Optimal Systematic t-Deletion Correcting Codes

Jin Sima[1], Ryan Gabrys[2] and Jehoshua Bruck[1]

[1]Department of Electrical Engineering, California Institute of Technology
[2]Department of Electrical and Computer Engineering, University of California San Diego

***Abstract*—Systematic deletion correcting codes play an important role in applications of document exchange. Yet despite a series of recent advances made in deletion correcting codes, most of them are non-systematic. To the best of the authors' knowledge, the only known deterministic systematic t-deletion correcting code constructions with rate approaching 1 achieve $O(t\log^2 n)$ bits of redundancy for constant t, where n is the code length. In this paper, we propose a systematic t-deletion correcting code construction that achieves $4t\log n + o(\log n)$ bits of redundancy, which is asymptotically within a factor of 4 from being optimal. Our encoding and decoding algorithms have complexity $O(n^{2t+1})$, which is polynomial for constant t.**

I. Introduction

In the work [14], Levenshtein introduced the problem of constructing optimal t-deletion correcting codes, defined as a set of length n binary sequences no two of which share a common length $n-t$ subsequence. He proved that the optimal redundancy is within the range $t\log n + o(\log n)$ to $2t\log n + o(\log n)$, where log is base-2 throughout the paper. In addition, he showed that the Varshamov-Tennengolts (VT) construction [20]

$$\left\{(c_1,\ldots,c_N) : \sum_{i=1}^{N} ic_i \equiv 0 \bmod (N+1)\right\}, \tag{1}$$

forms an optimal single-deletion correcting code. Though several generalizations [1], [11], [16] of the VT construction were proposed to correct multiple deletions, t-deletion correcting codes with rate 1 were not constructed, until a recent breakthrough [4] that achieves $O(t^2\log t\log n)$ redundancy. Following [4], a series of advances were made. The work of [8] and [19] constructed two-deletion correcting codes that require $8\log n + o(\log n)$ and $7\log n + o(\log n)$ bits of redundancy, respectively. In [6], a t-deletion correcting code with $O(t\log n)$ bits of redundancy was proposed. A t-deletion correcting code with $8t\log n + o(\log n)$ bits of redundancy was given in [17].

The problem of constructing t-deletion correcting codes has a wide range of applications. One of them is document exchange [7], which is the focus of this paper. In the document exchange setting, two parties Alice and Bob keep their own files, represented by sequences X and Y, respectively. The edit distance between X and Y, measured by the minimum number of deletions, insertions, or substitutions needed to change X into Y, is upper bounded by t, where t is a constant in this paper. Alice wants to send the sequence X to Bob by transmitting a hash of X. The goal is to minimize the size of the hash, such that Bob can use it and the sequence Y to recover X. The problem of document exchange is highly related to that of constructing systematic deletion correcting codes, since a document exchange scheme implies a systematic deletion correcting code and vice versa.

Despite the progress in multiple deletion correcting codes, none of the constructions above are systematic and suitable for document exchange settings. For systematic deletion correcting codes and document exchange schemes, when randomized setting is considered, i.e., fading error probability is allowed, document exchange algorithms with hash sizes $O(t\log^2 n)$ and $O(t\log^2 n\log^* n)$[1] were presented in [12] and [13], respectively. The results for randomized settings were improved to $O(t^2\log n)$ in [5] and to $O(t\log n)$ in [3]. In [10], a randomized systematic t-deletion correcting code with $O(t^2\log n)$ bits of redundancy was presented. For deterministic settings, the state of the art document exchange schemes [2], [6], [9] have hash size $O(t\log^2 n)$, which is bounded away from the lower bound $t\log n + o(\log n)$.

The main contribution of this paper is to provide a systematic t-deletion correcting code with $4t\log n + o(\log n)$ bits of redundancy, which is asymptotically within a factor of 4 from being optimal. The encoding/decoding complexity of our construction is $O(n^{2t+1})$. We note that while in this paper we focus on deletion errors, our codes are capable of correcting r deletions, o insertions, and s substitutions for any r, o, and s satisfying $r+o+s \leqslant t$. Hence, it implies a document exchange scheme with hash size $4t\log n + o(\log n)$.

This work was supported in part by NSF grants CCF-1816965 and CCF-1717884.

[1]$\log^* n$ is the minimum number of times the logarithm needs to be iteratively applied before getting n to a result at most 1.

Theorem 1. *For any sequence $\mathbf{c} \in \{0,1\}^n$, there exists a hash function $Hash_t : \{0,1\}^n \to \{0,1\}^{4t\log n + o(\log n)}$, computable in $O(n^{2t+1})$ time, such that $\{(\mathbf{c}, Hash_t(\mathbf{c})) : \mathbf{c} \in \{0,1\}^n\}$ forms a t-deletion correcting code. The decoding complexity of the code $\{(\mathbf{c}, Hash_t(\mathbf{c})) : \mathbf{c} \in \{0,1\}^n\}$ is $O(n^{t+1})$.*

The key ideas behind our construction generalize and apply the techniques in our previous works. They are sketched as follows: (i) generalizing the VT-construction to correct deletions and substitutions for constrained sequences, (ii) decomposing a sequence into multiple versions of it with different resolution levels such that the version with the lowest resolution is a constrained sequence, (iii) compressing the hash by using modulo operations.

In [17] we generalized the VT construction and proved that the higher order parities $\sum_{i=1}^n c_i i^e \bmod kn^e$, $e \in [0,2t] \triangleq \{0,1,\ldots,2t\}$ is a t-deletion correcting hash for sequences $\mathbf{c}$, in which any two 1 entries are separated by at least $t-1$ 0 entries. Motivated by this observation, we define the $\mathbf{u}$-indicator vector $\mathbb{1}_{\mathbf{u}}(\mathbf{c}) \in \{0,1\}^n$ of $\mathbf{c}$ element-wise for binary sequences $\mathbf{c}$ and $\mathbf{u}$. Let n and $\ell \leqslant n$ be the length of $\mathbf{c}$ and $\mathbf{u}$ respectively, define $\mathbb{1}_{\mathbf{u}}(\mathbf{c})$ by

$$\mathbb{1}_{\mathbf{u}}(\mathbf{c})_i = \begin{cases} 1, & \text{if } i \leqslant n-\ell+1 \text{ and } (c_i,\ldots,c_{i+\ell-1}) = \mathbf{u}, \\ 0, & \text{else.} \end{cases}$$

The special cases of $\mathbb{1}_{\mathbf{u}}(\mathbf{c})$ when $\mathbf{u} = (0,1)$ or $\mathbf{u} = (1,0)$ were considered in [19], where a sequence $\mathbf{c}$ is decomposed into a $(1,0)$-indicator vector and a $(0,1)$-indicator vector for $t = 2$. In this paper we generalize this decomposition for $t > 2$. We iteratively generate t versions of $\mathbf{c}$ with different levels of resolution. Let $I_1(\mathbf{c}) = \mathbf{c}$ and

$$I_{w+1}(\mathbf{c}) = \mathbb{1}_{(1,\mathbf{0}^w)}(I_w(\mathbf{c})),$$

for $w \in [t-1] \triangleq \{1,\ldots,t-1\}$, where $(1,\mathbf{0}^w)$ is a sequence of a 1 entry followed by w 0 entries. The 1 entries of $I_{w+1}(\mathbf{c})$ are also 1 entries of $I_w(\mathbf{c})$, $w \in [t-1]$.

Example 1. *For $t = 3$ and $\mathbf{c} = 10010110100$, we have that $I_1(\mathbf{c}) = \mathbb{1}_1(\mathbf{c}) = 10010110100$, $I_2(\mathbf{c}) = \mathbb{1}_{(1,0)}(I_1(\mathbf{c})) = 10010010100$, and that $I_3(\mathbf{c}) = \mathbb{1}_{(1,0,0)}(I_2(\mathbf{c})) = 10010000100$.*

The nice properties of $I_w(\mathbf{c})$ are as follows. (1) Any two 1 entries in $I_t(\mathbf{c})$ are separated by at least $t-1$ 0 entries. (2) The vector $I_w(\mathbf{c})$ is highly constrained when $I_{w+1}(\mathbf{c})$ is known, as will be discussed in Section IV. The first property guarantees that $I_t(\mathbf{c})$ can be protected using higher order parities. The second property enables a successive decoding algorithm.

Next we define the generalized VT constraint. Define the integer vectors

$$\mathbf{m}^{(e)} \triangleq (1^e, 1^e + 2^e, \ldots, \sum_{j=1}^n j^e),$$

for $e \in [0, 2t^2]$. For any sequence $\mathbf{c} \in \{0,1\}^n$, let $f(\mathbf{c})$ be a $2t^2+1$-dimensional vector given by

$$f(\mathbf{c})_e = \mathbf{c} \cdot \mathbf{m}^{(e)} \bmod t^2 n^{e+1},$$

for $e \in [0, 2t^2]$.

The rest of the paper is organized as follows. Section II describes the key ingredients of the construction and defines the notations. In Section III we prove that $f(I_w(\mathbf{c}))$ is a t-deletion correcting hash for the vector $I_t(\mathbf{c})$. Section IV shows how to correct $I_w(\mathbf{c})$, when $I_{w+1}(\mathbf{c})$ is known. Section V presents the encoding/decoding algorithms and proves Theorem 1. Finally, Section VI concludes the paper.

II. Preliminaries

In this section we outline the key ingredients that are needed in our construction, as well as presenting notations that will be used throughout the paper. For any sequence $\mathbf{c} \in \{0,1\}^n$, define its deletion ball $\mathcal{D}_t(\mathbf{c})$ as the set of sequences that can be obtained after deleting t bits in $\mathbf{c}$. For a non-negative integer i, let $\mathcal{B}_{t,i}(\mathbf{c})$ be the set of sequences that can be obtained after deleting t bits and substituting i bits in $\mathbf{c}$. For any integer m, let $\mathcal{R}_{t,m}$ be the set of length m sequences any two 1 entries in which are separated by at least $t-1$ 0 entries. The following lemma gives an upper bound on the number of deletions and substitutions in $I_w(\mathbf{c})$, caused by t deletions in $\mathbf{c}$.

Lemma 1. *For sequences $\mathbf{c}, \mathbf{c}' \in \{0,1\}^n$, if there exists a subsequence $\mathbf{d} \in \mathcal{D}_t(\mathbf{c}) \cap \mathcal{D}_t(\mathbf{c}')$, then $I_w(\mathbf{d}) \in \mathcal{B}_{t,t(w-1)}(I_w(\mathbf{c})) \cap \mathcal{B}_{t,t(w-1)}(I_w(\mathbf{c}'))$ for $w \in [t]$.*

Proof. We show that a deletion in $\mathbf{c}$ causes at most a deletion and $w-1$ substitutions in $I_w(\mathbf{c})$. To this end, we prove in the following that the deletion of c_i causes a deletion in $I_w(\mathbf{c})$ and multiple substitutions that occur within interval $[i - w(w-1)/2, i-1]$ in $I_w(\mathbf{c})$. Since for any sequence $\mathbf{c} \in \{0,1\}^m$, we have $I_w(\mathbf{c}) \in \mathcal{R}_{w,m}$ and $I_w(\mathbf{c}) \in \mathcal{R}_{w,m-1}$ before and after deleting c_i in $\mathbf{c}$, respectively, it follows that there are at most $(w-1)/2$ substitution errors that change 1 entries to 0 entries and 0 entries to 1 entries respectively. Hence, a deletion in $\mathbf{c}$ causes a deletion and at most $w-1$ substitutions in $I_w(\mathbf{c})$. Since $\mathbf{d} \in \mathcal{D}_t(\mathbf{c})$, it follows that $I_w(\mathbf{d}) \in \mathcal{B}_{t,t(w-1)}(\mathbf{c})$. Similarly, we have that $I_w(\mathbf{d}) \in \mathcal{B}_{t,t(w-1)}(\mathbf{c}')$.

We show how a deletion or substitution that occurs at $I_j(\mathbf{c})_{i_j}$, $i_j \in [n]$, affects $I_{j+1}(\mathbf{c})$, $j \in [1, w-1]$. Let $i_j^* = \max_{\ell < i_j, I_j(\mathbf{c})_\ell = 1} \ell$ be the index of the last 1 entry in $I_j(\mathbf{c})$ before $I_j(\mathbf{c})_{i_j}$, and $i_j^{**} = \min_{\ell > i_j, I_j(\mathbf{c})_\ell = 1} \ell$ be the index of the first 1 entry in $I_j(\mathbf{c})$ after $I_j(\mathbf{c})_{i_j}$, where it is assumed that $I_j(\mathbf{c})_\ell = 1$ when $\ell = 0$ or $\ell = n+1$.

1) If $i_j^* \leqslant i_j - j - 1$, then the deletion or substitution of $I_j(\mathbf{c})_{i_j}$ in $I_j(\mathbf{c})$ causes the deletion or substitution of $I_{j+1}(\mathbf{c})_{i_j}$ in $I_{j+1}(\mathbf{c})$ respectively.
2) If $i_j^* \geqslant i_j - j$ and $I_j(\mathbf{c})_{i_j} = 1$, then the deletion or substitution of $I_j(\mathbf{c})_{i_j}$ in $I_j(\mathbf{c})$ causes a deletion

or at most a substitution of $I_{j+1}(\mathbf{c})_{i_j}$ in $I_{j+1}(\mathbf{c})$, respectively.

3) If $i_j^* \geqslant i_j - j$ and $I_j(\mathbf{c})_{i_j} = 0$, then the substitution of $I_j(\mathbf{c})_{i_j}$ in $I_j(\mathbf{c})$ causes at most two substitutions of $I_{j+1}(\mathbf{c})_{i_j^*}$ and $I_{j+1}(\mathbf{c})_{i_j}$ in $I_{j+1}(\mathbf{c})$. The deletion of $I_j(\mathbf{c})_{i_j}$ in $I_j(\mathbf{c})$ causes the deletion of $I_{j+1}(\mathbf{c})_{i_j}$ and the substitution of $I_{j+1}(\mathbf{c})_{i_j^*}$ in $I_{j+1}(\mathbf{c})$, when $i_j^{**} - i_j^* \leqslant j+1$, and causes only the deletion of $I_{j+1}(\mathbf{c})_{i_j}$ in $I_{j+1}(\mathbf{c})$ when $i_j^{**} - i_j^* \geqslant j+2$.

In all cases above, a deletion of $I_j(\mathbf{c})_{i_j}$ in $I_j(\mathbf{c})$ causes a deletion and at most one substitution that occurs in the range $[i_j - j, i_j] = \{i_j - j, i_j - j + 1, \ldots, i_j\}$, and a substitution of $I_j(\mathbf{c})_{i_j}$ in $I_j(\mathbf{c})$ causes at most two substitutions that occur in the range $[i_j - j, i_j]$, for $j \in [w-1]$. Using induction on j we can prove that the deletion of $c_i = I_1(\mathbf{c})_i$ causes one deletion and substitutions that occur in the range $[i - (1 + \ldots + w - 1), i - 1] = [i - w(w-1)/2, i-1]$. □

Given the upper bounds of deletions and substitutions in Lemma 1, the following lemma shows that the generalization of VT constraints $f(\mathbf{x})$ can be used to correct these deletions and substitutions for constrained sequences $\mathbf{x} \in \mathcal{R}_{t,n}$. It will be proved in Section III.

Lemma 2. *For any two sequences $\mathbf{x}, \mathbf{x}' \in \{0,1\}^n$, if there exists a sequence $\mathbf{z} \in \mathcal{R}_{t,n-t}$ satisfying $\mathbf{z} \in \mathcal{B}_{t,t(t-1)}(\mathbf{x}) \cap \mathcal{B}_{t,t(t-1)}(\mathbf{x}')$, we have that $f(\mathbf{x}) \neq f(\mathbf{x}')$.*

After protecting $I_t(\mathbf{c})$, the following lemma provides a hash function that recovers $I_w(\mathbf{c})$, $w \in [t-1]$, from deletions and substitutions, the numbers of which are upper bounded in Lemma 1, when $I_{w+1}(\mathbf{c})$ is known. It will be proved in Section IV.

Lemma 3. *For any two sequences $\mathbf{c}, \mathbf{c}' \in \{0,1\}^n$, if there exists a sequence $\mathbf{d}$ satisfying $\mathbf{d} \in \mathcal{D}_t(\mathbf{x}) \cap \mathcal{D}_t(\mathbf{c}')$, then there exists a hash function $H_w : \{0,1\}^n \to \{0,1\}^{2tw \log n}$, such that given $\mathbf{d}$, $I_{w+1}(\mathbf{c})$, and $H_w(\mathbf{c})$, we can recover that $I_w(\mathbf{c})$, for $w \in [t-1]$.*

The t-deletion correcting hashes described in Lemma 2 and Lemma 3 have sizes larger than $O(t \log n)$. To compress the hash functions, we apply the *syndrome compression* technique, which is presented in our work [18].

Lemma 4. *[c.f. [18]] Let $g : \{0,1\}^n \to [2^{o((\log\log n \cdot \log n))}]$ be a labeling function such that for any fixed $\mathbf{x} \in \{0,1\}^n$ and any $\mathbf{y}$ satisfying $\mathbf{y} \neq \mathbf{x}$ and $\mathcal{D}_t(\mathbf{x}) \cap \mathcal{D}_t(\mathbf{y}) \neq \varnothing$, we have that $g(\mathbf{x}) \neq g(\mathbf{y})$. Then there exists an integer $\alpha \leqslant 2^{\log|\mathcal{D}_t(\mathbf{x})| + o(\log m)}$ such that for any $\mathbf{y}$ satisfying $\mathbf{y} \neq \mathbf{x}$ and $\mathcal{D}_t(\mathbf{x}) \cap \mathcal{D}_t(\mathbf{y}) \neq \varnothing$, we have $g(\mathbf{x}) \not\equiv g(\mathbf{y}) \bmod \alpha$.*

Throughout the paper, we interchangeably represent a binary vector as an integer and vice versa.

III. Correcting $I_t(\mathbf{c})$

In this section we prove Lemma 2. Note that by Lemma 1 we have that $I_t(\mathbf{z}) \in \mathcal{R}_{t,n-t}$ and that $I_t(\mathbf{z}) \in \mathcal{B}_{t,t(t-1)}(I_t(\mathbf{x})) \cap \mathcal{B}_{t,t(t-1)}(I_t(\mathbf{x}'))$. By virtue of this lemma, the redundancy $f(I_t(\mathbf{c}))$ can be used to correct $I_t(\mathbf{c})$. The proof of Lemma 2 follows similar arguments to those in [17], which show that any sequence in $\mathcal{R}_{t,n}$ can be protected from t deletions by using higher oder parities. Here slight changes are made in order to deal with additional substitutions.

Let $\boldsymbol{\delta} = \{\delta_1, \ldots, \delta_t\} \subset [n]$ be a set of deletion indices and $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_{t(t-1)}\} \subset [n]$ be a set of substitution indices, such that deleting bits $(x_i : i \in \boldsymbol{\delta})$ and substituting bits $(x_i : i \in \boldsymbol{\sigma})$ in $\mathbf{x}$ result in $\mathbf{z}$. Similarly, let $\boldsymbol{\delta}' = \{\delta_1', \ldots, \delta_t'\} \subset [n]$ and $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_{t(t-1)}\} \subset [n]$ be two sets such that deleting bits $(x_i' : i \in \boldsymbol{\delta}')$ and substituting bits $(x_i' : i \in \boldsymbol{\sigma}')$ in $\mathbf{x}'$ result in $\mathbf{z}$. Let $\mathbf{y}$ and $\mathbf{y}'$ be the sequences obtained by substituting bits $(x_i : i \in \boldsymbol{\sigma})$ in $\mathbf{x}$ and substituting bits $(x_i' : i \in \boldsymbol{\sigma}')$ in $\mathbf{x}'$, respectively. Then we have that $\mathbf{z} \in \mathcal{D}_t(\mathbf{y}) \cap \mathcal{D}_t(\mathbf{y}')$. Moreover, the sequence $\mathbf{z}$ can be obtained by deleting $(y_i : i \in \boldsymbol{\delta})$ from $\mathbf{y}$ or deleting $(y_i' : i \in \boldsymbol{\delta}')$ from $\mathbf{y}'$.

We now compute the difference $\mathbf{x} \cdot \mathbf{m}^{(e)} - \mathbf{x}' \cdot \mathbf{m}^{(e)}$ and show that it cannot be 0 for all $e \in [0, 2t^2]$ unless $\mathbf{x} = \mathbf{x}'$. Following the same steps as in [17], let $\boldsymbol{\Delta} = \{i : y_i = 1\}$ and $\boldsymbol{\Delta}' = \{i : y_i' = 1\}$ be indices of 1 entries in $\mathbf{y}$ and $\mathbf{y}'$ respectively. Let $S_1 = \boldsymbol{\Delta} \cap \boldsymbol{\delta}$ and $S_2 = \boldsymbol{\Delta}' \cap \boldsymbol{\delta}'$be indices of the 1 entries, after deleting which in $\mathbf{y}$ and $\mathbf{y}'$, respectively, we have $\mathbf{z}$. Then, the sets $S_1^c = [n] \backslash S_1$ and $S_2^c = [n] \backslash S_2$ are indices of the 1 entries that are not deleted in $\mathbf{y}$ and $\mathbf{y}'$ respectively. We have that

$$\begin{aligned}
\mathbf{y} \cdot \mathbf{m}^{(e)} - \mathbf{y}' \cdot \mathbf{m}^{(e)} &= \sum_{\ell \in \boldsymbol{\Delta}} \Big(\sum_{i=1}^{\ell} i^e\Big) - \sum_{\ell \in \boldsymbol{\Delta}'} \Big(\sum_{i=1}^{\ell} i^e\Big) \\
&= \sum_{i=1}^{n} (|S_1 \cap [i,n]| + |S_1^c \cap [i,n]| \\
&\quad - |S_2 \cap [i,n]| - |S_2^c \cap [i,n]|) i^e \qquad (2)
\end{aligned}$$

Sort all elements in sets $\boldsymbol{\delta}, \boldsymbol{\delta}', \boldsymbol{\sigma}$, and $\boldsymbol{\sigma}'$ by $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_{2t^2}$. Let $p_0 = 0$ and $p_{2t^2+1} = n$. The sets S_1 and S_2 satisfy the following properties, the proof of which follows the same steps as in [17].

1) $-1 \leqslant |S_1^c \cap [i,n]| - |S_2^c \cap [i,n]| \leqslant 1$ for $i \in [n]$.
2) For each interval $(p_j, p_{j+1}] \triangleq \{p_j + 1, \ldots, p_{j+1}\}$, $j \in [0, 2t^2]$, we have either $|S_1^c \cap [i,n]| - |S_2^c \cap [i,n]| \leqslant 0$ for all $i \in (p_j, p_{j+1}]$ or $|S_1^c \cap [i,n]| - |S_2^c \cap [i,n]| \geqslant 0$ for all $i \in (p_j, p_{j+1}]$.

Let the sets $S_3 = \Delta \cap \boldsymbol{\sigma} = \{i : y_i = 1, i \in \boldsymbol{\sigma}\}$ and $S_4 = \Delta' \cap \boldsymbol{\sigma}' = \{i : y_i' = 1, i \in \boldsymbol{\sigma}'\}$ be the indices of substitutions that flip 0 bits in $\mathbf{x}$ and $\mathbf{x}'$, in order to get $\mathbf{y}$ and $\mathbf{y}'$ respectively. Let $S_5 = \boldsymbol{\sigma} \backslash S_3 = \{i : y_i = 0, i \in \boldsymbol{\sigma}\}$, and $S_6 = \boldsymbol{\sigma} \backslash S_4 = \{i : y_i = 0, i \in \boldsymbol{\sigma}'\}$ be the indices of substitutions that flip 1 bits in $\mathbf{x}$ and $\mathbf{x}'$, to get $\mathbf{y}$ and $\mathbf{y}'$, respectively. Then,

$$\mathbf{x} \cdot \mathbf{m}^{(e)} - \mathbf{x}' \cdot \mathbf{m}^{(e)}$$

$$= \mathbf{y}\cdot\mathbf{m}^{(e)} - \mathbf{y}'\cdot\mathbf{m}^{(e)} + \sum_{\ell\in S_5}\mathbf{m}_\ell^{(e)} - \sum_{\ell\in S_6}\mathbf{m}_\ell^{(e)}$$
$$-(\sum_{\ell\in S_3}\mathbf{m}_\ell^{(e)} - \sum_{\ell\in S_4}\mathbf{m}_\ell^{(e)})$$
$$= \sum_{j=0}^{2t^2}\sum_{i=p_j+1}^{p_{j+1}} (|S_1^c \cap [i,n]| - |S_2^c\cap[i,n]| + k_i)i^e, \quad (3)$$

where $k_i = |S_1\cap[i,n]| - |S_2\cap[i,n]| + |S_5\cap[i,n]| - |S_3\cap[i,n]| + |S_4\cap[i,n]| - |S_6\cap[i,n]|$ for $i\in[n]$. Note that for any interval $(p_j,p_{j+1}]$, $j\in[0,2t^2]$, the number k_i is constant for all $i\in(p_j,p_{j+1}]$. Let $s_i = |S_1^c\cap[i,n]| - |S_2^c\cap[i,n]| + k_i$, then it follows from Property (1) and Property (2) that s_i is either negative or non-negative for all $i\in(p_j,p_{j+1}]$ for each $j\in[0,2t^2]$.

Next, we show that $\mathbf{x}\cdot\mathbf{m}^{(e)} - \mathbf{x}'\cdot\mathbf{m}^{(e)}$ cannot be zero for all $e\in[0,2t^2]$ when $\mathbf{x}\neq\mathbf{x}'$. Otherwise, define a vector $\mathbf{v} = (v_0,\ldots,v_{2t^2})\in\{-1,1\}^{2t^2+1}$ by

$$v_j = \begin{cases} -1, & \text{if } s_i < 0 \text{ for some } i\in(p_j,p_{j+1}] \\ 1, & \text{else.}\end{cases}.$$

and a $(2t^2+1)\times(2t^2+1)$ matrix A by $A_{e,j} = \sum_{i=p_j+1}^{p_j}|s_i|i^e$ for $e,j\in[0,2t^2]$. Then according to Eq. (3), we have that $\sum_{j=0}^{2t^2}A_{e,j}v_j = 0$, $e\in[0,2t^2]$, if $\mathbf{x}\cdot\mathbf{m}^{(e)} - \mathbf{x}'\cdot\mathbf{m}^{(e)} = 0$ for all $e\in[0,2t^2]$. This implies the linear equation $A\mathbf{v}=0$ has a solution $\mathbf{v}$ with no 0 entry. The remaining steps are the same as in [17]. Let $j_1,\ldots,j_Q$ be the indices of non-zero columns of A, and B be the submatrix of A by selecting the intersection of the first Q rows and the non-zero columns of A. Then the linear equation $B(v_{j_1},\ldots,v_{j_Q}) = 0$ has a non-zero solution, which is impossible since by multi-linearity of the determinant, we can prove that the determinant $|B|>0$. Therefore, $\mathbf{x}\cdot\mathbf{m}^{(e)} - \mathbf{x}'\cdot\mathbf{m}^{(e)} = 0$ for $e\in[0,2t^2]$ only when A is a zero matrix, which implies that

$$s_i = |S_1^c\cap[i,n]| - |S_2^c\cap[i,n]| + k_i$$
$$= |\{i : x_i = 1\}\cap[i,n]| - |\{i : x_i' = 1\}\cap[i,n]| = 0,$$

for $i\in[n]$. Then, $\{i : x_i = 1\} = \{i : x_i' = 1\}$ and thus $\mathbf{x} = \mathbf{x}'$.

Finally, we show that if $f(\mathbf{c}) = f(\mathbf{c}')$, then $\mathbf{x}\cdot\mathbf{m}^{(e)} - \mathbf{x}'\cdot\mathbf{m}^{(e)} = 0$ for $e\in[0,2t^2]$. Since $\mathbf{z}\in\mathcal{B}_{t,t(t-1)}(\mathbf{x})\cap\mathcal{B}_{t,t(t-1)}(\mathbf{x}')$, it follows that $(z_i,\ldots,z_{n-t})\in\mathcal{B}_{t,t(t-1)}((x_i,\ldots,x_n))\cap\mathcal{B}_{t,t(t-1)}((x_i',\ldots,x_n'))$ for $i\in[n-t]$. Hence, we have that $-t^2\leqslant|\{i : x_i=1\}\cap[i,n]| - |\{i : x_i'=1\}\cap[i,n]|\leqslant t^2$, and that

$$|\mathbf{x}\cdot\mathbf{m}^{(e)} - \mathbf{x}'\cdot\mathbf{m}^{(e)}| < t^2n^{e+1}.$$

Therefore, if $f(\mathbf{c}) = f(\mathbf{c}')$, we have that $\mathbf{x}\cdot\mathbf{m}^{(e)} - \mathbf{x}'\cdot\mathbf{m}^{(e)} \equiv 0 \bmod t^2n^{n+1}$, which implies that $\mathbf{x}\cdot\mathbf{m}^{(e)} - \mathbf{x}'\cdot\mathbf{m}^{(e)} = 0$, for $e\in[0,2t^2]$.

IV. Correcting $I_w(\mathbf{c})$ given $I_{w+1}(\mathbf{c})$

In this section we prove Lemma 3. The idea is to notice that given $I_{w+1}(\mathbf{c})$, the sequence $I_w(\mathbf{c})$ can be determined by the first 1 entry in $I_w(\mathbf{c})$ after each 1 entry in $I_{w+1}(\mathbf{c})$, $w\in[t-1]$. Specifically, let

$$(\pi_1^{w+1},\pi_2^{w+1},\ldots,\pi_{n'}^{w+1})$$

be the indices of the 1 entries in $I_{w+1}(\mathbf{c})$ such that $\pi_1^{w+1} < \pi_2^{w+1} < \ldots < \pi_{n'}^{w+1}$. Let

$$\tau_i^w = \min\{j : j > \pi_i^{w+1}, I_w(\mathbf{c})_j = 1 \text{ or } j = n+1\}$$

for $i\in[0,n']$, where $\pi_i^{w+1} = 0$ when $i=0$. We have the following proposition.

Proposition 1. *The sequence $I_w(\mathbf{c})$ can be determined by $(\pi_1^{w+1},\pi_2^{w+1},\ldots,\pi_{n'}^{w+1})$ and $(\tau_0^w,\tau_1^w,\ldots,\tau_{n'}^w)$, for $w\in[t-1]$.*

Proof. Note that the 1 entries of $I_w(\mathbf{c})$ in the interval $(\pi_i^{w+1},\pi_{i+1}^{w+1}]$ are spaced evenly with distance w in the interval $[\tau_i^w,\pi_{i+1}^{w+1}]$, for $i\in[0,n']$, where $\pi_{i+1}^{w+1} = n+1$ if $i = n'$. Otherwise there is an additional one entry in $I_{w+1}(\mathbf{c})$ in the interval $(\pi_i^{w+1},\pi_{i+1}^{w+1}) \triangleq \{\pi_i^{w+1}+1,\ldots,\pi_{i+1}^{w+1}-1\}$, which contradicts to the definition of $(\pi_1^{w+1},\ldots,\pi_{n'}^{w+1})$. □

From Proposition 1, it suffices to protect the indices $(\tau_0^w,\ldots,\tau_{n'}^w)$, in order to recover $I_w(\mathbf{c})$. For $w\in[t-1]$, let

$$H_w(\mathbf{c}) = RS_{2tw}((\tau_0^w - \pi_0^{w+1},\ldots,\tau_{n'}^w - \pi_{n'}^{w+1})),$$

where $RS_{2tw}((\tau_0^w - \pi_0^{w+1},\ldots,\tau_{n'}^w - \pi_{n'}^{w+1}))$ is the redundancy of the Reed-Solomon code that corrects $2tw$ substitution errors in the sequence $(\tau_0^w - \pi_0^{w+1},\ldots,\tau_{n'}^w - \pi_{n'}^{w+1})$, with entries $\tau_i^w - \pi_i^{w+1}$, $i\in[0,n']$. The size of $H_w(\mathbf{c})$ is at most $4tw\log n$ bits.

In the following we present the decoding procedure that recovers $I_w(\mathbf{c})$, given $\mathbf{d}\in\mathcal{D}_t(\mathbf{c})$, $I_{w+1}(\mathbf{c})$, and $H_w(\mathbf{c})$, for any $w\in[t-1]$.

1) **Initialization:** Let $\mathbf{a}\in[n]^{n'+1}$ be a vector, where n' is known given $I_{w+1}(\mathbf{c})$.
2) **Step 1:** For each $i\in[0,n'-1]$, if there exist two numbers $p_i^{w+1}\in[\pi_i^{w+1}-t,\pi_i^{w+1}]$ and $p_{i+1}^{w+1}\in[\pi_{i+1}^{w+1}-t,\ \pi_{i+1}^{w+1}]$ such that $I_{w+1}(\mathbf{d})_{p_i^{w+1}} = I_{w+1}(\mathbf{d})_{p_{i+1}^{w+1}} = 1$ and $p_{i+1}^{w+1} - p_i^{w+1} = \pi_{i+1}^{w+1} - \pi_i^{w+1}$, let $k_i^w = \min_{j>p_i^{w+1}, I_w(\mathbf{d})_j = 1} j$ be the first 1 entry in $I_w(\mathbf{d})$ after $I_w(\mathbf{d})_{p_i^{w+1}}$, where $\mathbb{1}_w(\mathbf{d})_j = 1$ when $j = n-t+1$. Let $a_i = k_i^w - p_i^w$. Else let $a_i = 0$.
3) **Step 2:** Apply the Reed-Solomon decoder on $\mathbf{a}$ to recover $(\tau_0^w - \pi_0^{w+1},\ldots,\tau_{n'}^w - \pi_{n'}^{w+1})$. Recover $(\tau_0^w,\ldots,\tau_{n'}^w)$, and $I_w(\mathbf{c})$ according to Proposition 1.
4) **Step 3:** Output $I_w(\mathbf{c})$.

We now show that the above procedure decodes $I_w(\mathbf{c})$ correctly, $w\in[t-1]$. According to Lemma 1, The sequence $I_w(\mathbf{d})$ can be obtained from $I_w(\mathbf{c})$ after t deletions and at most $t(w-1)$ substitutions. Note that for each $i \in [0, n']$, we have that $a_i = \tau_i^w - \pi_i^{w+1}$, if no deletion or substitution occurs in the interval $[\pi_i^{w+1}, \pi_{i+1}^{w+1}]$ in $I_w(\mathbf{c})$, where $\pi_{i+1}^{w+1} = n$ if $i = n'$. Since a deletion or a substitution occurs in at most two adjacent intervals $[\pi_i^{w+1}, \pi_{i+1}^{w+1}]$ and $[\pi_{i+1}^{w+1}, \pi_{i+2}^{w+1}]$, t deletions and $t(w-1)$ substitutions cause at most $2tw$ symbol errors $a_i \neq \tau_i^w - \pi_i^{w+1}$ in $\mathbf{a}$. Hence the sequence $(\tau_0^w - \pi_0^{w+1}, \ldots, \tau_{n'}^w - \pi_{n'}^{w+1})$, and thus $(\tau_0^w, \ldots, \tau_{n'}^w)$ can be recovered given $H_w(\mathbf{c})$. Finally, according to Proposition 1, the sequence $I_w(\mathbf{c})$ can be recovered, $w \in [t-1]$.

The complexities for computing $H_w(\mathbf{c})$ and decoding $I_w(\mathbf{c})$ are dominated by encoding and decoding the Reed-Solomon code and are polynomial.

V. Encoding/Decoding

In this section we describe the encoding and decoding procedures and prove Theorem 1. For any $\mathbf{c} \in \{0,1\}^n$, define the function

$$g(\mathbf{c}) = (f(I_t(\mathbf{c})), H_1(\mathbf{c}), H_2(\mathbf{c}), \ldots, H_{t-1}(\mathbf{c})).$$

We first show that $g(\mathbf{c})$ is a t-deletion correcting labeling for $\mathbf{c}$. For any length $n-t$ subsequence $\mathbf{d}$ of $\mathbf{c}$, according to Lemma 1, we have that $I_w(\mathbf{d}) \in \mathcal{B}_{t,t(w-1)}(I_w(\mathbf{c}))$ for $w \in [t]$. In particular, we have that $I_t(\mathbf{d}) \in \mathcal{B}_{t,t(t-1)}(I_t(\mathbf{c}))$. Since $I_t(\mathbf{d}) \in \mathcal{R}_{t,n-t}$, it follows from Lemma 2 that $f(I_t(\mathbf{c}) \neq f(I_t(\mathbf{c}'))$ for any $\mathbf{c}'$ satisfying $I_t(\mathbf{d}) \in \mathcal{B}_{t,t(t-1)}(I_w(\mathbf{c}'))$. Hence, the sequence $I_t(\mathbf{c})$ can be recovered, given $f(I_t(\mathbf{c}))$ and $\mathbf{d}$. According to Lemma 3, every sequence $I_w(\mathbf{c})$ can be recovered using $H_w(\mathbf{c})$, $I_{w+1}(\mathbf{c})$, and $\mathbf{d}$. Hence, after knowing $I_t(\mathbf{c})$, the sequence $\mathbf{c} = I_1(\mathbf{c})$ can be recovered by successively decoding $I_w(\mathbf{c})$, from $w = t-1$ to $w = 1$.

The size of $g(\mathbf{c})$ is $R = [(t^2+1)(2t^2+1) + 2t^2(t-1)]\log n + o(\log n)$, which is greater than $O(t \log n)$. By applying Lemma 4, there exists an integer $\alpha \in [2^{\log|\mathcal{D}_t(\mathbf{c})|+o(\log n)}] = [2^{2t\log n + o(\log n)}]$ such that $g(\mathbf{c}) \not\equiv g(\mathbf{c}') \bmod \alpha$ for any $\mathbf{c}' \in \mathcal{D}_t(\mathbf{c})$. Let

$$g_c(\mathbf{c}) = (g(\mathbf{c}), \alpha).$$

Then $g_c(\mathbf{c})$ is a t-deletion correcting hash for $\mathbf{c}$ of size $N_1 = 4t\log n + o(\log n)$. Let

$$Hash_t(\mathbf{c}) = (g_c(\mathbf{c}), Rep_{t+1}(g_c(g_c(\mathbf{c})))),$$

where $Rep_{t+1}(g_c(g_c(\mathbf{c})))$ is the $t+1$ fold repetition of $g_c(g_c(\mathbf{c}))$, of length $N_2 = 4t\log N_1 + o(\log N_1) = 4t(t+1)\log\log n + o(\log n)$. The size of $Hash_t(\mathbf{c})$ is $N_1 + N_2 = 4t\log n + o(\log n)$. We now show that $(\mathbf{c}, Hash_t(\mathbf{c}))$ is a t-deletion correcting code. For any length $n + N_1 + N_2 - t$ subsequence $\mathbf{z}$ of $(\mathbf{c}, Hash_t(\mathbf{c}))$, we have that $(z_{n+N_1+1}, \ldots, z_{n+N_1+N_2-t})$ is a length $N_2 - t$ subsequence of $Rep_{t+1}(g_c(g_c(\mathbf{c}))))$, which is a t-deletion correcting code. Therefore $g_c(g_c(\mathbf{c}))$ can be recovered. In addition, $(z_{n+1}, \ldots, z_{n+N_1-t})$ is a length $N_1 - t$ subsequence of $g_c(\mathbf{c})$. Since $g_c(g_c(\mathbf{c}))$ is a t-deletion correcting hash of $g_c(\mathbf{c})$, the hash $g_c(\mathbf{c})$ can be recovered. Finally, note that $(z_1, \ldots, z_{n-t})$ is a length $n-t$ subsequence of n, we can use $g_c(\mathbf{c})$ to recover $\mathbf{c}$. The decoding of $\mathbf{c}$ from $g_c(\mathbf{c})$ is done using brute force, over all sequences $\mathbf{c}'$ that satisfy $\mathbf{d} \in \mathcal{D}_t(\mathbf{c}')$. The computing of $g_c(\mathbf{c})$ is done by brute force, over sequences $\mathbf{c}' \in \mathcal{D}_t(\mathbf{c})$. Hence the encoding and decoding complexities are $O(n^{2t+1})$ and $O(n^{t+1})$ respectively.

VI. Conclusions and Future Work

Motivated by the applications in document exchange, we construct systematic t-deletion correcting codes that achieve $4t\log n + o(\log n)$ bits of redundancy, which is optimal up to a constant. Our codes are capable of correcting up to t deletion, insertion, and substitution errors, and thus provide a document exchange scheme. Our construction has encoding/decoding complexity polynomial in n but exponential in t, which works when t is a constant. It is intriguing, yet challenging, to come up with encoding/decoding algorithms polynomial in both n and t.

References

[1] K. A. Abdel-Ghaffar, F. Paluncic, H. C. Ferreira and W. A. Clarke, "On Helberg's generalization of the Levenshtein code for multiple deletion/insertion error correction," *IEEE Trans. on Inf. Th.*, vol. 58, no. 3, pp. 1804–1808, 2012.

[2] D. Belazzougui, "Efficient deterministic single round document exchange for edit distance," *arXiv:1511.09229*, 2015.

[3] D. Belazzougui and Q Zhang, "Edit distance: Sketching, streaming, and document exchange," *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 51–60, 2016.

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

[5] D. Chakraborty, E. Goldenberg and M. Koucký, "Low distortion embedding from edit to Hamming distance using coupling," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 22, no. 111, 2015.

[6] K. Cheng, Z. Jin, X. Li and K. Wu, "Deterministic document exchange protocols, and almost optimal binary codes for edit errors," *IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 200–211, 2018.

[7] G. Cormode, M. Paterson, S.C. Sahinalp and U. Vishkin "Communication complexity of document exchange," *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 197–206, 2000.

[8] R. Gabrys and F. Sala, "Codes correcting two deletions," *IEEE Trans. on Inf. Th.*, vol. 65, no. 2, pp. 965–974, Feb. 2019.

[9] B. Haeupler, "Optimal document exchange and new codes for small number of insertions and deletions," *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 334–347, 2019.

[10] S.K. Hanna and S. El Rouayheb, "Guess & check codes for deletions, insertions, and synchronization," *IEEE Trans. on Inf. Th.*, vol. 65, no. 1, pp. 3–15, Jan. 2019.

[11] A. S. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Trans. on Inf. Th.*, vol. 48, no. 1, pp. 305–308, 2002.

[12] U. Irmak, S. Mihaylov and T. Suel, "Improved single-round protocols for remote file synchronization," *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3, pp. 1665–1676, 2005.

[13] H. Jowhari, “Efficient communication protocols for deciding edit distance,” *European Symposium on Algorithms*, pp. 648–658, 2012.

[14] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet physics doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[15] M. Mitzenmacher, “A survey of results for deletion channels and related synchronization channels,” *Probability Surveys*, vol. 6, pp. 1–33, 2009.

[16] F. Paluncic, K. A. Abdel-Ghaffar, H. C. Ferreira and W. A. Clarke, “A multiple insertion/deletion correcting code for run-length limited sequences,” *IEEE Trans. on Inf. Th.*, vol. 58, no. 3, pp. 1809–1824, 2012.

[17] J. Sima and J. Bruck, “Optimal k-deletion correcting codes,” *arXiv:1910.12247*, 2019.

[18] J. Sima, R. Gabrys and J. Bruck, “Syndrome compression for optimal redundancy codes,” *proc. ISIT*, 2020.

[19] J. Sima, N. Raviv, and J. Bruck, “Two deletion correcting codes from indicator vectors,” in *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2375-2391, 2020.

[20] R. R. Varshamov and G. M. Tenengolts, “Codes which correct single asymmetric errors,” *Autom. Remote Control*, vol. 26, no. 2, pp. 286–290, 1965.