Guess & Check Codes for Deletions, Insertions, and Synchronization

Serge Kas Hanna, Salim El Rouayheb
ECE Department, IIT, Chicago
skashann@hawk.iit.edu, salim@iit.edu

***Abstract*—We consider the problem of constructing codes that can correct δ deletions occurring in an arbitrary binary string of length n bits. Varshamov-Tenengolts (VT) codes are zero-error single deletion ($\delta = 1$) correcting codes, and have an asymptotically optimal redundancy. Finding similar codes for $\delta \geq 2$ deletions is an open problem. We propose a new family of codes, that we call Guess & Check (GC) codes, that can correct, with high probability, up to δ deletions occurring in a binary string. Moreover, we show that GC codes can also correct up to δ insertions. GC codes are based on MDS codes and have an asymptotically optimal redundancy that is $\Theta(\delta \log n)$. We provide deterministic polynomial time encoding and decoding schemes for these codes. We also describe the applications of GC codes to file synchronization.**

I. Introduction

The deletion channel is probably the most notorious example of a point-to-point channel whose capacity remains unknown. The bits that are deleted by this channel are completely removed from the transmitted sequence and their locations are unknown at the receiver (unlike erasures). For example, if 1010 is transmitted, the receiver would get 00 if the first and third bits were deleted. Constructing efficient codes for the deletion channel has also been a challenging task. Varshamov-Tenengolts (VT) codes [2] are the only deletion codes with asymptotically optimal redundancy and can correct only a single deletion. The study of the deletion channel has many applications such as file synchronization [3]–[7] and DNA-based storage [8].

The capacity of the deletion channel has been studied in the probabilistic model. In the model where the deletions are i.i.d. and occur with a fixed probability p, an immediate upper bound on the channel capacity is given by the capacity of the erasure channel $1-p$. Mitzenmacher and Drinea showed in [9] that the capacity of the deletion channel in the i.i.d. model is at least $(1-p)/9$. Extensive work in the literature has focused on determining lower and upper bounds on the capacity [9]–[14]. We refer interested readers to the comprehensive survey by Mitzenmacher [15]. Ma *et al.* [7] also studied the capacity in the bursty model of the deletion channel, where the deletion process is modeled by a Markov chain.

A separate line of work has focused on constructing codes that can correct a given number of deletions. In this work we are interested in binary codes that correct δ deletions. Levenshtein showed in [16] that VT codes [2] are capable of correcting a single deletion ($\delta = 1$), with an asymptotically optimal redundancy ($\log(n+1)$ bits). More information about the VT codes and other properties of single deletion codes can be found in [17]. VT codes have been used to construct codes that can correct a combination of a single deletion and multiple adjacent transpositions [8]. However, finding VT-like codes for multiple deletions ($\delta \geq 2$) is an open problem. In [16], Levenshtein provided bounds showing that the asymptotic number of redundant bits needed to correct δ bit deletions in an n bit codeword is $\Theta(\delta \log n)$, i.e., $c\ \delta \log n$ for some constant $c > 0$. Levenshtein's bounds were later generalized and improved in [18].

The simplest code for correcting δ deletions is the $(\delta + 1)$ repetition code, where every bit is repeated $(\delta + 1)$ times. However, this code is inefficient because it requires δn redundant bits, i.e., a redundancy that is linear in n. Helberg codes [19,20] are a generalization of VT codes for multiple deletions. These codes can correct multiple deletions but their redundancy is at least linear in n even for two deletions. Schulman and Zuckerman in [21] presented codes that can correct a constant fraction of deletions. Their construction was improved in [22,23], but the redundancies in these constructions are $\mathcal{O}(n)$. Recently in [24], Brakensiek *et al.* provided an explicit encoding and decoding scheme, for fixed δ, that has $\mathcal{O}(\delta^2 \log \delta \log n)$ redundancy and a near-linear complexity. But the crux of the approach in [24] is that the scheme is limited to a specific family of strings, which the authors in [24] refer to as *pattern rich* strings. In summary, even for the case of two deletions, there are no known explicit codes for arbitrary strings, with $\mathcal{O}(\delta \log n)$ redundancy.

Contributions: We consider the problem of constructing codes that can correct multiple deletions with an asymptotically optimal redundancy. This is a problem that has been open for many decades. We relax the standard requirement in the literature (e.g., [19]–[24]) that these codes must satisfy zero-error decoding, and construct codes that have an asymptotically vanishing probability of decoding failure[1], assuming uniform i.i.d. binary messages[2]. Specifically, we make the following contributions: (i) We propose new explicit codes, which we

This is an extended version of the conference paper in [1], which will appear in the proceedings of the IEEE International Symposium on Information Theory, 2017.

This work was supported in parts by NSF Grant CCF 15-26875.

[1]The term *decoding failure* means that the decoder cannot make a correct decision and outputs a "failure to decode" error message.

[2]The other assumption we make is that the positions of the deletions are independent of the information message.

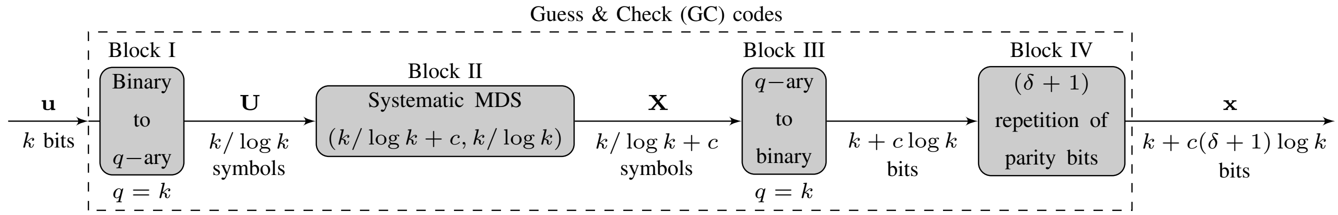

Fig. 1: General encoding block diagram of the GC code for δ deletions. Block I: The binary message of length k bits is chunked into adjacent blocks of length $\log k$ bits each, and each block is mapped to its corresponding symbol in $GF(q)$ where $q = 2^{\log k} = k$. Block II: The resulting string is coded using a systematic $(k/\log k + c, k/\log k)$ $q-$ary MDS code where $c > \delta$ is the number of parity symbols. Block III: The symbols in $GF(q)$ are mapped to their binary representations. Block IV: Only the parity bits are coded using a $(\delta + 1)$ repetition code.

call Guess & Check (GC) codes, that can correct, with high probability, and in polynomial time, up to a constant number of deletions (or insertions) δ occurring in arbitrary binary strings. The encoding and decoding schemes of GC codes are deterministic. Moreover, these codes have an asymptotically optimal redundancy of value $c(\delta + 1)\log k \approx c(\delta + 1)\log n$ (asymptotically), where k and n are the lengths of the message and codeword, respectively, and $c > \delta$ is a constant integer; (ii) GC codes enable different trade-offs between redundancy, decoding complexity, and probability of decoding failure; (iii) We provide numerical simulations on the decoding failure of GC codes and compare these simulations to our theoretical results. For instance, we observe that a GC code with rate 0.8 can correct up to 4 deletions in a message of 1024 bits with no decoding failure detected within 10000 runs of simulations; (iv) We describe how to use GC codes for file synchronization as part of the interactive algorithm proposed by Venkataramanan *et al.* in [3,4] and provide simulation results highlighting the resulting savings in number of rounds and total communication cost.

Organization: The paper is organized as follows. In Section II, we introduce the necessary notations used throughout the paper. We state and discuss the main result of this paper in Section III. In Section IV, we provide encoding and decoding examples on GC codes. In Section V, we describe in detail the encoding and decoding schemes of GC codes. The proof of the main result of this paper is given in Section VI. In Section VII, we explain the trade-offs achieved by GC codes. In Section VIII, we explain how these codes can be used to correct δ insertions instead of δ deletions. The results of the numerical simulations on the decoding failure of GC codes and their applications to file synchronization are shown in Section IX and X, respectively. We conclude with some open problems in Section XI.

II. Notation

Let k and n be the lengths in bits of the message and codeword, respectively. Let δ be the number of deletions. Without loss of generality, we assume that k is a power of 2. Our code is based on a $q-$ary systematic $(\lceil k/\log k\rceil + c, \lceil k/\log k\rceil)$ MDS code, where $q = k > \lceil k/\log k\rceil + c$ and $c > \delta$ is a code parameter representing the number of MDS parity symbols[3].

[3]Explicit constructions of systematic Reed-Solomon codes (based on Cauchy or Vandermonde matrices) always exist for these parameters.

We will drop the ceiling notation for $\lceil k/\log k\rceil$ and simply write $k/\log k$. All logarithms in this paper are of base 2. The block diagram of the encoder is shown in Fig. 1. We denote binary and $q-$ary vectors by lower and upper case bold letters respectively, and random variables by calligraphic letters.

III. Main Result

Let $\mathbf{u}$ be a binary vector of length k with i.i.d. Bernoulli$(1/2)$ components representing the information message. The message $\mathbf{u}$ is encoded into the codeword $\mathbf{x}$ of length n bits using the Guess & Check (GC) code illustrated in Fig. 1. The GC decoder, explained in Section V, can either decode successfully and output the decoded string, or output a "failure to decode" error message because it cannot make a correct decision. The latter case is referred to as a *decoding failure*, and its corresponding event is denoted by F.

Theorem 1. *The Guess & Check (GC) code can correct in polynomial time up to a constant number of δ deletions occurring within $\mathbf{x}$. Let $c > \delta$ be a constant integer. The code has the following properties:*

1) *Redundancy: $n - k = c(\delta + 1)\log k$ bits.*
2) *Encoding complexity is $\mathcal{O}(k\log k)$, and decoding complexity is $\mathcal{O}\left(\frac{k^{\delta+2}}{\log^\delta k}\right)$.*
3) *Probability of decoding failure: $Pr(F) = \mathcal{O}\left(\frac{k^{2\delta-c}}{\log^\delta k}\right)$.*

The probability of decoding failure in Theorem 1 is true for any given δ deletion positions which are independent of $\mathbf{x}$. Hence, the same result can be also obtained for any random distribution over the δ deletion positions (like the uniform distribution for example), by averaging over all the possible δ deletion positions.

GC codes enable trade-offs between the code properties shown in Theorem 1, this will be highlighted later in Section VII. These properties show that: (i) the code rate, $R = k/(k + c(\delta + 1)\log k)$, is asymptotically optimal and approaches one as k goes to infinity; (ii) the order of complexity is polynomial in k and is not affected by the constant c; (iii) the probability of decoding failure goes to zero polynomially in k if $c > 2\delta$; and exponentially in c for a fixed k. Note that the decoder can always detect when it cannot decode successfully. This can serve as an advantage in models which allow feedback. There, the decoder can ask for additional redundancy to be able to decode successfully.

IV. Examples

The GC code we propose can correct up to δ deletions with high probability. We provide examples to illustrate the encoding and decoding schemes. The examples are for $\delta = 1$ deletion just for the sake of simplicity[4].

Example 1 (Encoding)**.** *Consider a binary message* $\mathbf{u}$ *of length* $k = 16$ *given by*

$$\mathbf{u} = 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1.$$

$\mathbf{u}$ *is encoded by following the different encoding blocks illustrated in Fig. 1.*
1) Binary to $q-$ary (Block I, Fig. 1). *The message* $\mathbf{u}$ *is chunked into adjacent blocks of length* $\log k = 4$ *bits each,*

$$\mathbf{u} = \underbrace{\overbrace{1\ 1\ 1\ 0}^{\text{block 1}}}_{\alpha^{11}}\ \underbrace{\overbrace{0\ 0\ 0\ 0}^{\text{block 2}}}_{0}\ \underbrace{\overbrace{1\ 1\ 0\ 1}^{\text{block 3}}}_{\alpha^{13}}\ \underbrace{\overbrace{0\ 0\ 0\ 1}^{\text{block 4}}}_{1}.$$

Each block is then mapped to its corresponding symbol in $GF(q)$*,* $q = k = 2^4 = 16$*, by considering its leftmost bit as its most significant bit. This results in a string* $\mathbf{U}$ *which consists of* $k/\log k = 4$ *symbols in* $GF(16)$*. The extension field used here has a primitive element* α*, with* $\alpha^4 = \alpha + 1$*. Hence,* $\mathbf{U} \in GF(16)^4$ *is given by*

$$\mathbf{U} = (\alpha^{11}, 0, \alpha^{13}, 1).$$

2) Systematic MDS code (Block II, Fig. 1). $\mathbf{U}$ *is then coded using a systematic* $(k/\log k + c, k/\log k) = (6, 4)$ *MDS code over* $GF(16)$*, with* $c = 2 > \delta$*. The encoded string is denoted by* $\mathbf{X} \in GF(16)^6$ *and is given by multiplying* $\mathbf{U}$ *by the following code generator matrix,*

$$\mathbf{X} = (\alpha^{11}, 0, \alpha^{13}, 1)\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & \alpha \\ 0 & 0 & 1 & 0 & 1 & \alpha^2 \\ 0 & 0 & 0 & 1 & 1 & \alpha^3 \end{pmatrix},$$
$$= (\alpha^{11}, 0, \alpha^{13}, 1, \alpha, \alpha^{10}).$$

3) $Q-$ary to binary (Block III, Fig. 1). *The binary codeword corresponding to* $\mathbf{X}$*, of length* $n = k + 2\log k = 24$ *bits, is*

$$\mathbf{x} = \underbrace{1\ 1\ 1\ 0}\ \underbrace{0\ 0\ 0\ 0}\ \underbrace{1\ 1\ 0\ 1}\ \underbrace{0\ 0\ 0\ 1}\ \underbrace{0\ 0\ 1\ 0}\ \underbrace{0\ 1\ 1\ 1}.$$

For simplicity we skip the last encoding step (Block IV) intended to protect the parity bits and assume that deletions affect only the systematic bits.

The high level idea of the decoding algorithm is to: (i) make an assumption on in which block the bit deletion has occurred (the guessing part); (ii) chunk the bits accordingly, treat the affected block as erased, decode the erasure and check whether the obtained sequence is consistent with the parities (the checking part); (iii) go over all the possibilities.

[4] VT codes can correct one deletion with zero-error. However, GC codes are generalizable to multiple deletions.

Example 2 (Successful Decoding)**.** *Suppose that the* 14^{th} *bit of* $\mathbf{x}$ *gets deleted,*

$$\mathbf{x} = 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ \underline{\boldsymbol{0}}\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1.$$

The decoder receives the following 23 *bit string* $\mathbf{y}$*,*

$$\mathbf{y} = 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1.$$

The decoder goes through all the possible $k/\log k = 4$ *cases, where in each case* i*,* $i = 1, \ldots, 4$*, the deletion is assumed to have occurred in block* i *and* $\mathbf{y}$ *is chunked accordingly. Given this assumption, symbol* i *is considered erased and erasure decoding is applied over* $GF(16)$ *to recover this symbol. Furthermore, given two parities, each symbol* i *can be recovered in two different ways. Without loss of generality, we assume that the first parity* p_1*,* $p_1 = \alpha$*, is the parity used for decoding the erasure. The decoded* $q-$*ary string in case* i *is denoted by* $\mathbf{Y_i} \in GF(16)^4$*, and its binary representation is denoted by* $\mathbf{y_i} \in GF(2)^{16}$*. The four cases are shown below:*
Case 1: *The deletion is assumed to have occurred in block 1, so* $\mathbf{y}$ *is chunked as follows and the erasure is denoted by* $\mathcal{E}$*,*

$$\underbrace{1\ 1\ 1}_{\mathcal{E}}\ \underbrace{0\ 0\ 0\ 0}_{0}\ \underbrace{0\ 1\ 1\ 0}_{\alpha^5}\ \underbrace{1\ 0\ 0\ 1}_{\alpha^{14}}\ \underbrace{0\ 0\ 1\ 0}_{\alpha}\ \underbrace{0\ 1\ 1\ 1}_{\alpha^{10}}.$$

Applying erasure decoding over $GF(16)$*, the recovered value of symbol 1 is* α^{13}*. Hence, the decoded* $q-$*ary string* $\mathbf{Y_1} \in GF(16)^4$ *is*

$$\mathbf{Y_1} = (\alpha^{13}, 0, \alpha^5, \alpha^{14}).$$

Its equivalent in binary $\mathbf{y_1} \in GF(2)^{16}$ *is*

$$\mathbf{y_1} = \underbrace{1\ 1\ 0\ 1}_{\alpha^{13}}\ \underbrace{0\ 0\ 0\ 0}_{0}\ \underbrace{0\ 1\ 1\ 0}_{\alpha^5}\ \underbrace{1\ 0\ 0\ 1}_{\alpha^{14}}.$$

Now, to check our assumption, we test whether $\mathbf{Y_1}$ *is consistent with the second parity* $p_2 = \alpha^{10}$*. However, the computed parity is*

$$\left(\alpha^{13}, 0, \alpha^5, \alpha^{14}\right)\left(1, \alpha, \alpha^2, \alpha^3\right)^T = \alpha \neq \alpha^{10}.$$

This shows that $\mathbf{Y_1}$ *does not satisfy the second parity. Therefore, we deduce that our assumption on the deletion location is wrong, i.e., the deletion did not occur in block* 1*. Throughout the paper we refer to such cases as* impossible *cases.*
Case 2: *The deletion is assumed to have occurred in block 2, so the sequence is chunked as follows*

$$\underbrace{1\ 1\ 1\ 0}_{\alpha^{11}}\ \underbrace{0\ 0\ 0}_{\mathcal{E}}\ \underbrace{0\ 1\ 1\ 0}_{\alpha^5}\ \underbrace{1\ 0\ 0\ 1}_{\alpha^{14}}\ \underbrace{0\ 0\ 1\ 0}_{\alpha}\ \underbrace{0\ 1\ 1\ 1}_{\alpha^{10}}.$$

Applying erasure decoding, the recovered value of symbol 2 is α^4*. Now, before checking whether the decoded string is consistent with the second parity* p_2*, one can notice that the binary representation of the decoded erasure* (0011) *is not a supersequence of the sub-block* (000)*. So, without checking* p_2*, we can deduce that this case is* impossible.

Definition 1. *We restrict this definition to the case of* $\delta = 1$ *deletion with two MDS parity symbols in* $GF(q)$*. A case* i*,* $i = 1, 2, \ldots, k/\log k$*, is said to be* possible *if it satisfies the*

two criteria below simultaneously.
Criterion 1: The $q-$ary string decoded based on the first parity in case i, denoted by $\mathbf{Y_i}$, satisfies the second parity.
Criterion 2: The binary representation of the decoded erasure is a supersequence of its corresponding sub-block.
If any of the two criteria is not satisfied, the case is said to be impossible.

The two criteria mentioned above are both necessary. For instance, in this example, case 2 does not satisfy Criterion 2 but it is easy to verify that it satisfies Criterion 1. Furthermore, case 1 satisfies Criterion 1 but does not satisfy Criterion 2. A case is said to be possible *if it satisfies both criteria simultaneously.*
Case 3: *The deletion is assumed to have occurred in block 3, so the sequence is chunked as follows*

$$\underbrace{1\,1\,1\,0}_{\alpha^{11}}\ \underbrace{0\,0\,0\,0}_{0}\ \underbrace{1\,1\,0}_{\mathcal{E}}\ \underbrace{1\,0\,0\,1}_{\alpha^{14}}\ \underbrace{0\,0\,1\,0}_{\alpha}\ \underbrace{0\,1\,1\,1}_{\alpha^{10}}.$$

In this case, the decoded binary string is

$$\mathbf{y_3} = \underbrace{1\,1\,1\,0}_{\alpha^{11}}\ \underbrace{0\,0\,0\,0}_{0}\ \underbrace{0\,1\,0\,1}_{\alpha^{8}}\ \underbrace{1\,0\,0\,1}_{\alpha^{14}}\ .$$

By following the same steps as cases 1 and 2, it is easy to verify that both criteria are not satisfied in this case, i.e., case 3 is also impossible.
Case 4: *The deletion is assumed to have occurred in block 4, so the sequence is chunked as follows*

$$\underbrace{1\,1\,1\,0}_{\alpha^{11}}\ \underbrace{0\,0\,0\,0}_{0}\ \underbrace{1\,1\,0\,1}_{\alpha^{13}}\ \underbrace{0\,0\,1}_{\mathcal{E}}\ \underbrace{0\,0\,1\,0}_{\alpha}\ \underbrace{0\,1\,1\,1}_{\alpha^{10}}.$$

In this case, the decoded binary string is

$$\mathbf{y_4} = \underbrace{1\,1\,1\,0}_{\alpha^{11}}\ \underbrace{0\,0\,0\,0}_{0}\ \underbrace{1\,1\,0\,1}_{\alpha^{13}}\ \underbrace{0\,0\,0\,1}_{1}\ .$$

Here, it is easy to verify that this case satisfies both criteria and is indeed possible.
After going through all the cases, case 4 stands alone as the only possible *case. So the decoder declares successful decoding and outputs $\mathbf{y_4}$ ($\mathbf{y_4} = \mathbf{u}$).*

The next example considers another message $\mathbf{u}$ and shows how the proposed decoding scheme can lead to a decoding failure. The importance of Theorem 1 is that it shows that the probability of a decoding failure vanishes a k goes to infinity.

Example 3 (Decoding failure)**.** *Let $k = 16, \delta = 1$ and $c = 2$. Consider the binary message $\mathbf{u}$ given by*

$$\mathbf{u} = 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1.$$

Following the same encoding steps as before, the $q-$ary codeword $\mathbf{X} \in GF(16)^6$ is given by

$$\mathbf{X} = (\alpha^{13}, 0, \alpha^3, \alpha^8, 0, \alpha^8).$$

We still assume that the deletion affects only the systematic bits. Suppose that the 14^{th} bit of the binary codeword $\mathbf{x} \in GF(2)^{24}$ gets deleted

$$\mathbf{x} = 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ \underline{\boldsymbol{1}}\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1.$$

The decoder receives the following 23 bit binary string $\mathbf{y} \in GF(2)^{23}$,

$$\mathbf{y} = 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1.$$

The decoding is carried out as explained in Example 2. The $q-$ary strings decoded in case 1 and case 4 are

$$\mathbf{Y_1} = (\alpha^{13}, \alpha^3, \alpha^2, 1),$$
$$\mathbf{Y_4} = (\alpha^{13}, 0, \alpha^3, \alpha^8).$$

It is easy to verify that both cases 1 and 4 are possible *cases. The decoder here cannot know which of the two cases is the correct one, so it declares a decoding failure.*

Remark 1. *In the previous analysis, each case refers to the assumption that a certain block is affected by the deletion. Hence, among all the cases considered, there is only one correct case that corresponds to the actual deletion location. That correct case always satisfies the two criteria for* possible *cases (Definition 1). So whenever there is only one* possible *case (like in Example 2), the decoding will be successful since that case would be for sure the correct one. However, in general, the analysis may yield multiple* possible *cases. Nevertheless, the decoding can still be successful if all these* possible *cases lead to the same decoded string. An example of this is when the transmitted codeword is the all 0's sequence. Regardless of the deletion position, this sequence will be decoded as all 0's in all the cases. In fact, whenever the deletion occurs within a run of 0's or 1's that extends to multiple blocks, the cases corresponding to these blocks will all be* possible *and lead to the same decoded string. However, sometimes the* possible *cases can lead to different decoded strings like in Example 3, thereby causing a decoding failure.*

V. General Encoding and Decoding of GC Codes

In this section, we describe the general encoding and decoding schemes that can correct up to δ deletions. The encoding and decoding steps for $\delta > 1$ deletions are a direct generalization of the steps for $\delta = 1$ described in the previous section. For decoding, we assume without loss of generality that exactly δ deletions have occurred. Therefore, the length of the binary string $\mathbf{y}$ received by the decoder is $n - \delta$ bits.

A. Encoding Steps

The encoding steps follow from the block diagram that was shown in Fig. 1.
1) *Binary to $q-$ary (Block I, Fig. 1).* The message $\mathbf{u}$ is chunked into adjacent blocks of length $\log k$ bits each. Each block is then mapped to its corresponding symbol in $GF(q)$, where $q = 2^{\log k} = k$. This results in a string $\mathbf{U}$ which consists of $k/\log k$ symbols in $GF(q)$ [5].
2) *Systematic MDS code (Block II, Fig. 1).* $\mathbf{U}$ is then coded using a systematic $(k/\log k + c, k/\log k)$ MDS code over

[5] After chunking, the last block may contain fewer than $\log k$ bits. In order to map the block to its corresponding symbol, it is first padded with zeros to a length of $\log k$ bits. Hence, $\mathbf{U}$ consists of $\lceil k/\log k \rceil$ symbols. We drop the ceiling notation throughout the paper and simply write $k/\log k$.

$GF(q)$, where $c > \delta$ is a code parameter. The $q-$ary codeword $\mathbf{X}$ consists of $k/\log k + c$ symbols in $GF(q)$.
3) *$Q-$ary to binary (Block III, Fig. 1)*. The binary representations of the symbols in $\mathbf{X}$ are concatenated respectively.
4) *Coding parity bits by repetition (Block IV, Fig. 1)*. Only the parity bits are coded using a $(\delta+1)$ repetition code, i.e., each bit is repeated $(\delta + 1)$ times. The resulting binary codeword $\mathbf{x}$ to be transmitted is of length $n = k + c\,(\delta + 1) \log k$.

B. Decoding Steps

1) Decoding the parity symbols of Block II (Fig. 1): these parities are protected by a $(\delta+1)$ repetition code, and therefore can be always recovered correctly by the decoder. A simple way to do this is to examine the bits of $\mathbf{y}$ from right to left and decode deletions instantaneously until the total length of the decoded sequence is $c(\delta + 1)$ bits (the original length of the coded parity bits). Therefore, for the remaining steps we will assume without loss of generality that all the δ deletions have occurred in the systematic bits.
2) The guessing part: the number of possible ways to distribute the δ deletions among the $k/\log k$ blocks is

$$t = \binom{k/\log k + \delta - 1}{\delta}.$$

We index these possibilities by $i, i = 1, \ldots, t$, and refer to each possibility by case i.
The decoder goes through all the t cases (guesses).
3) The checking part: for each case i, $i = 1, \ldots, t$, the decoder (i) chunks the sequence according to the corresponding assumption; (ii) considers the affected blocks erased and maps the remaining blocks to their corresponding symbols in $GF(q)$; (iii) decodes the erasures using the first δ parity symbols; (iv) checks whether the case is *possible* or not based on the criteria described below.

Definition 2. *For δ deletions, a case i, $i = 1, 2, \ldots, t$, is said to be* possible *if it satisfies the two criteria below simultaneously.*
Criterion 1: the decoded $q-$ary string in case i, denoted by $\mathbf{Y_i} \in GF(q)^{k/\log k}$, satisfies the last $c - \delta$ parities simultaneously.
Criterion 2: The binary representations of all the decoded erasures in $\mathbf{Y_i}$ are supersequences of their corresponding sub-blocks.
If any of these two criteria is not satisfied, the case is said to be impossible.

4) After going through all the cases, the decoder declares successful decoding if (i) only one *possible* case exists; or (ii) multiple *possible* cases exist but all lead to the same decoded string. Otherwise, the decoder declares a decoding failure.

VI. Proof of Theorem 1

A. Redundancy

The $(k/\log k + c, k/\log k)$ $q-$ary MDS code in step 2 of the encoding scheme adds a redundancy of $c \log k$ bits. These $c \log k$ bits are then coded using a $(\delta + 1)$ repetition code. Therefore, the overall redundancy of the code is $c(\delta+1)\log k$ bits.

B. Complexity

i) Encoding Complexity: The complexity of mapping a binary string to its corresponding $q-$ary symbol (step 1), or vice versa (step 3), is $\mathcal{O}(k)$. Furthermore, the encoding complexity of a $(k/\log k+c, k/\log k)$ $q-$ary systematic MDS code is quantified by the complexity of computing the c MDS parity symbols. Computing one MDS parity symbol involves $k/\log k$ multiplications of symbols in $GF(q)$. The complexity of multiplying two symbols in $GF(q)$ is $\mathcal{O}(\log^2 q)$. Recall that in our code $q = k$. Therefore, the complexity of step 2 is $\mathcal{O}(\log^2 k \times c \times k/\log k) = \mathcal{O}(c\ k \log k)$. Step 4 in the encoding scheme codes $c \log k$ bits by repetition, its complexity is $\mathcal{O}(c \log k)$. Therefore, the encoding complexity of GC codes is $\mathcal{O}(c\ k \log k) = \mathcal{O}(k \log k)$ since $c > \delta$ is a constant.

ii) Decoding Complexity: The computationally dominant step in the decoding scheme is step 3, that goes over all the t cases and decodes the erasures in each case[6]. The coding scheme works for any systematic MDS code construction. Suppose we apply the Berlekamp-Massey algorithm which can be used in Reed-Solomon codes to decode erasures. The Berlekamp-Massey algorithm has a complexity of $\mathcal{O}(k^2)$ [25]. Since the erasure decoding is performed for all the t cases, the total decoding complexity is $\mathcal{O}\left(tk^2\right)$. The number of cases t is given by

$$t = \binom{k/\log k + \delta - 1}{\delta} = \mathcal{O}\left(\frac{k^\delta}{\log^\delta k}\right).$$

Therefore the decoding complexity is

$$\mathcal{O}\left(\frac{k^{\delta+2}}{\log^\delta k}\right),$$

which is polynomial in k for constant δ.

C. Proof of the probability of decoding failure for $\delta = 1$

To prove the upper bound on the probability of decoding failure, we first introduce the steps of the proof for $\delta = 1$ deletion. Then, we generalize the proof to the case of $\delta > 1$.

The probability of decoding failure for $\delta = 1$ is computed over all possible $k-$bit messages. Recall that the bits of the message $\mathbf{u}$ are i.i.d. Bernoulli$(1/2)$. The message $\mathbf{u}$ is encoded as shown in Fig. 1. For $\delta = 1$, the decoder goes through a total of $k/\log k$ cases, where in a case i it decodes by assuming that block i is affected by the deletion. Let $\boldsymbol{\mathcal{Y}_i}$ be the random variable representing the $q-$ary string decoded in case i, $i = 1, 2, \ldots, k/\log k$, in step 3 of the decoding scheme. Let $\mathbf{Y} \in GF(q)^{k/\log k}$ be a realization of the random variable $\boldsymbol{\mathcal{Y}_i}$. We denote by $\mathcal{P}_r \in GF(q), r = 1, 2, \ldots, c$, the random

[6]The complexity of checking whether a decoded erasure, of length $\log k$ bits, is a supersequence of its corresponding sub-block is $\mathcal{O}(\log^2 k)$ using the Wagner-Fischer algorithm. Hence, Criterion 2 (Definition 2) does not affect the order of decoding complexity.

variable representing the r^{th} MDS parity symbol (Block II, Fig. 1). Also, let $\mathbf{G_r} \in GF(q)^{k/\log k}$ be the MDS encoding vector responsible for generating $\mathcal{P}_r$. Consider $c > \delta$ arbitrary MDS parities $p_1, \dots, p_c$, for which we define the following sets. For $r = 1, \dots, c$,

$$\begin{aligned} \mathrm{A_r} &\triangleq \{\mathbf{Y} \in GF(q)^{k/\log k} |\ \mathbf{G_r^T Y} = p_r\}, \\ \mathrm{A} &\triangleq \mathrm{A_1} \cap \mathrm{A_2} \cap \ldots \cap \mathrm{A_c}. \end{aligned}$$

$\mathrm{A_r}$ and A are affine subspaces of dimensions $k/\log k - 1$ and $k/\log k - c$, respectively. Therefore,

$$|\mathrm{A_r}| = q^{\frac{k}{\log k} - 1} \text{ and } |\mathrm{A}| = q^{\frac{k}{\log k} - c}. \tag{1}$$

Recall that the correct values of the MDS parities are recovered at the decoder, and that for $\delta = 1$, $\boldsymbol{\mathcal{Y}_i}$ is decoded based on the first parity. Hence, for a fixed MDS parity p_1, and for $\delta = 1$ deletion, $\boldsymbol{\mathcal{Y}_i}$ takes values in $\mathrm{A_1}$. Note that $\boldsymbol{\mathcal{Y}_i}$ is not necessarily uniformly distributed over $\mathrm{A_1}$. For instance, if the assumption in case i is wrong, two different message inputs can generate the same decoded string $\boldsymbol{\mathcal{Y}_i} \in \mathrm{A_1}$. We illustrate this later through Example 4. The next claim gives an upper bound on the probability mass function of $\boldsymbol{\mathcal{Y}_i}$ for $\delta = 1$ deletion.

Claim 1. *For any case i, $i = 1, 2, \dots, k/\log k$,*

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | \mathcal{P}_1 = p_1\right) \leq \frac{2}{q^{\frac{k}{\log k} - 1}}.$$

Claim 1 can be interpreted as that at most 2 different input messages can generate the same decoded string $\boldsymbol{\mathcal{Y}_i} \in \mathrm{A_1}$. We assume Claim 1 is true for now and prove it later in Section VI-E. Next, we use this claim to prove the following upper bound on the probability of decoding failure for $\delta = 1$,

$$Pr(F) < \frac{2}{k^{c-2} \ \log k}. \tag{2}$$

In the general decoding scheme, we mentioned two criteria which determine whether a case is *possible* or not (Definition 2). Here, we upper bound $Pr(F)$ by taking into account Criterion 1 only. Based on Criterion 1, if a case i is possible, then $\boldsymbol{\mathcal{Y}_i}$ satisfies all the c MDS parities simultaneously, i.e., $\boldsymbol{\mathcal{Y}_i} \in \mathrm{A}$. Without loss of generality, we assume case 1 is the correct case, i.e., the deletion occurred in block 1. A decoding failure is declared if there exists a *possible* case j, $j = 2, \dots, k/\log k$, that leads to a decoded string different than that of case 1. Namely, $\boldsymbol{\mathcal{Y}_j} \in \mathrm{A}$ and $\boldsymbol{\mathcal{Y}_j} \neq \boldsymbol{\mathcal{Y}_1}$.

Therefore,

$$Pr\left(F | \mathcal{P}_1 = p_1\right) \leq Pr\left(\bigcup_{j=2}^{k/\log k} \{\boldsymbol{\mathcal{Y}_j} \in \mathrm{A}, \boldsymbol{\mathcal{Y}_j} \neq \boldsymbol{\mathcal{Y}_1}\} \middle| \mathcal{P}_1 = p_1 \right) \tag{3}$$

$$\leq \sum_{j=2}^{k/\log k} Pr\left(\boldsymbol{\mathcal{Y}_j} \in \mathrm{A}, \boldsymbol{\mathcal{Y}_j} \neq \boldsymbol{\mathcal{Y}_1} | \mathcal{P}_1 = p_1\right) \tag{4}$$

$$\leq \sum_{j=2}^{k/\log k} Pr\left(\boldsymbol{\mathcal{Y}_j} \in \mathrm{A} | \mathcal{P}_1 = p_1\right) \tag{5}$$

$$= \sum_{j=2}^{k/\log k} \sum_{\mathbf{Y} \in \mathrm{A}} Pr\left(\boldsymbol{\mathcal{Y}_j} = \mathbf{Y} | \mathcal{P}_1 = p_1\right) \tag{6}$$

$$\leq \sum_{j=2}^{k/\log k} \sum_{\mathbf{Y} \in \mathrm{A}} \frac{2}{q^{\frac{k}{\log k} - 1}} \tag{7}$$

$$= \sum_{j=2}^{k/\log k} |\mathrm{A}| \frac{2}{q^{\frac{k}{\log k} - 1}} \tag{8}$$

$$= \sum_{j=2}^{k/\log k} 2 \ \frac{q^{\frac{k}{\log k} - c}}{q^{\frac{k}{\log k} - 1}} \tag{9}$$

$$= \left(\frac{k}{\log k} - 1\right) \frac{2}{q^{c-1}} \tag{10}$$

$$< \frac{2}{k^{c-2} \ \log k}. \tag{11}$$

(4) follows from applying the union bound. (5) follows from the fact that $Pr\left(\mathcal{Y}_j \neq \mathcal{Y}_1 | \mathcal{Y}_j \in A, \mathcal{P}_1 = p_1\right) \leq 1$. (7) follows from Claim 1. (9) follows from (1). (11) follows from the fact that $q = k$ in the coding scheme. Next, to complete the proof of (2), we use (11) and average over all values of p_1.

$$\begin{aligned} Pr(F) &= \sum_{p_1 \in GF(q)} Pr\left(F | \mathcal{P}_1 = p_1\right) Pr\left(\mathcal{P}_1 = p_1\right) \\ &< \sum_{p_1 \in GF(q)} \frac{2}{k^{c-2} \ \log k} \ Pr\left(\mathcal{P}_1 = p_1\right) \\ &< \frac{2}{k^{c-2} \ \log k}. \end{aligned}$$

D. Proof of the probability of decoding failure for δ deletions

We now generalize the previous proof for $\delta > 1$ deletions and show that

$$Pr(F) = \mathcal{O}\left(\frac{1}{k^{c-2\delta} \ \log^{\delta} k}\right).$$

For δ deletions the number of cases is given by

$$t = \binom{k/\log k + \delta - 1}{\delta} = \mathcal{O}\left(\frac{k^{\delta}}{\log^{\delta} k}\right). \tag{12}$$

Consider the random variable $\boldsymbol{\mathcal{Y}_i}$ which represents the $q-$ary string decoded in case i, $i = 1, 2, \dots, t$. The next claim generalizes Claim 1 for $\delta > 1$ deletions.

Claim 2. *There exists a deterministic function h of δ, $h(\delta)$ independent of k, such that for any case i, $i=1,2,\dots,t$,*

$$Pr\left(\boldsymbol{\mathcal{Y}_i}=\mathbf{Y}|\mathcal{P}_1=p_1,\dots,\mathcal{P}_\delta=p_\delta\right)\leq\frac{h(\delta)}{q^{\frac{k}{\log k}-\delta}}.$$

We assume Claim 2 is true for now and prove it later (see Appendix A). To bound the probability of decoding failure for δ deletions, we use the result of Claim 2 and follow the same steps of the proof for $\delta=1$ deletion while considering t cases instead of $k/\log k$. Some of the steps will be skipped for the sake of brevity.

$$Pr\left(F|p_1,\dots,p_\delta\right)\leq Pr\left(\bigcup_{j=2}^{t}\{\boldsymbol{\mathcal{Y}_j}\in\mathrm{A},\boldsymbol{\mathcal{Y}_j}\neq\boldsymbol{\mathcal{Y}_1}\}\middle|p_1,\dots,p_\delta\right) \tag{13}$$

$$\leq\sum_{j=2}^{t}Pr\left(\boldsymbol{\mathcal{Y}_j}\in\mathrm{A}|p_1,\dots,p_\delta\right) \tag{14}$$

$$\leq\sum_{j=2}^{t}\sum_{\mathbf{Y}\in A}\frac{h(\delta)}{q^{\frac{k}{\log k}-1}} \tag{15}$$

$$<\frac{t\cdot h(\delta)}{q^{c-\delta}}. \tag{16}$$

Furthermore,

$$Pr(F)=\sum_{p_1,\dots,p_\delta\in GF(q)}Pr\left(F|p_1,\dots,p_\delta\right)Pr\left(p_1,\dots,p_\delta\right) \tag{17}$$

$$<\sum_{p_1,\dots,p_\delta\in GF(q)}\frac{t\times h(\delta)}{q^{c-\delta}}\,Pr\left(p_1,\dots,p_\delta\right) \tag{18}$$

$$<\frac{t\cdot h(\delta)}{q^{c-\delta}}. \tag{19}$$

Therefore,

$$Pr(F)=\mathcal{O}\left(\frac{1}{k^{c-2\delta}\ \log^{\delta}k}\right). \tag{20}$$

(15) follows from Claim 2. (18) follows from (16). (20) follows from (12), (19) and the fact that $h(\delta)$ is constant (independent of k) for a constant δ.

E. Proof of Claim 1

We focus on case i (i fixed) that assumes that the deletion has occurred in block i. We observe the output $\mathcal{Y}_i$ of the decoder in step 3 of the decoding scheme for all possible input messages, for a fixed deletion position and a given parity p_1. Recall that $\boldsymbol{\mathcal{Y}_i}$ is a random variable taking values in A_1. Claim 1 gives an upper bound on the probability mass function of $\boldsymbol{\mathcal{Y}_i}$ for any i and for a given p_1. We distinguish here between two cases. If case i is correct, i.e., the assumption on the deletion position is correct, then $\boldsymbol{\mathcal{Y}_i}$ is always decoded correctly and it is uniformly distributed over A_1. If case i is wrong, then $\boldsymbol{\mathcal{Y}_i}$ is not uniformly distributed over A_1 as illustrated in the next example.

Example 4. *Let the length of the binary message $\mathbf{u}$ be $k=16$ and consider $\delta=1$ deletion. Let the first MDS parity p_1 be the sum of the $k/\log k=4$ message symbols. Consider the previously defined set A_1 with $p_1=0$. Consider the messages,*

$$\mathbf{U_1}=(0,0,0,0)\in\mathrm{A}_1,$$
$$\mathbf{U_2}=(\alpha,0,0,\alpha)\in\mathrm{A}_1.$$

For the sake of simplicity, we skip the last encoding step (Block IV, Fig. 1), and assume that p_1 is recovered at the decoder. Therefore, the corresponding codewords to be transmitted are

$$\mathbf{x_1}=0\ 0\ \underline{\mathit{0}}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0,$$
$$\mathbf{x_2}=0\ 0\ \underline{\mathit{1}}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0.$$

Now, assume that the 3^{rd} bit of $\mathbf{x_1}$ and $\mathbf{x_2}$ was deleted, and case 4 (wrong case) is considered. It easy to verify that in this case, for both codewords, the $q-$ary output of the decoder will be

$$\boldsymbol{\mathcal{Y}_4}=(0,0,0,0)\in\mathrm{A}_1.$$

This shows that there exists a wrong case i, where the same output can be obtained for two different inputs and a fixed deletion position. Thereby, the distribution of $\boldsymbol{\mathcal{Y}_i}$ over A_1 is not uniform.

The previous example suggests that to find the bound in Claim 1, we need to determine the maximum number of different inputs that can generate the same output for an arbitrary fixed deletion position and a given parity p_1. We call this number γ. Once we obtain γ we can write

$$Pr\left(\boldsymbol{\mathcal{Y}_i}=\mathbf{Y}|\mathcal{D}=d,\mathcal{P}_1=p_1\right)\leq\frac{\gamma}{|\mathrm{A}_1|}=\frac{\gamma}{q^{\frac{k}{\log k}-1}}, \tag{21}$$

where $\mathcal{D}\in\{1,\dots,n\}$ is the random variable representing the position of the deleted bit. We will explain our approach for determining γ by going through an example for $k=16$ that can be easily generalized for any k. We denote by $b_z\in GF(2)$, $z=1,2,\dots,k$, the bit of the message $\mathbf{u}$ in position z.

Example 5. *Let $k=16$ and $\delta=1$. Consider the binary message $\mathbf{u}$ given by*

$$\mathbf{u}=b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8\ b_9\ b_{10}\ b_{11}\ b_{12}\ b_{13}\ b_{14}\ b_{15}\ b_{16}.$$

The extension field used here has a primitive element α, with $\alpha^4=\alpha+1$. Assume that b_3 was deleted and case 4 is considered. Hence, the binary string received at the decoder is chunked as follows

$$\underbrace{b_1\ b_2\ b_4\ b_5}_{S_1}\ \underbrace{b_6\ b_7\ b_8\ b_9}_{S_2}\ \underbrace{b_{10}\ b_{11}\ b_{12}\ b_{13}}_{S_3}\ \underbrace{b_{14}\ b_{15}\ b_{16}}_{\mathcal{E}},$$

where the erasure is denoted by $\mathcal{E}$, and S_1,S_2 and S_3 are the first 3 symbols of $\boldsymbol{\mathcal{Y}_4}$ given by

$$S_1=\alpha^3b_1+\alpha^2b_2+\alpha b_4+b_5\in GF(16),$$
$$S_2=\alpha^3b_6+\alpha^2b_7+\alpha b_8+b_9\in GF(16),$$
$$S_3=\alpha^3b_{10}+\alpha^2b_{11}+\alpha b_{12}+b_{13}\in GF(16).$$

The fourth symbol S_4 of $\boldsymbol{\mathcal{Y}_4}$ is to be determined by erasure decoding. Suppose that there exists another message $\mathbf{u'}\neq\mathbf{u}$ such that $\mathbf{u}$ and $\mathbf{u'}$ lead to the same decoded string

$\boldsymbol{\mathcal{Y}_4} = (S_1, S_2, S_3, S_4)$. Since these two messages generate the same values of S_1, S_2 and S_3, then they should have the same values for the following bits

$$b_1\ b_2\ b_4\ b_5\ b_6\ b_7\ b_8\ b_9\ b_{10}\ b_{11}\ b_{12}\ b_{13}.$$

We refer to these $k - \log k = 12$ bits by "fixed" bits. The only bits that can be different in $\mathbf{u}$ and $\mathbf{u}'$ are b_{14}, b_{15} and b_{16} which correspond to the erasure, and the deleted bit b_3. We refer to these $\log k = 4$ bits by "free" bits. Although these "free" bits can be different in $\mathbf{u}$ and $\mathbf{u}'$, they are constrained by the fact that the first parity in their corresponding $q-$ary codewords $\mathbf{X}$ and $\mathbf{X}'$ should have the same value p_1. Next, we express this constraint by a linear equation in $GF(16)$. Without loss of generality, we assume that $p_1 \in GF(16)$ is the sum of the $k/\log k = 4$ message symbols. Hence, p_1 is given by

$$\begin{aligned} p_1 = \alpha^3 \left(b_1 + b_5 + b_9 + b_{13}\right) + \alpha^2 \left(b_2 + b_6 + b_{10} + b_{14}\right) \\ + \alpha \left(b_3 + b_7 + b_{11} + b_{15}\right) + \left(b_4 + b_8 + b_{12} + b_{16}\right). \end{aligned}$$

Rewriting the previous equation by having the "free" bits on the LHS and the "fixed" bits and p_1 on the RHS we get

$$\alpha b_3 + \alpha^2 b_{14} + \alpha b_{15} + b_{16} = p', \tag{22}$$

where $p' \in GF(16)$ and is given by $p' = p_1 + \alpha^3 \left(b_1 + b_5 + b_9 + b_{13}\right) + \alpha^2 \left(b_2 + b_6 + b_{10}\right) + \alpha \left(b_7 + b_{11}\right) + \left(b_4 + b_8 + b_{12}\right)$. The previous equation can be written as the following linear equation in $GF(16)$,

$$0\alpha^3 + b_{14}\alpha^2 + \left(b_3 + b_{15}\right)\alpha + b_{16} = p'. \tag{23}$$

Now, to determine γ, we count the number of solutions of (23). If the unknowns in (23) were symbols in $GF(16)$, then the solutions of (23) would span an affine subspace of size $q^{k/\log k - 1} = 16^3$. However, the unknowns in (23) are binary, so we show next that it has at most 2 solutions. Let

$$a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0 = p' \tag{24}$$

be the polynomial representation of p' in $GF(16)$ where $(a_3, a_2, a_1, a_0) \in GF(2)^4$. Every element in $GF(16)$ has a unique polynomial representation of degree at most 3. Comparing (23) and (24), we obtain the following system of equations

$$\begin{cases} b_{16} &= a_0, \\ b_3 + b_{15} &= a_1, \\ b_{14} &= a_2, \\ 0 &= a_3. \end{cases}$$

If $a_3 \neq 0$, then (23) has no solution. If $a_3 = 0$, then (23) has 2 solutions because $b_3 + b_{15} = a_1$ has 2 solutions. Therefore, (23) has at most 2 solutions, i.e., $\gamma \leq 2$.

The analysis in Example 5 can be directly generalized for messages of any length k. In general, the analysis yields $\log k$ "free" bits and $k - \log k$ "fixed" bits. Now, we generalize (23) and show that $\gamma \leq 2$ for any k. Without loss of generality, we assume that $p_1 \in GF(q)$ is the sum of the $k/\log k$ symbols of the $q-$ary message $\mathbf{U}$. Consider a wrong case i that assumes that the deletion has occurred in block i. Let d_j be a fixed bit position in block j, $j \neq i$, that represents the position of the deletion. Depending on whether the deletion occurred before or after block i, the generalization of (23) is given by one of the two following equations in $GF(q)$.
If $j < i$,

$$b_{d_j}\alpha^w + b_{\ell+1}\alpha^{m-1} + b_\ell\alpha^{m-2} + \ldots + b_{\ell+m} = p'', \tag{25}$$

If $j > i$,

$$b_{d_j}\alpha^w + b_\ell\alpha^m + b_{\ell+1}\alpha^{m-1} + \ldots + b_{\ell+m-1}\alpha = p'', \tag{26}$$

where $\ell = (i-1)\log k + 1$, $m = \log k - 1$, $w = j\log k - b_j$ and $p'' \in GF(q)$ (the generaliztion of p' in Example 5) is the sum of p_1 and the part corresponding to the "fixed" bits. Suppose that $j < i$. Note that $1 \leq w \leq m$, so (25) can be written as

$$b_{\ell+1}\alpha^{m-1} + \ldots + (b_{d_j} + b_{\ell+o})\alpha^w + \ldots + b_{\ell+m} = p'', \tag{27}$$

where o is an integer such that $1 \leq o \leq m$. Hence, by the same reasoning used in Example 5 we can conclude that (27) has at most 2 solutions. The same reasoning applies for (26), where $j > i$. Therefore, $\gamma \leq 2$ and from (21) we have

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | \mathcal{D} = d, \mathcal{P}_1 = p_1\right) \leq \frac{2}{q^{\frac{k}{\log k} - 1}}. \tag{28}$$

The bound in (28) holds for arbitrary d. Therefore, the upper bound on the probability of decoding failure in (2) holds for any deletion position picked independently of the codeword. Moreover, for any given distribution on $\mathcal{D}$ (like the uniform distribution for example), we can apply the total law of probability with respect to $\mathcal{D}$ and use the result from (28) to get

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | \mathcal{P}_1 = p_1\right) \leq \frac{2}{q^{\frac{k}{\log k} - 1}}.$$

VII. Trade-offs

As previously mentioned, the first encoding step in GC codes (Block I, Fig. 1) consists of chunking the message into blocks of length $\log k$ bits. In this section, we generalize the results in Theorem 1 by considering chunks of arbitrary length ℓ bits ($\ell \leq k$) [7], instead of $\log k$ bits. We show that if $\ell = \Omega(\log k)$, then GC codes have an asymptotically vanishing probability of decoding failure. This generalization allows us to demonstrate two trade-offs achieved by GC codes, based on the code properties in Theorem 2.

Theorem 2. *The Guess & Check (GC) code can correct in polynomial time up to a constant number of δ deletions. Let $c > \delta$ be a constant integer. The code has the following properties:*

1) *Redundancy: $n - k = c(\delta + 1)\ell$ bits.*
2) *Encoding complexity is $\mathcal{O}(k\ell)$, and decoding complexity is $\mathcal{O}\left(\frac{k^{\delta+2}}{\ell^\delta}\right)$.*

[7] For $\ell = k$ and $c = 1$, the code becomes a $(\delta + 1)$ repetition code.

3) *Probability of decoding failure:* $Pr(F) = \mathcal{O}\left(\frac{(k/\ell)^{\delta}}{2^{\ell(c-\delta)}}\right)$.

Proof. See Appendix B. □

The code properties in Theorem 2 enable two trade-offs for GC codes:
1) *Trade-off A:* By increasing ℓ for a fixed k and c, we observe from Theorem 2 that the redundancy increases linearly while the decoding complexity decreases as a polynomial of degree δ. Moreover, increasing ℓ also decreases the probability of decoding failure as a polynomial of degree δ. Therefore, trade-off A shows that by paying a linear price in terms of redundancy, we can simultaneously gain a degree δ polynomial improvement in both decoding complexity and probability of decoding failure.
2) *Trade-off B:* By increasing c for a fixed k and ℓ, we observe from Theorem 2 that the redundancy increases linearly while the probability of decoding failure decreases exponentially. Here, the order of complexity is not affected by c. Therefore, trade-off B shows that by paying a linear price in terms of redundancy, we can gain an exponential improvement in probability of decoding failure, without affecting the order of complexity. This trade-off is of particular importance in models which allow feedback, where asking for additional parities (increasing c) will highly increase the probability of successful decoding.

VIII. Correcting δ Insertions

In this section we show that by modifying the decoding scheme of GC codes, and keeping the same encoding scheme, we can obtain codes that can correct δ insertions, instead of δ deletions. In fact, the resulting code properties for δ insertions (Theorem 3), are the same as that of δ deletions. Recall that ℓ (Section VII) is the code parameter representing the chunking length, i.e., the number of bits in a single block. For correcting δ insertions, we keep the same encoding scheme as in Fig. 1, and for decoding we only modify the following: (i) Consider the assumption that a certain block B is affected by δ' insertions, then while decoding we chunk a length of $\ell+\delta'$ at the position of block B (compared to chunking $\ell-\delta'$ bits when decoding deletions); (ii) The blocks assumed to be affected by insertions are considered to be erased (same as the deletions problem), but now for Criterion 2 (Definition 2) we check if the decoded erasure is a subsequence of the chunked super-block (of length $\ell+\delta'$). The rest of the details in the decoding scheme stay the same.

Theorem 3. *The Guess & Check (GC) code can correct in polynomial time up to a constant number of δ insertions. Let $c > \delta$ be a constant integer. The code has the following properties:*

1) *Redundancy:* $n - k = c(\delta+1)\ell$ *bits.*
2) *Encoding complexity is* $\mathcal{O}(k\ell)$, *and decoding complexity is* $\mathcal{O}\left(\frac{k^{\delta+2}}{\ell^{\delta}}\right)$.
3) *Probability of decoding failure:* $Pr(F) = \mathcal{O}\left(\frac{(k/\ell)^{\delta}}{2^{\ell(c-\delta)}}\right)$.

We omit the proof of Theorem 3 because the same analysis applies as in the proof of Theorem 1. More specifically, the redundancy and the encoding complexity are the same as in Theorem 1 because we use the same encoding scheme. The decoding complexity is also the same as in Theorem 1 because the total number of cases to be checked by the decoder is unchanged. The proof of the upper bound on the probability of decoding failure also applies similarly using the same techniques.

IX. Simulation Results

We simulated the decoding of GC codes and compared the obtained probability of decoding failure to the upper bound in Theorem 1. We tested the code for messages of length $k = 256, 512$ and 1024 bits, and for $\delta = 2, 3$ and 4 deletions. To guarantee an asymptotically vanishing probability

Config.	δ					
	2		3		4	
k	R	$Pr(F)$	R	$Pr(F)$	R	$Pr(F)$
256	0.78	$1.3e^{-3}$	0.67	$4.0e^{-4}$	0.56	0
512	0.86	$3.0e^{-4}$	0.78	0	0.69	0
1024	0.92	$2.0e^{-4}$	0.86	0	0.80	0

TABLE I: The table shows the code rate $R = k/n$ and the probability of decoding failure $Pr(F)$ of GC codes for different message lengths k and different number of deletions δ. The results shown are for $c = \delta + 1$ and $\ell = \log k$. The results of $Pr(F)$ are averaged over 10000 runs of simulations. In each run, a message $\mathbf{u}$ chosen uniformly at random is encoded into the codeword $\mathbf{x}$. δ bits are then deleted uniformly at random from $\mathbf{x}$, and the resulting string is decoded.

of decoding failure, the upper bound in Theorem 1 requires that $c > 2\delta$. Therefore, we make a distinction between two regimes, (i) $\delta < c < 2\delta$: Here, the theoretical upper bound is trivial. Table I gives the results for $c = \delta + 1$ with the highest probability of decoding failure observed in our simulations being of the order of 10^{-3}. This indicates that GC codes can decode correctly with high probability in this regime, although not reflected in the upper bound; (ii) $c > 2\delta$: The upper bound is of the order of 10^{-5} for $k = 1024, \delta = 2$, and $c = 2\delta + 1$. In the simulations no decoding failure was detected within 10000 runs for $\delta + 2 \leq c \leq 2\delta + 1$. In general, the simulations show that GC codes perform better than what the upper bound indicates. This is due to the fact that the effect of Criterion 2 (Definition 2) is not taken into account when deriving the upper bound in Theorem 1. These simulations were performed on a personal computer and the programming code was not optimized. The average decoding time[8] is in the order of milliseconds for $(k = 1024, \delta = 2)$, order of seconds for $(k = 1024, \delta = 3)$, and order of minutes for $(k = 1024, \delta = 4)$. Going beyond these values of k and δ will largely increase the running time due to the number of cases to be tested by the decoder. However, for the file synchronization application in which we are interested (see next section) the values k and δ are relatively small and decoding can be practical.

[8]These results are for $\ell = \log k$, the decoding time can be decreased if we increase ℓ (trade-off A, Section VII).

X. Application to File Synchronization

In this section, we describe how our codes can be used to construct interactive protocols for file synchronization. We consider the model where two nodes (servers) have copies of the same file but one is obtained from the other by deleting d bits. These nodes communicate interactively over a noiseless link to synchronize the file affected by deletions. Some of the most recent work on synchronization can be found in [3]–[7]. In this section, we modify the synchronization algorithm by Venkataramanan *et al.* [3,4], and study the improvement that can be achieved by including our code as a black box inside the algorithm. The key idea in [3,4] is to use *center bits* to divide a large string, affected by d deletions, into shorter segments, such that each segment is affected by one deletion at most. Then, use VT codes to correct these segments. Now, consider a similar algorithm where the large string is divided such that the shorter segments are affected by δ $(1 < \delta \ll d)$ or fewer deletions. Then, use the GC code to correct the segments affected by more than one deletion[9]. We set $c = \delta + 1$ and $\ell = \log k$, and if the decoding fails for a certain segment, we send one extra MDS parity at a time within the next communication round until the decoding is successful. By implementing this algorithm, the gain we get is two folds: (i) reduction in the number of communication rounds; (ii) reduction in the total communication cost. We performed simulations for $\delta = 2$ on files of size 1 Mb, for different numbers of deletions d. The results are illustrated in Table II. We refer to the original scheme in [3,4] by *Sync-VT*, and to the modified version by *Sync-GC*. The savings for $\delta = 2$ are roughly 43% to 73% in number of rounds, and 5% to 14% in total communication cost.

	Number of rounds		Total communication cost	
d	Sync-VT	Sync-GC	Sync-VT	Sync-GC
100	14.52	10.15	5145.29	4900.88
150	16.45	10.48	7735.32	7199.20
200	17.97	10.88	10240.60	9332.68
250	18.93	11.33	12785.20	11415.90
300	20.29	11.70	15318.20	13397.80

TABLE II: Results are averaged over 1000 runs. In each run, a string of size 1 Mb is chosen uniformly at random, and the file to be synchronized is obtained by deleting d bits from it uniformly at random. The total communication cost is expressed in bits. The number of *center bits* used is 25.

Note that the interactive algorithm in [3,4] also deals with the general file synchronization problem where the edited file is affected by both deletions and insertions. There, the string is divided into shorter segments such that each segment is either: (i) not affected by any deletions/insertions; or (ii) affected by only one deletion; or (iii) affected by only one insertion. Then, VT codes are used to correct the short segments. GC codes can also be used in a similar manner when both deletions and insertions are involved. In this case, the string would be divided such as the shorter segments are affected by: (i) δ or fewer deletions; or (ii) δ or fewer insertions.

[9]VT codes are still used for segments affected by only one deletion.

XI. Conclusion

In this paper, we introduced a new family of codes, that we called Guess & Check (GC) codes, that can correct multiple deletions (or insertions) with high probability. We provided deterministic polynomial time encoding and decoding schemes for these codes. We validated our theoretical results by numerical simulations. Moreover, we showed how these codes can be used in applications to remote file synchronization. In conclusion, we point out some open problems and possible directions of future work:

1) GC codes can correct δ deletions or δ insertions. Generalizing these constructions to deal with mixed deletions and insertions (*indels*) is a possible future direction.
2) In our analysis, we declare a decoding failure if the decoding yields more than one possible candidate string. A direction is to study the performance of GC codes in list decoding.
3) From the proof of Theorem 1, it can be seen that bound on the probability of decoding failure of GC codes holds if the deletion positions are chosen by an adversary that does not observe the codeword. It would be interesting to study the performance of GC codes under more powerful adversaries which can observe part of the codeword, while still allowing a vanishing probability of decoding failure.

Appendix A
Proof of Claim 2

Claim 2 is a generalization of Claim 1 for $\delta > 1$ deletions. Recall that the decoder goes through t cases where in each case corresponds to one possible way to distribute the δ deletions among the $k/\log k$ blocks. Claim 2 states that there exists a deterministic function h of δ, $h(\delta)$ independent of k, such that for any case i, $i = 1, 2, \ldots, t$,

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | \mathcal{P}_1 = p_1, \ldots, \mathcal{P}_\delta = p_\delta\right) \leq \frac{h(\delta)}{q^{\frac{k}{\log k} - \delta}},$$

where $\boldsymbol{\mathcal{Y}_i}$ is the random variable representing the $q-$ary string decoded in case i, $i = 1, 2, \ldots, t$, and $\mathcal{P}_r, r = 1, \ldots, c$, is the random variable representing the r^{th} MDS parity symbol.

To prove the claim, we follow the same approach used in the proof of Claim 1. Namely, we count the maximum number of different inputs (messages) that can generate the same output (decoded string) for δ fixed deletion positions $(d_1, \ldots, d_\delta)$ and δ given parities $(p_1, \ldots, p_\delta)$. Again, we call this number γ. Recall that, for δ deletions, $\boldsymbol{\mathcal{Y}_i}$ is decoded based on the first δ parities[10]. Hence, $\boldsymbol{\mathcal{Y}_i} \in \mathrm{A}^\delta$, where

$$\mathrm{A}^\delta \triangleq \mathrm{A}_1 \cap \mathrm{A}_2 \cap \ldots \cap \mathrm{A}_\delta, \tag{29}$$

$$\mathrm{A_r} \triangleq \{\mathbf{Y} \in GF(q)^{k/\log k} | \ \mathbf{G_r^T Y} = p_r\}, \tag{30}$$

for $r = 1, \ldots, \delta$ [11]. We are interested in showing that γ is independent of the binary message length k. To this end, we

[10]The underlying assumption here is that the δ deletions affected exactly δ blocks. In cases where it is assumed that less than δ blocks are affected, then less than δ parities will be used to decode $\boldsymbol{\mathcal{Y}_i}$, and the same analysis applies.

[11]The set A^δ is the generalization of set A_1 for $\delta = 1$.

upper bound γ by a deterministic function of δ denoted by $h(\delta)$. Hence, we establish the following bound

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | d_1, \ldots, d_\delta, p_1, \ldots, p_\delta\right) \leq \frac{\gamma}{|\mathrm{A}^\delta|} < \frac{h(\delta)}{q^{\frac{k}{\log k} - \delta}}. \tag{31}$$

We will now explain the approach for bounding γ through an example for $\delta = 2$ deletions.

Example 6. *Let $k = 32$ and $\delta = 2$. Consider the binary message $\mathbf{u}$ given by*

$$\mathbf{u} = b_1 \ b_2 \ \ldots \ b_{32}.$$

Its corresponding $q-$ary message $\mathbf{U}$ consists of 7 symbols (blocks) of length $\log k = 5$ bits each. The message $\mathbf{u}$ is encoded into a codeword $\mathbf{x}$ using the GC code (Fig. 1). We assume that the first parity is the sum of the systematic symbols and the encoding vector for the second parity is $(1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6)$ [12]*. Moreover, we assume that the actual deleted bits in $\mathbf{x}$ are b_1 and b_7, but the chunking is done based on the assumption that deletions occurred in the 3^{rd} and 5^{th} block (wrong case). Similar to Example 5, it can be shown that the "free" bits are constrained by the following system of two linear equations in $GF(32)$,*

$$\begin{cases} \alpha^4 b_1 + \alpha^3 b_7 + \alpha^2 b_{13} + \alpha^4 b_{14} + b_{15} + \alpha^4 b_{16} \\ \qquad\qquad + \alpha^3 b_{22} + \alpha^2 b_{23} + \alpha b_{24} + b_{25} = p_1', \\ \\ \alpha^4 b_1 + \alpha^4 b_7 + \alpha^2(\alpha^2 b_{13} + \alpha^4 b_{14} + b_{15}) + \alpha^7 b_{16} \\ \qquad\qquad + \alpha^5(\alpha^3 b_{22} + \alpha^2 b_{23} + \alpha b_{24} + b_{25}) = p_2'. \end{cases} \tag{32}$$

To upper bound γ, we upper bound the number of solutions of the system given by (32)*. Equation* (32) *can be written as follows*

$$\begin{cases} (b_1 + b_{16})\alpha^4 + b_7 \alpha^3 + B_1 + B_2 = p_1', \\ (b_1 + b_7)\alpha^4 + b_{16}\alpha^7 + \alpha^2 B_1 + \alpha^5 B_2 = p_2', \end{cases} \tag{33}$$

where B_1 and B_2 are two symbols in $GF(32)$ given by

$$B_1 = \alpha^2 b_{13} + \alpha^4 b_{14} + b_{15}, \tag{34}$$

$$B_2 = \alpha^3 b_{22} + \alpha^2 b_{23} + \alpha b_{24} + b_{25}. \tag{35}$$

Notice that the coefficients of B_1 and B_2 in (33) *originate from the MDS encoding vectors. Hence, for given bit values of b_1, b_7 and b_{16}, the MDS property implies that* (33) *has a unique solution for B_1 and B_2. Furthermore, since B_1 and B_2 have unique polynomial representations in $GF(32)$ of degree at most 4, for given values of B_1 and B_2,* (34) *and* (35) *have at most one solution for $b_{13}, b_{14}, b_{15}, b_{22}, b_{23}, b_{24}$ and b_{25}. We think of bits $b_{13}, b_{14}, b_{15}, b_{22}, b_{23}, b_{24}$ and b_{25} as "free" bits of type I, and bits b_1, b_7, b_{16} as "free" bits of type II. The previous reasoning indicates that for given values of the bits of type II,* (32) *has at most one solution. Therefore, an upper bound on γ is given by the number of possible choices of the bits of type II. Hence, $\gamma \leq 2^3 = 8$.*

[12]The extension field used is $GF(32)$ and has a primitive element α, with $\alpha^5 = \alpha^2 + 1$.

Now, we generalize the previous example and upper bound γ for $\delta > 2$ deletions. Without loss of generality, we assume that the δ deletions occur in δ different blocks. Then, the "free" bits are constrained by a system of δ linear equations in $GF(q)$, where $q = k$. Let $\nu_{(.)}$ and $\mu_{(.)}$ be non-negative integers of value at most k. Each of the δ equations has the following form

$$\sum_{i=1}^{\beta} \alpha^{\nu_i} b_{\mu_i} + \sum_{j=1}^{\delta} \alpha^{\nu_j} B_j = p', \tag{36}$$

where β is the number of "free" bits of type II, $B_j \in GF(q)$ is a linear combination of part of the bits in block j ("free" bits of type I), and $p' \in GF(q)$. The coefficients of B_j, $j = 1, \ldots, \delta$, originate from the MDS code generator matrix. Hence, for given values of the bits of type II, the system of δ equations has a unique solution for B_j, $j = 1, \ldots, \delta$. Furthermore, the linear combination of the bits of type I which gives B_j has the following form

$$B_j = \alpha^m b_{j_1} + \alpha^{m-1} b_{j_2} + \ldots + \alpha^{m-\lambda+1} b_{j_\lambda}, \tag{37}$$

where $m < \log k$ is an integer, and λ is the number of "free" bits of type I. Notice that (37) corresponds to a polynomial representation in $GF(q)$, $q = k$, of degree less than $\log k$. Hence, for a given B_j, (37) has at most one solution. Therefore, γ is upper bounded by the number of possible choices of the bits of type II, i.e., $\gamma \leq 2^\beta$. Recall that, when it is assumed that the block j is affected by deletions, a sub-block of bits is chunked at the position of block j. However, because of the shift caused by the δ deletions, that sub-block may contain bits which do not originate from block j. The λ "free" bits of type I in B_j, $b_{j_1}, \ldots, b_{j_\lambda}$, are the bits of the sub-block which do originate from block j. Since the shift at block j is at most δ positions, it is easy to see that $\lambda > \log k - \delta$. Hence, since $\beta = \delta + \delta(\log k - \lambda)$, we can show that $\gamma < 2^{\delta(\delta+1)} \triangleq h(\delta)$ [13]. Therefore, we have shown that γ is upper bounded by a deterministic function of δ that is independent of k.

Since the bound in (31) holds for arbitrary deletion positions $(d_1, \ldots, d_\delta)$, the upper bound on the probability of decoding failure in Theorem 1 holds for any δ deletion positions picked *independently* of the codeword. Moreover, for any given distribution on the δ deletion positions (like the uniform distribution for example), we can apply the total law of probability and use the result from (31) to get

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | \mathcal{P}_1 = p_1, \ldots, \mathcal{P}_\delta = p_\delta\right) < \frac{h(\delta)}{q^{\frac{k}{\log k} - \delta}}.$$

APPENDIX B
PROOF OF THEOREM 2

We consider the same encoding scheme illustrated in Fig. 1 with the only modification that the message is chunked into blocks of length ℓ bits, and the field is size is $q = 2^\ell$. Taking

[13]If the δ deletions occur in $z < \delta$ blocks, then $\beta = \delta + z(\log k - \lambda)$, thus the upper bound would still hold.

this modification into account, the proof of Theorem 2 follows the same steps of the proof of Theorem 1. It is easy to see from Fig. 1 that the redundancy here becomes $c(\delta+1)\ell$ bits. Also, the number of $q-$ary systematic symbols becomes k/ℓ. Therefore, the total number of cases to be checked by the decoder is

$$t = \binom{k/\ell + \delta - 1}{\delta} = \mathcal{O}\left(\frac{k^\delta}{\ell^\delta}\right). \tag{38}$$

Furthermore, from the proof of Theorem 1 we have that the encoding complexity is

$$\mathcal{O}\left(c \cdot \frac{k}{\ell} \cdot \log^2 q\right) = \mathcal{O}\left(k\ell\right), \tag{39}$$

and the decoding complexity is

$$\mathcal{O}\left(tk^2\right) = \mathcal{O}\left(\frac{k^{\delta+2}}{\ell^\delta}\right). \tag{40}$$

As for the probability of decoding failure, the same intermediary steps of the proof of Theorem 1 apply for chunks of length ℓ instead of $\log k$. In particular, the key property used in the proofs of Claim 1 and Claim 2, is that each binary vector is mapped to a unique element in $GF(q)$. This property also applies here because the field size is $q = 2^\ell$. Hence, from (19) we have

$$Pr(F) < \frac{t \cdot h(\delta)}{q^{c-\delta}} = \mathcal{O}\left(\frac{(k/\ell)^\delta}{2^{\ell(c-\delta)}}\right). \tag{41}$$

From (41) we can see that the probability of decoding failure vanishes asymptotically if ℓ satisfies the following condition

$$2^{\ell(c-\delta)} > (k/\ell)^\delta. \tag{42}$$

By taking logarithms on both sides, the condition on ℓ in (42) can be reduced to $\ell = \Omega(\log k)$.

Acknowledgment

The authors would like to thank Kannan Ramchandran for valuable discussions related to an earlier draft of this work. They also thank Ramezan Paravi Torghabeh for his help with the simulations related to the work in [3,4].

References

[1] S. Kas Hanna and S. El Rouayheb, "Guess & check codes for deletions and synchronization," *arXiv preprint arXiv:1702.04466*, 2017.

[2] R. Varshamov and G. Tenengol'ts, "Correction code for single asymmetric errors," *Automat. Telemekh*, vol. 26, no. 2, pp. 286–290, 1965.

[3] R. Venkataramanan, V. N. Swamy, and K. Ramchandran, "Low-complexity interactive algorithms for synchronization from deletions, insertions, and substitutions," *IEEE Transactions on Information Theory*, vol. 61, no. 10, pp. 5670–5689, 2015.

[4] R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions," in *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1412–1419, Sept 2010.

[5] S. S. T. Yazdi and L. Dolecek, "A deterministic polynomial-time protocol for synchronizing from deletions," *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 397–409, 2014.

[6] F. Sala, C. Schoeny, N. Bitouzé, and L. Dolecek, "Synchronizing files from a large number of insertions and deletions," *IEEE Transactions on Communications*, vol. 64, pp. 2258–2273, June 2016.

[7] N. Ma, K. Ramchandran, and D. Tse, "Efficient file synchronization: A distributed source coding approach," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pp. 583–587, IEEE, 2011.

[8] R. Gabrys, E. Yaakobi, and O. Milenkovic, "Codes in the damerau distance for DNA storage," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 2644–2648, July 2016.

[9] M. Mitzenmacher and E. Drinea, "A simple lower bound for the capacity of the deletion channel," *IEEE Transactions on Information Theory*, vol. 52, pp. 4657–4660, Oct 2006.

[10] E. Drinea and M. Mitzenmacher, "On lower bounds for the capacity of deletion channels," *IEEE Transactions on Information Theory*, vol. 52, pp. 4648–4657, Oct 2006.

[11] S. Diggavi, M. Mitzenmacher, and H. D. Pfister, "Capacity upper bounds for the deletion channel," in *2007 IEEE International Symposium on Information Theory*, pp. 1716–1720, June 2007.

[12] R. Venkataramanan, S. Tatikonda, and K. Ramchandran, "Achievable rates for channels with deletions and insertions," *IEEE Transactions on Information Theory*, vol. 59, pp. 6990–7013, Nov 2013.

[13] Y. Kanoria and A. Montanari, "Optimal coding for the binary deletion channel with small deletion probability," *IEEE Transactions on Information Theory*, vol. 59, pp. 6192–6219, Oct 2013.

[14] M. Rahmati and T. M. Duman, "Upper bounds on the capacity of deletion channels using channel fragmentation," *IEEE Transactions on Information Theory*, vol. 61, pp. 146–156, Jan 2015.

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

[16] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," in *Soviet physics doklady*, vol. 10, p. 707, 1966.

[17] N. J. Sloane, "On single-deletion-correcting codes," *Codes and Designs, de Gruyter, Berlin*, pp. 273–291, 2002.

[18] D. Cullina and N. Kiyavash, "An improvement to levenshtein's upper bound on the cardinality of deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 60, pp. 3862–3870, July 2014.

[19] A. S. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 305–308, 2002.

[20] 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 Transactions on Information Theory*, vol. 58, no. 3, pp. 1804–1808, 2012.

[21] L. J. Schulman and D. Zuckerman, "Asymptotically good codes correcting insertions, deletions, and transpositions," *IEEE transactions on information theory*, vol. 45, no. 7, pp. 2552–2557, 1999.

[22] V. Guruswami and C. Wang, "Deletion codes in the high-noise and high-rate regimes," *IEEE Transactions on Information Theory*, vol. 63, pp. 1961–1970, April 2017.

[23] V. Guruswami and R. Li, "Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 620–624, July 2016.

[24] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1884–1892, SIAM, 2016.

[25] G. Garrammone, "On decoding complexity of reed-solomon codes on the packet erasure channel," *IEEE Communications Letters*, vol. 17, pp. 773–776, April 2013.