(\boldsymbol{u}) = \mathrm{Syn}(\boldsymbol{v}) \pmod{q(P+1)}$, we have

$$\sum_{j=1}^{j_1-2} ju_j + \sum_{j=j_1-1}^{j_2} ju_j + \sum_{j=j_2+1}^{n} ju_j = \sum_{j=1}^{j_1-2} jv_j + \sum_{j=j_1-1}^{j_2} jv_j + \sum_{j=j_2+1}^{n} jv_j \pmod{q(P+1)}.$$

It implies that

$$\sum_{j=j_1-1}^{j_2} ju_j = \sum_{j=j_1-1}^{j_2} jv_j \pmod{q(P+1)}.$$

We then have

$$(j_1-2)\Big(\sum_{j=j_1-1}^{j_2} u_j\Big) + \sum_{j=1}^{j_2-j_1+2} ju_{j+j_1-2} = (j_1-2)\Big(\sum_{j=j_1-1}^{j_2} v_j\Big) + \sum_{j=1}^{j_2-j_1+2} jv_{j+j_1-2} \pmod{q(P+1)},$$

or

$$\sum_{j=1}^{j_2-j_1+2} ju_{j+j_1-2} = \sum_{j=1}^{j_2-j_1+2} jv_{j+j_1-2} \pmod{q(P+1)}.$$

Thus, we obtain two sequences $\boldsymbol{x}_3, \boldsymbol{x}_4$ such that $\mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x}_3)) = \mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x}_4)) \pmod{q(P+1)}$, where

$$\boldsymbol{x}_3 = (x'_{j_1-1},\gamma,x'_{j_1},\ldots x'_{j_2-1},x'_{j_2}), \text{ and}$$
$$\boldsymbol{x}_4 = (x'_{j_1-1},x'_{j_1},\ldots x'_{j_2-1},\gamma,x'_{j_2}).$$

Note that the length of $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ is j_2-j_1+2. In addition, we have $j_2-j_1+2 \leqslant (i+P-1)-i+2 = P+1$, and hence we conclude that $\boldsymbol{x}_3, \boldsymbol{x}_4 \in \mathrm{Diff\_VT}^*_a(j_2-j_1+2, P+1; q)$ for some $0 \leqslant a < q(P+1)$. Recall that such a code can correct a single deletion (refer to Corollary 1). On the other hand, we observe that $\boldsymbol{x}'' = (x'_{j_1-1}, x'_{j_1}, \ldots x'_{j_2-1}, x'_{j_2})$ can be obtained from both $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ by deleting the symbol γ. We have a contradiction.

It remains to consider the case when $j_1 = 1$.

Case 2. If $j_1 = 1$ and $1 < j_2 \leqslant P$, i.e. $i = 1$ and $L = [1, 2, \ldots P]$. Again, according to Lemma 1, if $\boldsymbol{u} = \mathrm{Diff}(\boldsymbol{x}_1)$

then u_1 is deleted for some $u_1 \leqslant q-1$. Consequently, if $\boldsymbol{v} = \mathrm{Diff}(\boldsymbol{x}_2)$, we also have $v_{j_2-1} + v_{j_2} \equiv u_1 \leqslant q-1$. Similarly, we obtain two sequences $\boldsymbol{x}'_3, \boldsymbol{x}'_4$ of length at most $(P+1)$ such that $\mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x}'_3)) = \mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x}'_4)) \pmod{q(P+1)}$, where

$$\boldsymbol{x}'_3 = (\gamma, x'_1, \ldots x'_{j_2-1}, x'_{j_2}), \text{ and}$$
$$\boldsymbol{x}'_4 = (x'_1, \ldots x'_{j_2-1}, \gamma, x'_{j_2}).$$

We have a contradiction. We conclude that there is at most one location to insert the deleted symbol γ into $\boldsymbol{x}'$, and thus, the constructed q-ary differential VT code $\mathrm{Diff\_SVT}_{a,b}(n; q, P)$ is a P-bounded single deletion correcting code. ■

Remark 4. We observe that our designed differential shifted VT codes $\mathrm{Diff\_SVT}_{a,b}(n; q, P)$ incur at most one more redundant symbol as compared to the q-ary shifted VT codes, proposed by Schoeny *et al.* [12] (refer to Construction 4). Particularly, the redundancy of a q-ary shifted VT code is $\log_q(P+1) + 1 + \log_q 2$ symbols while the redundancy of a differential shifted VT code in Construction 5 is then $\log_q(P+1) + 1 + \log_q(q+1)$ symbols. On the other hand, it provides an alternative simpler, and more efficient encoder (with the improvement of at least two redundant symbols as presented in Section III).

For completeness, we present an efficient encoder for the differential shifted VT codes $\mathrm{Diff\_SVT}_{a,b}(n; q, P)$, given arbitrary code parameters. Note that, in general, the value of P is $\log_q n + O(1) = o(n)$.

Differential SVT Encoder $\mathrm{ENC}_{\mathrm{Diff\_SVT}}$. Given n, q, P, where $3q(P+1) \leqslant n$. Given $0 \leq a < q(P+1)$ and $0 \leqslant b \leqslant q$. Set $m \triangleq \lceil \log_q q(P+1) \rceil$ and $k \triangleq n - m - 2$. The message is of length k, and hence, the redundancy of our encoder is then $m + 2 = \lceil \log_q q(P+1) \rceil + 2 \approx \lceil \log_q P \rceil + 3$.

INPUT: a sequence $\boldsymbol{x} \in \Sigma_q^k$, where $k \triangleq n - \lceil \log_q q(P+1) \rceil - 2$
OUTPUT: $\boldsymbol{c} \triangleq \mathrm{ENC}_{\mathrm{Diff\_SVT}}(\boldsymbol{x}) \in \mathrm{Diff\_SVT}_{a,b}(n; q, P)$

(I) Set index $i_0 = 2q(P+1)$ and $i_1 = 3q(P+1)$ where $i_0, i_1 \leqslant n$. Set $S \triangleq \{q^{j-1} : j \in [m]\} \cup \{i_0, i_1\}$ and $I \triangleq [n] \setminus S$.

(II) Consider $\boldsymbol{y}' \in \Sigma_q^n$, where $\boldsymbol{y}'|_I = \boldsymbol{x}$ and $\boldsymbol{y}'|_S = 0$. Compute the difference $a' \triangleq a - \mathrm{Syn}(\boldsymbol{y}') \pmod{q(P+1)}$. In the next step, we modify $\boldsymbol{y}'$ to obtain a codeword $\boldsymbol{y}$ with $\mathrm{Syn}(\boldsymbol{y}) = a \pmod{q(P+1)}$.

(III) Let $z_m \ldots z_1 z_0$ be the q-ary representation of a'. Here, we note that any number less than $q(P+1)$ has a representation of length at most $\lceil \log_q q(P+1) \rceil$. Suppose that $a' = \sum_{i=0}^{m} z_i q^i$. Then we set $y_{q^{j-1}} = z_{j-1}$ for $j \in [m]$.

(IV) Next, we set the symbols at the index $i_0 = 2q(P+1)$ and $i_1 = 3q(P+1)$ so that

$$y_{i_0} + y_{i_1} = b - \sum_{i \in [n] \setminus \{i_0, i_1\}} y_i \pmod{(q+1)}.$$

(V) Finally, we output $\boldsymbol{c} = \mathrm{Diff}^{-1}(\boldsymbol{y})$.

Theorem 6. *Given n, q, P, where $3q(P+1) \leqslant n$, $0 \leq a < q(P+1)$ and $0 \leqslant b \leqslant q$, the constructed encoder $\mathrm{ENC}_{\mathrm{Diff\_SVT}}$ is correct and has redundancy $\lceil \log_q q(P+1) \rceil + 2$ symbols. In other words, $\mathrm{ENC}_{\mathrm{Diff\_SVT}}(\boldsymbol{x}) \in \mathrm{Diff\_SVT}_{a,b}(n; q, P)$ for all $\boldsymbol{x} \in \Sigma_q^k$, where $k = n - \lceil \log_q q(P+1) \rceil - 2$.*

Proof. Suppose that $\boldsymbol{c} \triangleq \mathrm{ENC}_{\mathrm{Diff\_SVT}}(\boldsymbol{x})$ for some $\boldsymbol{x} \in \Sigma_q^k$. It suffices to show that

$$\mathrm{Syn}(\mathrm{Diff}(\boldsymbol{c})) = a \pmod{q(P+1)} \text{ and}$$
$$\sum_{i=1}^{n} \mathrm{Diff}(\boldsymbol{c})_i = b \pmod{(q+1)}.$$

From Step (V), we have $\boldsymbol{c} = \mathrm{Diff}^{-1}(\boldsymbol{y})$, i.e. $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{c})$, and it remains to show that $\mathrm{Syn}(\boldsymbol{y}) = a \pmod{q(P+1)}$. Recall that from Step (I), $S \triangleq \{q^{j-1} : j \in [m]\} \cup \{i_0 = 2q(P+1), i_1 = 3q(P+1)\}$ and $I \triangleq [n] \setminus S$. Therefore,

$$\begin{aligned}
\mathrm{Syn}(\boldsymbol{y}) &= \sum_{j \in S} j y_j + \sum_{j \in I} j y_j \pmod{q(P+1)} \\
&= \sum_{j \in [m]} q^{j-1} y_j + 2q(P+1) y_{2q(P+1)} + \\
&\quad + 3q(P+1) y_{3q(P+1)} + \sum_{j \in I} j y_j \pmod{q(P+1)} \\
&= a' + 0 + 0 + (a - a') \pmod{q(P+1)} \\
&= a \pmod{q(P+1)}.
\end{aligned}$$

In addition, from Step (IV), we have

$$y_{i_0} + y_{i_1} = b - \sum_{i \in [n] \setminus \{i_0, i_1\}} y_i \pmod{(q+1)},$$

and hence, it implies that

$$\sum_{i=1}^{n} y_i = b \pmod{(q+1)}, \text{ or}$$
$$\sum_{i=1}^{n} \mathrm{Diff}(\boldsymbol{c})_i = b \pmod{(q+1)}.$$ ■

Remark 5. We observe that reserving only one redundant symbol for the parity check constraint is not sufficient since the constraint is over modulo $(q+1)$. Similar to the construction of the differential VT decoder $\mathrm{DEC}_{\mathrm{Diff\_VT}}$, one can easily obtain a corresponding differential shifted VT decoder $\mathrm{DEC}_{\mathrm{Diff\_SVT}}$. We skip the detailed construction of such a decoder.

B. Codes Correcting a Burst of t Deletions with Efficient Encoder

We now present a construction of non-binary codes correcting a burst of t deletions, and the coding method is based on the differential VT codes and the differential shifted VT codes as presented in earlier sections. Recall that we represent the codewords of length n as $t \times n/t$ codeword arrays, where t divides n. Given $\ell \geqslant 1$, a code $\mathcal{C}$ is called ℓ-runlength limited if the maximum run of identical symbols in every codeword in $\mathcal{C}$ is at most ℓ.

Construction 6 (q**-ary** t**-burst-deletion correcting codes).** *Given* n, q. *Set* $\ell = \lceil \log_q n/t \rceil + 3, P = \ell + 1$. *For* $0 \leqslant a_1 < q(n/t)$, $0 \leq a_2 < q(P+1)$ *and* $0 \leqslant b \leqslant q$, *let* $\mathcal{C}_{a_1,a_2,b}(n,t;q)$ *be a set of all sequences* $\boldsymbol{x} \in \Sigma_q^n$ *such that the following constraints are satisfied:*

- **Constraints on the first row** $A_t(\boldsymbol{x})_1$*:*

(i) $A_t(\boldsymbol{x})_1 \in \text{Diff\_VT}_{a_1}(n/t;q)$, *and*

(ii) $A_t(\boldsymbol{x})_1$ *is* ℓ*-runlength limited,*

- **Constraints on the remaining rows:**

$$A_t(\boldsymbol{x})_{\text{i}} \in \text{Diff\_SVT}_{a_2,b}(n/t;q,P) \text{ for } 2 \leqslant i \leqslant t.$$

Theorem 7. *The code* $\mathcal{C}_{a_1,a_2,b}(n,t;q)$ *from Construction 6 can correct a burst of* t *deletions, and the redundancy is*

$$\log_q(n/t) + (t-1)\log_q\log_q(n/t) + O(t) \text{ (symbols)}.$$

In terms of bits, the redundancy is $\log n + (t-1)\log\log n + O(t\log q)$ *bits.*

Proof. The error-decoding procedure is similar to the construction of Schoeny *et al.* [12]. Suppose that a codeword $\boldsymbol{x} \in \mathcal{C}_{a_1,a_2,b}(n,t;q)$ suffers from a burst of t deletions and the received sequence is $\boldsymbol{x}' \in \Sigma_q^{n-t}$. Observe that a burst of t deletions deletes in $\boldsymbol{x}$ exactly one symbol from each row of the array $A_t(\boldsymbol{x})$. To recover the first row, the decoder sets $A_t(\boldsymbol{x}')_1 = (x'_1, x'_{t+1}, \ldots x'_{(n/t-2)t+1})$. Since the first row belongs to a differential VT code, the decoder can recover the first row $A_t(\boldsymbol{x})_1$ from $A_t(\boldsymbol{x}')_1$. Note that, after recovering the first row $A_t(\boldsymbol{x})_1$, the decoder may not identify exactly the location of the deletion since the deleted symbol would belong to a run of identical symbols. Nevertheless, since the maximum run of identical symbols in the first row is at most ℓ, we can locate the error in each of the other rows to be within at most $P = \ell + 1$ positions. Furthermore, since each of the remaining rows belongs to a differential shifted VT code, the decoder can recover each row accordingly.

It remains to compute the redundancy of our constructed code. The redundancy used for the first row in the array $A_t(\boldsymbol{x})$ is

$$r_1 = \log_q(qn/t) + 1 = \log_q(n/t) + 2 \text{ (symbols)}.$$

Here, $\log_q(qn/t)$ symbols are used to encode the differential VT code while one additional symbol is to enforce the runlength-limited constraint. On the other hand, the redundancy used for each of the other $(t-1)$ rows in the array $A_t(\boldsymbol{x})$ is

$$\begin{aligned} r_i &= \log_q(q(P+1)) + \log_q(q+1) \\ &= \log_q(\lceil \log_q n/t \rceil + 5) + 1 + \log_q(q+1) \\ &= \log_q\log_q(n/t) + O(1) \text{ (symbols) for } 2 \leqslant i \leqslant t. \end{aligned}$$

Thus, the total redundancy of such a code $\mathcal{C}_{a_1,a_2,b}(n,t;q)$ is $r_1 + \sum_{i=2}^{t} r_i = \log_q(n/t) + (t-1)\log_q\log_q(n/t) + O(t)$ symbols or $\log n + (t-1)\log\log n + O(t\log q)$ bits. ■

To conclude this subsection, we provide a linear-time encoder for such a code $\mathcal{C}_{a_1,a_2,b}(n,t;q)$ with given arbitrary code parameters. Observe that for $2 \leqslant i \leqslant t$, the ith row $A_t(\boldsymbol{x})_i$ can be encoded/decoded independently by using the differential SVT Encoder $\text{ENC}_{\text{Diff\_SVT}}$, since there is no joint constraint among these rows. The redundancy to encode each of these rows is then $\lceil \log_q q(P+1) \rceil + 2 \approx \lceil \log_q P \rceil + 3$ for any P. On the other hand, to encode the first row $A_t(\boldsymbol{x})_1$, we need to enforce the runlength-limited constraint with the differential VT syndrome property. Recall that the encoder $\text{ENC}_{\text{Diff\_VT}}$ for a differential VT code of length n (as presented in Section III-B) uses only $\lceil \log_q n \rceil + 1$ redundant symbols.

Lemma 6. *Given* n, q. *Set* $\ell' = \lceil \log_q n \rceil + 3$, *and* $k = n - \lceil \log_q n \rceil - 2$. *For an arbitrary sequence* $\boldsymbol{x} \in \Sigma_q^k$, *suppose that* $\boldsymbol{y} = \text{ENC}_{\ell'\text{\_RLL}}(\boldsymbol{x}) \in \Sigma_q^{k+1}$ *and* $\boldsymbol{c} = \text{ENC}_{\text{Diff\_VT}}(\boldsymbol{y}) \in \text{Diff\_VT}_a(n;q)$. *We then have the maximum run of identical symbols in* $\boldsymbol{c}$ *is at most* $\ell = 2\lceil \log_q n \rceil + 5$.

Proof. Note that the differential VT encoder $\text{ENC}_{\text{Diff\_VT}}$ of a code of length n uses only $\lceil \log_q n \rceil + 1$ redundant symbols at predetermined positions. Therefore, if the maximum run of identical symbols in $\boldsymbol{c} = \text{ENC}_{\text{Diff\_VT}}(\boldsymbol{y})$ is at least $\ell + 1 = 2\lceil \log_q n \rceil + 6$, in other words, $\text{Diff}(\boldsymbol{c})$ has at least $2\lceil \log_q n \rceil + 5$ consecutive zeros (by definition of a differential vector), then the sequence $\boldsymbol{y}$ (before inserting $\lceil \log_q n \rceil + 1$ redundant symbols) has a run of at least $2\lceil \log_q n \rceil + 5 - \lceil \log_q n \rceil - 1 = \lceil \log_q n \rceil + 4$ zeros. We have a contradiction since $\boldsymbol{y}$ is ℓ'-runlength limited. ■

According to Lemma 6, to construct a t-burst encoder, we can set the value of P to be $P = \ell + 1 = 2\lceil \log_q n \rceil + 6$, and amend the differential shifted VT code in the last $(t-1)$ rows and the corresponding encoder for such codes. For completeness, we present the detailed construction of a t-burst encoder as follows.

Input. Given q, n, $\ell' = \lceil \log_q n \rceil + 3$, $\ell = 2\lceil \log_q n \rceil + 5$, $P = \ell + 1 = 2\lceil \log_q n \rceil + 6$, $0 \leqslant a_1 < q(n/t)$, $0 \leq a_2 < q(P+1)$ and $0 \leqslant b \leqslant q$. The message $\boldsymbol{x} \in \Sigma_q^k$ is of length

$$\begin{aligned} k = &\Big(\underbrace{n/t - \lceil \log_q n/t \rceil - 2}_{\text{first row encoding}}\Big) + \\ &(t-1)\Big(\underbrace{n/t - \lceil \log_q q(P+1) \rceil - 2}_{i\text{th row encoding, } 2 \leqslant i \leqslant t}\Big) \\ = &\, n - \lceil \log_q n/t \rceil - (t-1)\Big(\lceil \log_q q(P+1) \rceil + 2\Big) - 2. \end{aligned}$$

We observe that for $P = 2\lceil \log_q n \rceil + 6$, the total redundancy is then $\log_q(n/t) + (t-1)\log_q\log_q(n/t) + O(t)$ symbols or $\log n + (t-1)\log\log n + O(t\log q)$ bits. The encoding procedure is as follows.

t-Burst-Encoder $\text{ENC}_{t\_\text{burst}}$.

INPUT: Given n, q, and a sequence $\boldsymbol{x} \in \Sigma_q^k$, where k is defined above

OUTPUT: $\boldsymbol{y} \triangleq \text{ENC}_{t\_\text{burst}}(\boldsymbol{x}) \in \mathcal{C}_{a_1,a_2,b}(n,t;q)$

(I) Suppose that $\boldsymbol{x} = \boldsymbol{x}_1\boldsymbol{x}_2\ldots\boldsymbol{x}_t$, where $\boldsymbol{x}_1$ includes the first $(n/t - \lceil \log_q n/t \rceil - 2)$ symbols in $\boldsymbol{x}$, and for $2 \leqslant i \leqslant t$, the sequence $\boldsymbol{x}_i$ is of length exactly $n/t - \lceil \log_q q(P+1) \rceil - 2$. We then set $k_1 = n/t - \lceil \log_q n/t \rceil - 2$, and set $k_2 = n/t - \lceil \log_q q(P+1) \rceil - 2$.

(II) **Encoding the first row in $A_t(\boldsymbol{x})$:**
- Obtain $\boldsymbol{x}'_1 = \text{ENC}_{\ell'\_\text{RLL}}(\boldsymbol{x}_1) \in \Sigma_q^{k_1+1}$
- Obtain $\boldsymbol{y}_1 = \text{ENC}_{\text{Diff\_VT}}(\boldsymbol{x}'_1) \in \text{Diff\_VT}_{a_1}(n/t; q)$

(III) **Encoding the ith row in $A_t(\boldsymbol{x})$:** for $2 \leqslant i \leqslant t$, we use the differential shifted VT encoder to obtain

$$\boldsymbol{y}_i = \text{ENC}_{\text{Diff\_SVT}}(\boldsymbol{x}_i) \in \text{Diff\_SVT}_{a_2,b}(n/t; q, P).$$

(IV) Finally, we output $\boldsymbol{c} = \boldsymbol{y}_1||\boldsymbol{y}_2||\dots||\boldsymbol{y}_t$ (the interleaved sequence of $\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_t$).

The following result is then immediate.

Theorem 8. *The encoder* $\text{ENC}_{t\_\text{burst}}$ *is correct. In other words, the output codewords belong to* $\mathcal{C}_{a_1,a_2,b}(n, t; q)$ *that is capable of correcting a burst of* t *deletions. The redundancy of the encoder is* $\log_q(n/t) + (t-1)\log_q\log_q(n/t) + O(t)$ *symbols or* $\log n + (t-1)\log\log n + O(t\log q)$ *bits.*

V. Correcting a Burst of Variable Length

In this section, we focus on the case $t = 2$, i.e. when there are at most two deletions. We first review the coding method of Wang *et al.* [24]. To correct a burst of at most two deletions, the authors represent the codewords of length n as a $\lceil \log q \rceil \times n$ codeword array, where each symbol in Σ_q is converted to its binary representation of length $\lceil \log q \rceil$. For a sequence $\boldsymbol{u} \in \Sigma_q^n$,

$$A(\boldsymbol{u}) = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{\lceil \log q \rceil} \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\lceil \log q \rceil,1} & x_{\lceil \log q \rceil,2} & \cdots & x_{\lceil \log q \rceil,n} \end{bmatrix}.$$

Therefore, the q-ary sequence $\boldsymbol{u}$ is converted to a binary matrix with $\lceil \log q \rceil$ rows and n columns. Observe that a burst of up to two deletions in $\boldsymbol{u}$ spans at most two consecutive columns in $A(\boldsymbol{u})$ and there is a burst of up to two deletions in each binary row. Similar to the case of correcting a burst of t deletions (as discussed in Subsection IV-A), the overall coding strategy in [24] is split into two main parts.

- The first row in the array belongs to a binary code that can correct a burst of at most two deletions, proposed by Levenshtein in 1967 in [15], that has redundancy $\log n + 1$ bits for codewords of length n. In addition, such a code has an additional *pattern length limited (PLL)* property, that restricts the maximum length of any substring with period 2 (refer to Definition 8) to be at most $\ell = \lceil \log n \rceil + O(1)$. The authors also showed that the redundancy to enforce both constraints in the first row is at most $\log n + 3$ (refer to Construction 4, Lemma 2, [15]).
- Each of the remaining $(\lceil \log q \rceil - 1)$ rows in the array belongs to a modified version of the binary shifted VT code, which can correct a burst of at most 2 deletions with the positional knowledge (within P positions) after recovering the first row. To obtain the desired redundancy, Schoeny *et al.* also set $P = \ell + 1 = \lceil \log n \rceil + O(1)$. The redundancy used in each of the remaining $(\lceil \log q \rceil - 1)$ rows is at most $\log\log n + O(1)$ bits.

The total redundancy of the coding scheme in [15] is $\log n + \log q \log\log n + O(\log q)$ bits. In this work, we use the idea of the differential shifted VT codes to further reduce the redundancy to construct a code correcting a burst of at most two deletions. The major difference in our coding scheme is that we view each q-ary sequence of length n as a matrix with only two rows and n columns. The mapping is designed as follows.

Given $q > 2$. Set $q' = \lceil q/2 \rceil$. For each symbol $x \in \Sigma_q$, *the decomposition* of x in $\Sigma_{q'}$ is $\tau(x) = (x_1, x_2)$ where $x_1 \in \{0, 1\}, x_2 \in \Sigma_{q'}$ and $x = x_1 q' + x_2$. For example, when $q = 3$, we have $\tau(0) = (0, 0), \tau(1) = (0, 1)$, and $\tau(2) = (1, 0)$. When $q = 6$, we have $\tau(0) = (0, 0), \tau(1) = (0, 1), \tau(2) = (0, 2), \tau(3) = (1, 0), \tau(4) = (1, 1), \tau(5) = (1, 2)$.

For a q-ary sequence $\boldsymbol{x}$ of length n where $\boldsymbol{x} = (x_1, x_2, \dots x_n)$, we view it as the following matrix:

$$D(\boldsymbol{x}) = \begin{bmatrix} \tau(x_1) & \tau(x_2) & \cdots & \tau(x_n) \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \end{bmatrix},$$

where the first row $D(\boldsymbol{x})_1 = (x_{1,1}, x_{1,2}, \dots, x_{1,n}) \in \{0, 1\}^n$, the second row $D(\boldsymbol{x})_2 = (x_{2,1}, x_{2,2}, \dots, x_{2,n}) \in \Sigma_{q'}^n$, and finally, the ith column $\tau(x_i) = (x_{i,1}, x_{i,2})^T$ for $1 \leqslant i \leqslant n$.

Our overall coding scheme is as follows.

- For the first row, we also use the binary codes proposed by Levenshtein in 1967 in [15] that can correct a burst of at most two deletions with the PLL constraint as proposed by Wang *et al.* [24].
- For the second row, which is a q'-ary sequence, where $q' = \lceil q/2 \rceil$, we then use the differential shifted VT codes to correct the error given the positional knowledge of the errors. Before presenting our main contribution, we summarize the result of Wang *et al.* [24], which is used in our construction for the first row.

Definition 8. A sequence $\boldsymbol{x} \in \Sigma_q^n$ is said to have period 2 if $x_i = x_{i+2}$ for all $1 \leqslant i \leqslant n-2$. A substring $\boldsymbol{u}$ of $\boldsymbol{x}$ is a subsequence of $\boldsymbol{x}$ consisting of consecutive symbols in $\boldsymbol{x}$.

Lemma 7 (Lemma 1, Lemma 2, Wang *et al.* [24])**.** *There exists a linear-time encodable and decodable binary code correcting a burst of at most two deletions such that the length of the longest substring with period 2 is at most* $\lceil \log n \rceil + 5$*, and the code redundancy is at most* $\log n + 3$ *bits.*

We now present our main construction of non-binary codes correcting a burst of at most two deletions with only $\log n + 3\log\log n + O(\log q)$ redundant bits. For simplicity, suppose that n is even.

Definition 9. For a sequence $\boldsymbol{x} = (x_1, x_2, \dots x_n) \in \Sigma_q^n$, given $i, s > 0$, we define the $(i; s)$-subsequence of $\boldsymbol{x}$, denoted by $\boldsymbol{x}_{(i;s)}$, as follows:

$$\boldsymbol{x}_{(i;s)} = \Big(x_i, x_{i+s}, x_{i+2s}, \dots, x_{i+s\lfloor (n-i)/s \rfloor}\Big).$$

We observe that when $i = s = 1$, we have $\boldsymbol{x}_{(1;1)} \equiv \boldsymbol{x} = (x_1, x_2, \dots x_n)$.

Construction 7 (q**-ary codes correcting at most two deletions).** *Given n, q. Let $\mathcal{C}_1$ be a code obtained from Lemma 7. Set $\ell = \lceil \log n \rceil + 5, P = \ell + 1$ and $q' = \lceil q/2 \rceil$. For $\boldsymbol{a} = (a_1, a_2, a_3)$ and $\boldsymbol{b} = (b_1, b_2, b_3)$, where $0 \leqslant a_1, a_2, a_3 < q'(P+1)$, $0 \leq b_1, b_2, b_3 \leqslant q'$, let $\mathcal{C}_{\boldsymbol{a},\boldsymbol{b}}(n, \leqslant 2; q)$ be a set of all q-ary sequences of length n such that for each codeword $\boldsymbol{c}$ the following conditions hold:*

- *For the first row $D(\boldsymbol{c})_1$, we must have $D(\boldsymbol{c})_1 \in \mathcal{C}_1$*
- *For the second row $D(\boldsymbol{c})_2$, suppose that $\boldsymbol{x} = D(\boldsymbol{c})_2 = (x_1, x_2, \ldots, x_n)$, we must have:*

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \text{Diff\_SVT}_{a_1,b_1}(n; q', P) \tag{3}$$

$$\boldsymbol{x}_{(1;2)} = (x_1, x_3, \ldots, x_{n-1}) \in \text{Diff\_SVT}_{a_2,b_2}(n/2; q', P), \tag{4}$$

$$\boldsymbol{x}_{(2;2)} = (x_2, x_4, \ldots, x_n) \in \text{Diff\_SVT}_{a_3,b_3}(n/2; q', P). \tag{5}$$

Theorem 9. *The code $\mathcal{C}_{\boldsymbol{a},\boldsymbol{b}}(n, \leqslant 2; q)$ from Construction 7 can correct a burst of at most two deletions, and there exist sufficient values of $a_1, a_2, a_3, b_1, b_2, b_3$ such that the redundancy of such a code $\mathcal{C}_{\boldsymbol{a},\boldsymbol{b}}(n, \leqslant 2; q)$ is $\log n + 3 \log \log n + O(\log q)$ bits.*

Proof. We first show that such a code from Construction 7 can correct a burst of at most two deletions by providing an error-decoding algorithm. Suppose that from the codeword $\boldsymbol{c} \in \mathcal{C}_{\boldsymbol{a},\boldsymbol{b}}(n, \leqslant 2; q)$, the received sequence is $\boldsymbol{c}'$. Clearly, from the length of $\boldsymbol{c}'$, we can conclude the number of errors that occurred. If the length of $\boldsymbol{c}'$ is n then there is no error.

If the length of $\boldsymbol{c}'$ is $n-1$, we conclude that there is a single deletion. Consequently, both rows $D(\boldsymbol{c})_1$ and $D(\boldsymbol{c})_2$ suffer exactly one deletion. Since $D(\boldsymbol{c})_1 \in \mathcal{C}_1$, which is a binary code capable of correcting a burst of up to 2 deletions, we can recover $D(\boldsymbol{c})_1$ uniquely. Next, since the maximum length of any substring with period 2 in $D(\boldsymbol{c})_1$ is at most $\lceil \log n \rceil + 5$, the maximum run of identical bits in $D(\boldsymbol{c})_1$ is also at most $\ell = \lceil \log n \rceil + 5$. We then conclude the location of the other error in $D(\boldsymbol{c})_2$ to be within determined P positions where $P = \ell + 1$. We then use the constraint (3) from the construction that $\boldsymbol{x} = D(\boldsymbol{c})_2 = (x_1, x_2, \ldots, x_n) \in \text{Diff\_SVT}_{a_1,b_1}(n; q', P)$ to correct the error in $D(\boldsymbol{c})_2$.

If the length of $\boldsymbol{x}'$ is $n-2$, we conclude that there is a burst of exactly two deletions. Consequently, both rows $D(\boldsymbol{c})_1$ and $D(\boldsymbol{c})_2$ suffer exactly two consecutive deletions. In addition, we conclude that each subsequence, $\boldsymbol{x}_{(1;2)}$ or $\boldsymbol{x}_{(2;2)}$, suffers exactly a single deletion. Since $D(\boldsymbol{c})_1 \in \mathcal{C}_1$, which is a binary code capable of correcting a burst of up to 2 deletions, we can recover $D(\boldsymbol{c})_1$ uniquely. Next, since the maximum length of any substring with period 2 in $D(\boldsymbol{c})_1$ is at most $\lceil \log n \rceil + 5$, we then conclude the location of the other error in $D(\boldsymbol{c})_2$ to be within determined P positions where $P = \ell + 1$. We then use the constraint (4) from the construction that $\boldsymbol{x}_{(1;2)} = (x_1, x_3, \ldots, x_{n-1}) \in \text{Diff\_SVT}_{a_2,b_2}(n/2; q', P)$ to correct the error in $\boldsymbol{x}_{(1;2)}$. Similarly, we use the constraint (5) from the construction that $\boldsymbol{x}_{(2;2)} = (x_2, x_4, \ldots, x_n) \in \text{Diff\_SVT}_{a_3,b_3}(n/2; q', P)$ to correct $\boldsymbol{x}_{(2;2)}$.

Thus, the code $\mathcal{C}_{\boldsymbol{a},\boldsymbol{b}}(n, \leqslant 2; q)$ from Construction 7 can correct a burst of at most two deletions. It remains to show the redundancy of our designed codes. According to Lemma 7, the redundancy used for the first row is at most $\log n + 3$ bits. On the other hand, the redundancy for a differential shifted VT code is $\lceil \log_{q'} q'(P+1) \rceil + 2$ symbols, or $\log P + O(\log q)$ bits (see Theorem 6). In our construction, $P = \ell + 1 = \lceil \log n \rceil + 6$, and hence, the redundancy for the second row to enforce three constraints (3), (4), and (5) is at most $3 \log \log n + O(\log q)$ bits. Consequently, there exist sufficient values of $a_1, a_2, a_3, b_1, b_2, b_3$ such that the redundancy of $\mathcal{C}_{\boldsymbol{a},\boldsymbol{b}}(n, \leqslant 2; q)$ is $\log n + 3 \log \log n + O(\log q)$ bits. ∎

Remark 6. The idea of Construction 7 can be extended to construct non-binary codes correcting a burst of up to t deletions. We still view each q-ary sequence $\boldsymbol{c}$ of length n as a matrix with exactly two rows and n columns $D(\boldsymbol{c})$. Similar to the work of Wang *et al.* [25], the first row belongs to a binary code that is capable of correcting a burst of up to t deletions (for example, refer to the work of Lenz *et al.* [18] for an efficient design of such a code) with an additional constraint to restrict the location of errors. Particularly, in [25], the authors show that it is possible to locate the errors within $O(\log n)$ positions using the concept of (w, δ)-*dense string*. We then design the constraints for the second rows as in Construction 7 to handle every single case of s deletions for any $s \leqslant t$. In general, we would need $t(t+1)/2$ such constraints, resulting in a redundancy of at most $t(t+1) \log \log n + o(\log q) = O(t^2 \log \log n)$ bits (see Example 5). However, the encoding and decoding procedures are much more complicated than the case of a burst of at most two errors. We defer the study of codes correcting a burst of up to t deletions and the design of efficient encoders for such codes to our future works.

Example 5. To correct a burst of at most three deletions, for each codeword $\boldsymbol{c}$, the first row $D(\boldsymbol{c})_1$ belongs to a binary code, which is capable of correcting a burst of at most three deletions with an additional constraint to locate the errors within $O(\log n)$ positions. On the other hand, the constraints for the second row $\boldsymbol{x} = D(\boldsymbol{x})_2 = (x_1, x_2, \ldots, x_n)$ are as follows:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \text{Diff\_SVT}_{a_1,b_1}(n; q', P) \tag{6}$$

$$\boldsymbol{x}_{(1;2)} = (x_1, x_3, \ldots, x_{n-1}) \in \text{Diff\_SVT}_{a_2,b_2}(n/2; q', P), \tag{7}$$

$$\boldsymbol{x}_{(2;2)} = (x_2, x_4, \ldots, x_n) \in \text{Diff\_SVT}_{a_3,b_3}(n/2; q', P), \tag{8}$$

$$\boldsymbol{x}_{(1;3)} = (x_1, x_4, x_7, \ldots) \in \text{Diff\_SVT}_{a_4,b_4}(n/3; q', P), \tag{9}$$

$$\boldsymbol{x}_{(2;3)} = (x_2, x_5, x_8, \ldots) \in \text{Diff\_SVT}_{a_5,b_5}(n/3; q', P), \tag{10}$$

$$\boldsymbol{x}_{(3;3)} = (x_3, x_6, x_9 \ldots) \in \text{Diff\_SVT}_{a_6,b_6}(n/3; q', P). \tag{11}$$

We observe that when there is exactly one deletion, the constraint (6) is sufficient to correct the error in the second row $\boldsymbol{x} = D(\boldsymbol{x})_2$. On the other hand, when there is a burst of two deletions, the decoder uses the constraints (7) and (8) to correct the errors in $\boldsymbol{x}_{(1;2)}$ and $\boldsymbol{x}_{(2;2)}$, accordingly. Similarly, when there is a burst of three deletions, the decoder uses the remaining constraints (9), (10), and (11) to correct the errors in $\boldsymbol{x}_{(1;3)}$, $\boldsymbol{x}_{(2;3)}$ and $\boldsymbol{x}_{(3;3)}$, when each of them suffers from a single deletion.

To conclude this section, we present an efficient encoder for non-binary codes correcting a burst of at most two deletions with $\log n + 3 \log \log n + O(\log q)$ redundant bits, which sig-

nificantly improves on the redundancy $\log q \log n + O(\log q)$ bits of the encoder in [25]. Recall that in Construction 7, for each codeword $\boldsymbol{c}$, the rows $D(\boldsymbol{c})_1$ and $D(\boldsymbol{c})_2$ can be encoded independently. While the construction for a binary code satisfying the constraints required in the first row was presented in [25], it remains to present an efficient encoding algorithm for the second row $D(\boldsymbol{c})_2$.

Note that the redundancy used in the differential shifted VT encoder $\text{ENC}_{\text{Diff\_SVT}}$ is $\lceil \log_q q(P+1) \rceil + 2$ symbols (see Theorem 6), where $\lceil \log_q q(P+1) \rceil$ symbols are used to enforce the syndrome constraint and the other two symbols are used to enforce the parity check constraint.

Construction 8. *Given n, q, P, where $q > 2$. Set $k = n - 3(\lceil \log_q q(P+1) \rceil + 2) - 7$, $m = \lceil \log_q q(P+1) \rceil$. We construct an encoder $\text{ENC}^* : \Sigma_q^k \to \Sigma_q^n$ as follows. Suppose that k is even and for a sequence $\boldsymbol{x} = (x_1, x_2, \ldots x_k) \in \Sigma_q^k$, we obtain:*

$$\begin{aligned} a_1 &= \text{Syn}(\text{Diff}(\boldsymbol{x})) \pmod{q(P+1)}, \\ a_2 &= \text{Syn}(\text{Diff}(\boldsymbol{x}_{(1;2)})) \pmod{q(P+1)}, \\ a_3 &= \text{Syn}(\text{Diff}(\boldsymbol{x}_{(2;2)})) \pmod{q(P+1)}, \end{aligned}$$

and

$$\begin{aligned} b_1 &= \sum_{i=1}^{k} \text{Diff}(\boldsymbol{x})_i \pmod{(q+1)}, \\ b_2 &= \sum_{i=1}^{k/2} \text{Diff}(\boldsymbol{x}_{(1;2)})_i \pmod{(q+1)}, \\ b_3 &= \sum_{i=1}^{k/2} \text{Diff}(\boldsymbol{x}_{(2;2)})_i \pmod{(q+1)}. \end{aligned}$$

Let $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3$ be the q-ary representation of length $m = \lceil \log_q q(P+1) \rceil$ of a_1, a_2, and a_3, respectively. On the other hand, let $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ be the q-ary representation of length 2 of b_1, b_2, and b_3, respectively. Recall the last symbol in $\boldsymbol{x}$ is x_k and suppose that the first symbol in $\boldsymbol{u}_1$ is β. Let γ be the smallest symbol in Σ_q that is different from x_k and β, and obtain a marker $M = (x_k, x_k, \gamma, \gamma, \gamma, \beta, \beta)$ of length 7. We then set $\text{ENC}^*(\boldsymbol{x}) \equiv \boldsymbol{x} M \boldsymbol{u}_1 \boldsymbol{v}_1 \boldsymbol{u}_2 \boldsymbol{v}_2 \boldsymbol{u}_3 \boldsymbol{v}_3 \in \Sigma_q^n$.

Theorem 10. *Let $\mathcal{C} = \left\{ \text{ENC}^*(\boldsymbol{x}) : \boldsymbol{x} \in \Sigma_q^k \right\}$. We then have that $\mathcal{C}$ can correct a burst of at most two deletions given the knowledge of the location of the deleted symbols to be within P consecutive positions.*

Proof. Suppose that $\boldsymbol{c} = \text{ENC}^*(\boldsymbol{x})$ for some $\boldsymbol{x} \in \Sigma_q^k$ and the decoder receives a sequence $\boldsymbol{c}'$, which is obtained from $\boldsymbol{c}$ via a burst of at most two deletions. Recall the construction of the marker $M = (x_k, x_k, \gamma, \gamma, \gamma, \beta, \beta)$ of length 7, hence, when there is a burst of at most two deletions, we must have $c'_k = x_k$, $c'_{k+3} = \gamma$ and $c'_{k+6} = \beta$. Therefore, given the received sequence $\boldsymbol{c}'$, the decoder is able to get the information of x_k, the last symbol in $\boldsymbol{x}$, the symbol γ, and finally β, which is the first symbol in $\boldsymbol{u}_1$. Base on the information of the marker M, it is able to locate the burst of at most two deletions, whether in $\boldsymbol{x}$, or in the marker M, or in the suffix $\boldsymbol{u}_1 \boldsymbol{v}_1 \boldsymbol{u}_2 \boldsymbol{v}_2 \boldsymbol{u}_3 \boldsymbol{v}_3$.

- If the errors occur at the marker M or in the suffix $\boldsymbol{u}_1 \boldsymbol{v}_1 \boldsymbol{u}_2 \boldsymbol{v}_2 \boldsymbol{u}_3 \boldsymbol{v}_3$, the decoder concludes that there is no error in $\boldsymbol{x}$ and simply takes the prefix of k symbols as the original sequence $\boldsymbol{x}$. To recover the suffix $\boldsymbol{u}_1 \boldsymbol{v}_1 \boldsymbol{u}_2 \boldsymbol{v}_2 \boldsymbol{u}_3 \boldsymbol{v}_3$, it proceeds to recompute $a_1, a_2, a_3, b_1, b_2, b_3$ as in Construction 8, and recover the suffix $\boldsymbol{u}_1 \boldsymbol{v}_1 \boldsymbol{u}_2 \boldsymbol{v}_2 \boldsymbol{u}_3 \boldsymbol{v}_3$.
- On the other hand, if the errors occur within the first k symbols in $\boldsymbol{x}$, the decoder concludes that there is no error in the suffix $\boldsymbol{u}_1 \boldsymbol{v}_1 \boldsymbol{u}_2 \boldsymbol{v}_2 \boldsymbol{u}_3 \boldsymbol{v}_3$. Based on the information of this suffix and given the knowledge of the location of the deleted symbols to be within P consecutive positions, the decoder follows the error-decoding procedure in Lemma 5 (refer to the q-ary differential shifted VT codes, Construction 5) to correct the errors in $\boldsymbol{x}$.

In conclusion, the code $\mathcal{C} = \left\{ \text{ENC}^*(\boldsymbol{x}) : \boldsymbol{x} \in \Sigma_q^k \right\}$ can correct a burst of at most two deletions given the knowledge of the location of the deleted symbols to be within P consecutive positions. ■

The following result is then immediate.

Corollary 2. *There exists a linear-time encoder* ENC *and a corresponding decoder* DEC *for non-binary codes correcting a burst of at most two deletions (or two insertions) with redundancy $\log n + 3 \log \log n + O(\log q)$ bits.*

VI. Conclusion

We have presented a new version of non-binary VT codes that are capable of correcting a single deletion or single insertion, providing an alternative simpler and more efficient encoder of the construction by Tenengoltz [22]. Our construction is based on the *differential vector*, and the codes are referred to as the *differential VT codes*. In addition, we have provided linear-time algorithms that encode user messages into these codes of length n over the q-ary alphabet for $q \geqslant 2$ with at most $\lceil \log_q n \rceil + 1$ redundant symbols, while the optimal redundancy required is at least $\log_q n + \log_q (q-1)$ symbols. Our designed encoder reduces the redundancy of the best-known encoder of Tenengoltz [22] by at least 2 redundant symbols or equivalently $2 \log_2 q$ bits.

Moreover, we have introduced the *q-ary differential shifted VT codes* to construct non-binary codes correcting a burst of deletions (or insertions). Particularly, when there are at most two errors, our designed codes incur $\log n + 3 \log \log n + O(\log q)$ redundant bits, which improves a recent result of Wang *et al.* [24] with redundancy $\log n + O(\log q \log \log n)$ bits for all $q \geqslant 8$. We have also presented an efficient encoder for codes correcting a burst of exactly t deletions (or insertions) for arbitrary $t \geqslant 1$, while the design of the encoder for codes correcting a burst of variable length (when the length is up to t for arbitrary $t > 3$) is deferred to our future work.

Acknowledgment

The authors would like to thank the editor and the reviewers for their constructive feedback and helpful suggestions.

REFERENCES

[1] T. T. Nguyen, K. Cai, and P. H. Siegel, "Every bit counts: a new version of non-binary VT codes with more efficient encoder", in *Proc. IEEE Intl. Conf. Commun.* (ICC), Rome, Italy, May 2023, pp. 5477-5482.

[2] Y. Ng, B. V. K. V. Kumar, K. Cai, S. Nabavi, and T. C. Chong, "Picketshift codes for bit-patterned media recording with insertion/deletion errors," in *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 2268-2271, Jun. 2010.

[3] W. Kang *et al.*, "Complementary skyrmion racetrack memory with voltage manipulation," in *IEEE Electron Device Lett.*, vol. 37, pp. 924-927, 2016.

[4] L. Dolecek and V. Anantharam, "Using Reed-Muller RM$(1, m)$ codes over channels with synchronization and substitution errors," in *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1430-1443, Apr. 2007.

[5] S. Agarwal, D. Starobinski, and A. Trachtenberg, "On the scalability of data synchronization protocols for PDAs and mobile devices," in *IEEE Netw.*, vol. 16, no. 4, pp. 22-28, Jul. 2002.

[6] S. Yazdi, H. M. Kiah, E. R. Garcia, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: trends and methods", in *IEEE Trans. Molecular, Biological, Multi-Scale Commun.*, vol. 1, no. 3, pp. 230-248, 2015.

[7] R. Heckel, G. Mikutis, and R. N. Grass, "A characterization of the DNA data storage channel," in *Sci. Rep.*, vol. 9, no. 1, pp. 1-12, Jul. 2019.

[8] K. Cai, Y. M. Chee, R. Gabrys, H. M. Kiah and T. T. Nguyen, "Correcting a single indel/edit for DNA-based data storage: linear-time encoders and order-optimality," in *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 3438-3451, June 2021, doi: 10.1109/TIT.2021.3049627.

[9] R. Gabrys, V. Guruswami, J. Ribeiro and K. Wu, "Beyond single-deletion correcting codes: substitutions and transpositions", in *IEEE Trans. Inf. Theory*, vol. 69, no. 1, pp. 169-186, Jan. 2023, doi: 10.1109/TIT.2022.3202856.

[10] T. T. Nguyen, K. Cai, K. A. Schouhamer Immink, and H. M. Kiah, "Capacity-approaching constrained codes with error correction for DNA-based data storage," *IEEE Trans. Inf. Theory*, vol. 67, no. 8, pp. 5602-5613, Aug. 2021, doi: 10.1109/TIT.2021.3066430.

[11] C. R. O'Donnell, H. Wang, and W. B. Dunbar, "Error analysis of idealized nanopore sequencing," in *Electrophoresis*, vol. 34, no. 15, pp. 2137-2144, Aug. 2013.

[12] C. Schoeny, F. Sala and L. Dolecek, "Novel combinatorial coding results for DNA sequencing and data storage," in *51st Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, USA, 2017, pp. 511-515, doi: 10.1109/ACSSC.2017.8335392.

[13] M. Zorzi and R. R. Rao, "On the impact of burst errors on wireless ATM," in *IEEE Pers. Commun.*, vol. 6, no. 4, pp. 65-76, Aug. 1999, doi: 10.1109/98.788217.

[14] Ching-Nung Yang, Chih-Yang Chiu, Tse-Shih Chen and Guo-Cin Ye, "Wireless image transmission using burst error correction codes," in *5th International Conference on Visual Information Engineering (VIE 2008)*, Xi'an, China, 2008, pp. 331-336, doi: 10.1049/cp:20080333.

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

[16] C. Schoeny, A. Wachter-Zeh, R. Gabrys and E. Yaakobi, "Codes correcting a burst of deletions or insertions," in *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 1971-1985, April 2017, doi: 10.1109/TIT.2017.2661747.

[17] L. Cheng, T. G. Swart, H. C. Ferreira and K. A. S. Abdel-Ghaffar, "Codes for correcting three or more adjacent deletions or insertions," in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), Honolulu, HI, USA, 2014, pp. 1246-1250, doi: 10.1109/ISIT.2014.6875032.

[18] A. Lenz and N. Polyanskii, "Optimal codes correcting a burst of deletions of variable length," in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), Los Angeles, CA, USA, 2020, pp. 757-762, doi: 10.1109/ISIT44484.2020.9174288.

[19] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors", in *Automatica i Telemekhanica*, vol. 26, no. 2, 1965.

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

[21] K. A. S. Abdel-Ghaffar and H. C. Ferreira, "Systematic encoding of the Varshamov-Tenengolts codes and the Constantin-Rao codes", in *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 340-345, 1998.

[22] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion", in *IEEE Trans. Inf. Theory*, vol. 30, no. 5, pp. 766-769, 1984.

[23] V. I. Levenshtein, "Binary codes capable of correcting spurious insertions and deletions of ones," in *Prob. Inf. Trans.*, vol. 1, no. 1, pp. 8-17, Jan. 1965.

[24] S. Wang, J. Sima and F. Farnoud, "Non-binary codes for correcting a burst of at most 2 deletions," in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), 2021, pp. 2804-2809, doi: 10.1109/ISIT45174.2021.9517917.

[25] S. Wang, Y. Tang, J. Sima, R. Gabrys, F. Farnoud, "Non-binary codes for correcting a burst of at most t deletions", *arXiv*, arXiv:2210.11818, available online at https://doi.org/10.48550/arXiv.2210.11818.

[26] Y. M. Chee, H. M. Kiah, and T. T. Nguyen, "Linear-time encoders for codes correcting a single edit for DNA-based data storage", in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), Paris, France, Jul. 2019, pp. 772-776.

[27] M. Abroshan, R. Venkataramanan and A. G. I. Fabregas, "Efficient Systematic Encoding of Non-binary VT Codes," in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), 2018, pp. 91-95.

[28] M. Abroshan, R. Venkataramanan, and A. G. i Fabregas, "Coding for segmented edit channels," in *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3086-3098, Apr. 2018.

[29] Z. Liu and M. Mitzenmacher, "Codes for deletion and insertion channels with segmented errors," in *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 224-232, Jan. 2010.

[30] K. Cai, H. M. Kiah, M. Motani and T. T. Nguyen, "Coding for segmented edits with local weight constraints," in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), 2021, pp. 1694-1699.

[31] T. T. Nguyen, K. Cai, W. Song and K. A. Schouhamer Immink, "Optimal single chromosome-inversion correcting codes for data storage in live DNA," in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), Espoo, Finland, 2022, pp. 1791-1796, doi: 10.1109/ISIT50566.2022.9834376.

Tuan Thanh Nguyen (Member, IEEE) received the B.Sc. degree and the Ph.D. degree in mathematics from the Nanyang Technological University, Singapore, in 2014 and 2019, respectively. He is currently a Research Fellow at the Singapore University of Technology and Design (SUTD), with the Advanced Coding and Signal Processing (ACSP) Lab of SUTD. He was a Research Fellow at the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, from Aug 2018 to Sep 2019. His research interest lies in the interplay between applied mathematics and computer science/engineering, particularly including combinatorics and coding theory. His research project concentrates on error correction codes and constrained codes for communication systems and data storage systems, especially codes for DNA-based data storage.

Kui Cai (Senior Member, IEEE) received the B.E. degree in information and control engineering from Shanghai Jiao Tong University, Shanghai, China, and the joint Ph.D. degree in electrical engineering from the Technical University of Eindhoven, The Netherlands, and the National University of Singapore. She is currently an Associate Professor at the Singapore University of Technology and Design (SUTD). She received the 2008 IEEE Communications Society Best Paper Award in Coding and Signal Processing for Data Storage. She served as the Vice-Chair (Academia) of the IEEE Communications Society, Data Storage. She was listed in the 2020 Who's Who in Engineering Singapore. Her main research interests are in the areas of coding theory, information theory, and signal processing for emerging data storage systems and computing.

Paul H. Siegel (Life Fellow, IEEE) received the S.B. and Ph.D. degrees in mathematics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1975 and 1979, respectively. He held a Chaim Weizmann Postdoctoral Fellowship with the Courant Institute, New York University, New York, NY, USA. He was with the IBM Research Division, San Jose, CA, USA, from 1980 to 1995. He joined the faculty at the University of California San Diego, La Jolla, CA, USA, in 1995, where he is currently a Distinguished Professor in electrical and computer engineering with the Jacobs School of Engineering. He is affiliated with the Center for Memory and Recording Research, where he holds an Endowed Chair and served as the Director from 2000 to 2011. His research interests include information theory, coding techniques, and machine learning, with applications to digital data storage and transmission. He is a member of the National Academy of Engineering. He was a member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2009 to 2014. He was the 2015 Padovani Lecturer of the IEEE Information Theory Society. He was a co-recipient of the 1992 IEEE Information Theory Society Paper Award, the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award, and the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He served as an Associate Editor for Coding Techniques of the IEEE TRANSACTIONS ON INFORMATION THEORY from 1992 to 1995 and the Editor-in-Chief from 2001 to 2004. He served as a Co-Guest Editor of the 1991 Special Issue on Coding for Storage Devices of the IEEE TRANSACTIONS ON INFORMATION THEORY. He was also a Co-Guest Editor of the two-part 2001 Special Issue on The Turbo Principle: From Theory to Practice and the 2016 Special Issue on Recent Advances in Capacity Approaching Codes of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He was a Co-Lead Guest Editor of the 2023 Special Issue on Dimensions of Channel Coding: From Theory to Algorithms to Applications (Dedicated to the Memory of Alexander Vardy) of the IEEE JOURNAL ON SELECTED AREAS IN INFORMATION THEORY.