Codes correcting deletions in oblivious and random models*

Venkatesan Guruswami† Ray Li‡

Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We prove the existence of binary codes of positive rate that can correct an arbitrary pattern of p fraction of deletions, for any $p < 1$, when the bit positions to delete are picked obliviously of the codeword. Formally, we prove that for every $p < 1$, there exists $\mathcal{R} > 0$ and a code family with a randomized encoder $\mathsf{Enc} : \{0,1\}^{\mathcal{R}n} \to \{0,1\}^n$ and (deterministic) decoder $\mathsf{Dec} : \{0,1\}^{(1-p)n} \to \{0,1\}^{\mathcal{R}n}$ such that for all deletion patterns τ with pn deletions and all messages $m \in \{0,1\}^{\mathcal{R}n}$, $\mathbf{Pr}[\mathsf{Dec}(\tau(\mathsf{Enc}(m))) \neq m] \leq o(1)$, where the probability is over the randomness of the encoder (which is private to the encoder and not known to the decoder).

The oblivious model is a significant generalization of the random deletion channel where each bit is deleted independently with probability p. For the random deletion channel, existence of codes of rate $(1-p)/9$, and thus bounded away from 0 for any $p < 1$, was already known. We give an explicit construction with polynomial time encoding and deletion correction algorithms with rate $c_0(1-p)$ for an absolute constant $c_0 > 0$.

*Research supported in part by NSF grant CCF-1422045

†Email: venkatg@cs.cmu.edu

‡Email: ryli@andrew.cmu.edu

Contents

1 Introduction

Tremendous progress has been made over the decades, including in recent years, on the problem of designing error-correcting codes that can recover from bit (or symbol) errors and erasures, for both probabilistic and adversarial noise models. The problem of correcting closely related insertion and deletion errors has also been studied since the work of Levenshtein in the 60s [22], but has has resisted such progress so far. The difficulty is that in the presence of deletions, the sender and receiver lose valuable synchronization information on the location of various symbols, and this renders many of the standard techniques for constructing either inapplicable or at least very tricky and messy to apply. To quote from Mitzenmacher's survey [25]: "[C]hannels with synchronization errors, including both insertions and deletions as well as more general timing errors, are simply not adequately understood by current theory. Given the near-complete knowledge we have [for] channels with erasures and errors . . . our lack of understanding about channels with synchronization errors is truly remarkable."

In particular, for both the random and adversarial noise models, there are big gaps in our knowledge of the power and limitations of codes against insertions and deletions. For concreteness, let us focus our discussion on the model when there are only deletions, which is already complicated enough, and captures most of the chief difficulties associated with coding against synchronization errors. Note that over large alphabets, in particular those which can grow with the code length, one can add the coordinate position as a header to the codeword symbol, and reduce the deletion model to the simpler erasure model, at the expense of a negligible decrease in rate (due to the addition of the header). However, for fixed alphabets, deletions seem much harder to cope with than erasures. We focus on the most interesting case of binary codes in this work.

In order to set the context and motivation for this work, let us review the situation for adversarial and random deletions in turn, before turning to the contributions of this paper. In each setting we want a code $C \subseteq \{0,1\}^n$ consisting of binary codewords of length n that has good rate (where rate $\mathcal{R}$ means $|C| = 2^{\mathcal{R}n}$, and we would like $\mathcal{R}$ to be as large as possible, and certainly bounded away from 0 as n grows). We also have a deletion fraction $p \in (0,1)$, roughly denoting, as a fraction of n, the number of bits the adversary can delete. In adversarial deletions, the adversary is allowed to delete up to pn bits with full knowledge of a codeword and codebook. A code C is decodable against pn adversarial deletions iff, for any two distinct codewords c and c' in C, it is impossible to apply pn (possibly different) deletions to c and c' and obtain the same result. This is easily seen to be equivalent to the condition that the longest common subsequence between any two codewords of C has less than $(1-p)n$ bits. By a lemma originally due to Levenshtein [22], this condition also ensures that C is capable of correcting any combination of adversarial insertions and deletions totaling pn in number.

When $p \geq 1/2$, the adversary can delete $n/2$ bits that includes either all the 0's or all the 1's in the codeword, resulting in just two possible received sequences. Therefore, it is impossible to correct an adversarial deletion fraction of $1/2$ with rate bounded away from 0. Rather remarkably, we do not know if this trivial limit can be approached: are there codes $C \subseteq \{0,1\}^n$ of size $2^{\Omega_\epsilon(n)}$ decodable against $(1/2-\epsilon)n$ deletions for any $\epsilon > 0$? Or is there some p^* bounded away from $1/2$ such that any code $C \subseteq \{0,1\}^n$ that is decodable against p^*n deletions must have size at most $2^{o(n)}$? This was explicitly raised as a key open problem in [23]. Upper bounds on the asymptotic rate function in terms of the deletion fraction were obtained in [19], improving in some respects Levenshtein's bounds [23]. New upper bounds on code size for a fixed number of deletions that improve over [22] were obtained in [5].

Turning to constructions of deletion codes, Kash et al. [17] proved that randomly chosen codes of small enough rate $R > 0$ can correctly decode against pn adversarial deletions when $p \leq 0.17$. Even non-constructively, this remained the best achievability result (in terms of correctable deletion fraction) until recently. Bukh, Guruswami, and Håstad [1] improved this and showed that there are binary codes of rate

bounded away from 0 decodable against pn adversarial deletions for any $p < \sqrt{2}-1$; furthermore, they also gave an efficient construction of such codes along with an efficient deletion correcting algorithm. Closing the gap between $\sqrt{2}-1$ and $\frac{1}{2}$ for the maximum fraction of correctable deletions remains a tantalizing open problem.

Turning to randomly caused deletions, the basic noise model is the *binary deletion channel* where each bit is deleted independently with probability p. Diggavi and Grossglauser [8] establish that the capacity of the deletion channel for $p \leq \frac{1}{2}$ is at least $1 - h(p)$. Kalai et al. [15] proved this lower bound is tight as $p \to 0$, and Kanoria and Montanari [16] determined a series expansion that can be used to determine the capacity exactly. The capacity of the binary deletion channel is certainly at most that of the simpler binary erasure channel with erasure probability, namely $1 - p$. Rahmati and Duman [26] prove that the capacity is at most $0.4143(1 - p)$ for $p \geq 0.65$. For large deletion probabilities, this is, to our knowledge, the best known upper bound. Drinea and Mitzenmacher [9, 10] proved that the capacity of random deletion codes is at least $(1 - p)/9$, which is within a constant factor of the upper bound. However, to the best of our knowledge, explicit efficiently decodable code constructions were not available for the binary deletion channel for arbitrary $p < 1$; quoting from the first page of Mitzenmacher's survey [25]: "Currently, we have no closed-form expression for the capacity, nor do we have an efficient algorithmic means to numerically compute this capacity." Furthermore, the literature after Mitzenmacher's 2009 survey seems to primarily address the capacity of the random deletion channel [18, 11, 6]. One work that considers efficient recovery against random deletions is by Yazdi and Dolecek [28]. In their setting, two parties Alice and Bob are connected by a two-way communication channel. Alice has a string X, Bob has string Y obtained by passing X through a binary deletion channel with deletion probability $p \ll 1$, and Bob must recover X. They produce a polynomial-time synchronization scheme that trasmits a total of $O(pn \log(1/p))$ bits and allows Bob to recover X with probability exponentially approaching 1.

1.1 Our contributions

With the above context, we now turn to our results in this work. Our main result concerns a natural model that bridges between the adversarial and random deletion models, namely *oblivious deletions*. Here we assume that an arbitrary subset of pn locations of the codeword can be deleted, but these positions must be picked without knowledge of the codeword. The oblivious model is well-motivated in settings where the noise may be mercurial and caused by hard to model physical phenomena, but not by an adversary.

If the code is deterministic, tackling oblivious deletions is equivalent to recovering from worst-case deletions. We allow the encoder to be randomized, and require that for every message m and every deletion pattern τ, most encodings of m can be decoded from the deletion pattern τ. The randomness used at the encoding is private to the encoder and is not needed at the decoder, which we require to be deterministic. Note that the oblivious model generalizes random deletions, as deleting each bit independently with probability p is oblivious to the actual codeword, and with high probability one has $\approx pn$ deletions. Of course, any code which is decodable against pn adversarial deletions is decodable also against pn oblvious deletions, even without any randomization in the encoding. Perhaps surprisingly, we prove that in the oblivious model, the limit of $p \leq 1/2$ does not apply, and in fact one can correct a deletion fraction p approaching 1. This generalizes the result that one can correct a fraction $p \to 1$ of random deletions.

Theorem 1.1 (Main)**.** *For every $p < 1$, there exists $\mathcal{R} > 0$ and a code family with a randomized encoder $\mathsf{Enc} : \{0,1\}^{\mathcal{R}n} \to \{0,1\}^n$ and (deterministic) decoder $\mathsf{Dec} : \{0,1\}^{(1-p)n} \to \{0,1\}^{\mathcal{R}n} \cup \{\perp\}$ such that for all deletion patterns τ with pn deletions and all messages $m \in \{0,1\}^{\mathcal{R}n}$,*

$$\mathbf{Pr}[\mathsf{Dec}(\tau(\mathsf{Enc}(m))) \neq m] \leq o(1) ,$$

where the probability is over the randomness of the encoder (which is private to the encoder and not known to the decoder).[1]

The above result is implied by deterministic codes C decodable from arbitrary pn deletions under average-error criterion; i.e., there is a decoding function $\mathsf{Dec} : \{0,1\}^{(1-p)n} \to C$ such that for every deletion pattern τ with pn deletions, for most codewords $c \in C$, $\mathsf{Dec}(\tau(c)) = c$. We stress that the above is an *existential* result, and the codes guaranteed by the above theorem are not explicitly specified. The decoding algorithm amounts to looking for a codeword which contains the received bit string as a subsequence, and outputting it if there is a unique such codeword.

We also consider the problem of constructing explicit codes with efficient decoding for the binary deletion channel. Here our result is the following. Our rate is worse than the $(1-p)/9$ achieved non-constructively, but has asymptotically the same dependence on p for $p \to 1$.

Theorem 1.2. *Let $p \in (0,1)$. There exists a family of binary codes that (1) has rate $(1-p)/180$, (2) is construtible and encodable in polynomial time, and (3) is decodable with high probability on the binary deletion channel in time $O(n^4)$.*

Our construction concatenates an outer Reed Solomon code concatenated with a good inner binary code. To construct the inner code, we first choose a binary code correcting a small fraction of adversarial deletions. By concentration bounds, duplicating bits of a codeword in a disciplined manner is effective against the random deletion channel, so we duplicate every bit of the binary code $\Theta(1/(1-p))$ times. We further ensure our initial adversarial code has only runs of length 1 and 2 to maximize the effectiveness of duplication.

One might wonder whether it would be possible to use Drinea and Mitzenmacher's existential result [9, 10] of a $(1-p)/9$ capacity lower bound as a black box to achieve a better rate efficiently. We discuss this approach in §5.2 and elaborate on several factors that make the approach nontrivial.

1.2 Related work on oblivious and other noise models

The model of oblivious errors (such as bit flips) has has been studied in the information-theory literature as a particular case of arbitrarily varying channels with state constraints [21] (see the related work section of [13] for more background on this connection). In particular, for the case of bit flips, the capacity against the model of pn oblivious bit flips (for $p \le 1/2$) equals $1 - h(p)$, matching the Shannon capacity of the binary symmetric channel that flips each bit independently with probability p. (This special case was re-proved in [20] by a different simpler random coding argument compared to the original works [3, 4].) Similarly, the capacity against the model of pn oblivious erasures is $1-p$, matching the Shannon capacity of the binary erasure channel. Explicit codes of rate approaching $1 - h(p)$ to correct pn oblivious bit flips (in the sense of Theorem 1.1, with randomized encoding) were given in [13]. This work also considered computationally bounded noise models, such as channels with bounded memory or with small circuits, and gave optimal rate codes for *list* decoding against those models. These models are more general than oblivious errors, but still not as pessimistic as adversarial noise.

Notice that in the case of both erasures and errors, the capacity in the oblivious and random models were the same. It is not clear if this is the case for deletions. The rate of the codes we guarantee in Theorem 1.1 for $p \to 1$ are much worse than the $\Omega(1-p)$ lower bound known for random deletions.

One other interesting model that sits between oblivious and adversarial noise is the *online* model. Here the channel chooses whether to erase/flip/delete bit i of the codeword based on bits $1, \ldots, i$, without knowledge of the future bits of the codeword, and is limited to corrupting at most pn bits total. In a beautiful

[1] The notation $\tau(y)$ for $y \in \{0,1\}^n$ denotes the bit string obtained by applying deletion pattern τ to y.

work, Chen, Jaggi, and Langberg [2] determined the exact capacities of the online erasure channel and the online bit-flip channel. A recent work studies a seemingly slight (but in fact fundamental) restriction of the online model where the channel's decision regarding the i'th bit depends only on the first $(i-1)$ bits and is independent of the current bit [7]. They proved that in the setting of erasures, the capacity in this restricted online model increases to match the capacity $1-p$ of the binary erasure channel, as opposed to $1-2p$ in the true online model.

Note that if a code C corrects pn adversarial deletions, it also corrects pn online deletions. Furthermore, for online deletions the adversary can choose to delete every 1 or every 0 it sees, which means online deletions cannot be decoded for noise rate $p \geq \frac{1}{2}$.

Thus for online deletions the zero rate threshold is between $\sqrt{2}-1$ and $\frac{1}{2}$. It is an interesting challenge to ascertain if one can take advantage of the online restriction, and push some of the ideas in [1] and this work, to enable decoding a fraction of online deletions approaching $\frac{1}{2}$.

To the best of our knowledge, ours is the first work to address oblivious deletions, and online deletions have not been studied in the literature. We feel that given the large gaps in our understanding of coding against adversarial deletions, and the potential of taking advantage of less stringent models of deletion exhibited in this work, further study of these models seems timely and important.

1.3 Structure of this paper

In §2 we outline our oblivious deletion construction and sketch the proof of why it is correct. In §3 we introduce definitions and notation for the remainder of the paper. In §4 we state and prove our construction on codes in the oblivious model with p deletion fraction where $p \in (0,1)$. In §5 we state and prove our construction of efficiently decodable codes in the random deletion setting.

2 Outline of Oblivious Deletion Proof

Our first, naive attempt at this problem is to choose a random subset of $2^{\mathcal{R}N}$ codewords in $\{0,1\}^N$. This technique, however, does not work in the same way it does for oblivious bit-flips. See Appendix B for a discussion on the difficulties of this approach.

Instead of proving Theorem 1.1 directly, we prove a related theorem for decoding in the average case. First, a definition.

Definition 2.1. We say a (nonstochastic) binary code C with rate $\mathcal{R}$ and length N *decodes pN deletions in the average case* if for any deletion pattern τ deleting pN bits, we have

$$|\{x \in C : \exists y \in C \text{ s.t. } x \neq y \text{ and } \tau(x) \leq y\}| \leq o_N(|C|). \tag{1}$$

Theorem 2.2 (Average Case Deletions)**.** *Let $p \in (0,1)$. There exists a constant $\mathcal{R}$ such that there exists infinitely many N for which there is a rate $\mathcal{R}$ code $C \subseteq \{0,1\}^N$ that decodes pN deletions in average case.*

It is standard to show that oblivious and average-case decoding are equivalent. In particular Theorem 2.2 implies Theorem 1.1. For completeness we provide a proof of this implication and state similar results relating adversarial, oblivious, average-case, and random error models in Appendix A. In fact the capacity for decoding in the average case is the same as the capacity for the oblivious channel. Roughly, if we have a length N code C with rate $\mathcal{R}$ that decodes against pN deletions in the average case, then we can group the codewords into sets of size $2^{0.01\mathcal{R}n}$. Then we associate every message with a set of codewords and encode the message by randomly choosing a codeword from its set. Our decoding function simply takes a received

word s and looks for a codeword c such that $s \sqsubseteq c$, i.e. c is a superstring of s. If there is exactly one such c, output the associated codeword, otherwise output $\perp$. Using the fact that C decodes pN deletions in the average case, we can show that for all deletion patterns τ, only a few codewords do not decode correctly via unique decoding in our new stochastic code.

We now outline our construction for Theorem 2.2. Our construction for the average deletion code uses the "clean construction" construction from [1] with appropriately selected parameters. The idea is to choose a concatenated code such that the inner code widely varies in the number of runs between codewords. Specifically, we choose sufficiently large constants R and K and set our inner code to have length $L = 2R^K$. For $i = 1, 2, \ldots, K$, set our inner codewords to be

$$g_i = \left(0^{R^{i-1}} 1^{R^{i-1}}\right)^{L/(2R^{i-1})} \tag{2}$$

where 0^k and 1^k denote strings of k 0s and 1s, respectively. In this way, the number of runs between any two codewords differs by a factor of at least R. Our outer code is a subset of $[K]^n$, so that the total code length is $N = nL$, and we concatenate the code via a function $\psi : [K]^* \to \{0,1\}^*$ that replaces a symbol $i \in [K]$ with the string $g_i \in \{0,1\}^L$. The outer code is chosen via a random process, detailed in the following paragraphs; the process throws out a small subset of "bad" elements of $[K]^n$ and chooses a constant rate code by including each remaining element independently with some small fixed probability. Our decoding function is *unique decoding*. That is, given a received word s, we find a codeword c such that $s \sqsubseteq c$. [2] If such a codeword is unique, our decoder returns that output. Otherwise, the decoder returns $\perp$.

The following example illustrates why the varying run length is powerful even for correct more than $0.5N$ deletions: Suppose $n = 100$, $p = 0.9$, $R \gg 20$, our received word is $s = g_1^{10}$ (that is, 10 copies of g_1 concatenated together) and our code contains the codeword $c = g_2^{100}$. Then s is a subsequence of c if and only if we can identify each of the 10 g_1s with non-overlapping bits of c. However, since g_1 contains over 20 times as many runs as g_2, each g_1 must be identified with a subsequence of c spanning at least 20 inner codewords, i.e. copies of g_2. This means the subsequence s roughly must span at least 200 inner codewords, but c only has 100 inner codewords, contradiction. While this imbalanced run-count behavior is a key to our argument, it is worth noting that the behavior is asymmetric. In particular, while it takes R copies of g_2 to produce g_1 as a subsequence, we only need two copies of g_1 to produce g_2 as a subsequence.

To analyze this code, we leverage the run-count beavhior of the inner codewords. This contrasts with the adversarial setting, where the run-count property of the same code is featured less centrally in the proof of correctness [1]. We show that for any deletion pattern τ with up to pN deletions, two random codewords X and Y are "confusable" with exponentially small probability in N. To be precise, we have for all τ,

$$\mathop{\mathbf{Pr}}_{X,Y\sim U([K]^n)}[\tau(\psi(X)) \sqsubseteq \psi(Y)] \;<\; 2^{-\Omega(n)}.^{3} \tag{3}$$

The idea for this proof can be illustrated in the case that τ deletes only entire inner codewords. If τ deletes pn of the n inner codewords and does not touch the remaining codewords, then $\tau(\psi(X))$ has the same distribution over length $(1-p)N$ binary strings as $\psi(X')$ where $X' \sim U([K]^{(1-p)n})$. Thus we would like to show

$$\mathop{\mathbf{Pr}}_{\substack{X'\sim U([K]^{(1-p)n})\\ Y\sim U([K]^n)}}[\psi(X') \sqsubseteq \psi(Y)] \;<\; 2^{-\Omega(n)}. \tag{4}$$

Consider trying to find $\psi(X')$ as a subtring of $\psi(Y)$ where $X' = X'_1 X'_2 \ldots X'_{(1-p)n}$ This is possible if and only if we match the bits of X' to bits of Y greedily. However, each inner codeword $g_{X'_i}$ spans a large number

[2] The relation $w \sqsubseteq w'$ means that w is a (not-necessarily consecutive) subsequence of w'.

[3] The notation $U([K]^n)$ denotes the uniform distribution on $[K]^n$.

(R) of inner codewords of Y unless the greedy matching encounters at least one Y_j such that $Y_j \leq X'_i$, i.e. a higher frequency inner codeword (if $Y_j < X'_i$, we may need to encounter two such higher frequency inner codewords, but two is at least one). Thus, if $X'_i = 1$ for some i, then the number of inner codewords spanned by $g_{X'_i}$ is approximately distributed as Geometric($1/K$). In general, conditioned on $X'_i = k$, the number of inner codewords spanned by $g_{X'_i}$ is approximately distributed as Geometric(k/K). Thus, the number of inner codewords of Y spanned by a single inner codeword X'_i is

$$\frac{1}{K} \cdot \Theta\left(\frac{K}{1}\right) + \frac{1}{K} \cdot \Theta\left(\frac{K}{2}\right) + \cdots + \frac{1}{K} \cdot \Theta\left(\frac{K}{K}\right) = \Theta(\log K) \tag{5}$$

If we choose K so that $\Theta(\log K) > \frac{2}{1-p}$ then the expected number of inner codewords of Y spanned by X' is more than $\Theta(\log K) \cdot (1-p)n > 2n$, so concentration bounds tell us that the probability that $\psi(X') \sqsubseteq \psi(Y)$ is exponentially small in n.

Note that there is a slight caveat to the above argument because the numbers of inner codewords in $\psi(Y)$ spanned by $\psi(X'_i)$ are not independent across all i. For example, if $\psi(Y)$ begins with $g_2 g_1$ and $\psi(X')$ begins with $g_1 g_1$, then the second inner codeword of $\psi(Y)$ has a few "leftover bits" that easily match with $\psi(X')$'s second inner codeword. However, we can adjust for this independence by relaxing our analysis by a constant factor.

The above addresses the case when the deletion pattern, for each inner codeword, either deletes the codeword entirely or does not modify it at all. We now show how to extend this to general deletion patterns, starting with the case of $p < \frac{1}{2}$.

Note that the above argument only depended on the inner codewords having widely varying runs. By a simple counting argument, we can verify that at least a $(0.5 - p)$ fraction of codewords have at most $(p + (0.5 - p)/2)L = (0.5 + p)L/2$ deletions. Since we are in the regime where $p < \frac{1}{2}$, applying $(0.5 + p)L/2$ deletions to an inner codeword with r runs cannot make the number of runs less than $(0.5 - p)r$, as it can delete at most $(0.5 + p)/2$ fraction of the runs, and deleting each run reduces the number of runs by at most two. If we choose $R \gg 1/(0.5 - p)^2$, then we can guarantee that even if we have a generic deletion pattern, we have a constant fraction $(0.5 - p)$ of positions for which the run-count properties of all inner codewords in those positions are preserved up to a factor of $\sqrt{R}$. Thus, as the number of runs between any two inner codewords differs by a factor of at least R, even after these corruptions the ratio between the number of runs of two "preserved" inner codewords is still at least $\sqrt{R}$. Using the same argument as above and now requiring $\Theta(\log K) > \frac{2}{0.5-p}$, we can conclude that even for general deletion patterns that the probability that two random candidate codewords are confused is exponentially small.

As a technical note, working with deletion patterns directly is messy as they encode a large amount of information, much of which we do not need in our analysis. Furthermore, the caveat mentioned in the clean deletion pattern case regarding the independence of number inner codewords spanned by some $\psi(X'_i)$ becomes more severe for general deletion patterns. This happens because in general deletion patterns, especially later for $p > \frac{1}{2}$, the inner codewords of $\psi(X)$ that have preserved their run-count property might nonetheless be shorter, so many (in particular, $\Theta(1/(1-p))$) inner codewords could match to single inner codewords of $\psi(Y)$. To alleviate this complexity in the analysis, we introduce a technical notion called a *matching* intended to approximate the subsequence relation. This notion allows us to capture only the run-count behavior of the deletion patterns with respect to the inner codewords while also accounting for the lack-of-independence caveat. For a deletion pattern τ, let σ be the associated deletion pattern such that $\sigma(X)$ to removes all outer codeword symbols except the ones in whose position τ preserves the run-count property of all the inner codewords (these exist because we are still in the case $p < 1/2$). In our proof, we argue that if $\tau(\psi(X))$ is a subsequence of $\psi(Y)$, then $\sigma(X)$ has a matching in Y, and that the probability that $\sigma(X)$ has a matching in Y for two codewords X and Y is exponentially small.

To extend the argument for generic deletion patterns from $p < 1/2$ to $p < 1$, we must use a "local list

decoding" idea. Note that when the number of deletion patterns exceeds $1/2$, for every codeword there exists deletion patterns that destroy all or almost all of the information in the codeword, e.g. the deletion pattern that deletes all the 1s (or 0s) of the codeword. For this reason, codes cannot correct against more than $1/2$ fraction of adversarial deletions. However, one can show that this does not happen too frequently allowing us to correct oblivious and average case deletions. In contrast to the $p < 1/2$ case where we found a small, constant fraction of inner codeword positions in which the deletion pattern of the inner codeword preserved the run-count property for *all* inner codewords, we can now find a small, constant fraction of inner codeword positions in which the deletion pattern of the inner codeword preserves the run-count property for *all but a few* inner codewords. For example, even if an inner code deletion pattern deletes every other bit and thus deletes all the information of g_1, the number of runs of $g_2, g_3, \ldots, g_K$ are still preserved. We call this idea "local list decoding" because while we cannot decode our constant fraction of inner codewords uniquely, we can still pin down the inner codewords to a few possibilities. By extending our definition of matching to account for a few inner codewords potentially losing their run-count behavior, we can prove, just as for $p < 1/2$, that $\tau(\psi(X)) \sqsubseteq \psi(Y)$ implies $\sigma(X)$ has a matching in Y, and $\sigma(X)$ having a matching in Y happens with small probability.

At this point of the proof, we have combinatorially established everything we need to prove that our code is decodable in the average case (and thus against oblivious deletions). That is, we have shown that a random candidate codeword in $\psi([K]^n)$ has an exponentially small probability of being confused with another random candidate codeword. Given that codewords have an exponentially small probability of being confusable with each other, it is natural to consider choosing a code by randomly selecting a subset of $[K]^n$. Using this construction, we might try using concentration bounds to show that, for any deletion pattern τ, the probability that we have more than $\epsilon|C|$ codewords (for $\epsilon = o(N)$) that are confusable with some other codeword is at most $2^{-\omega(N)}$, and we can union bound over the at-most-2^N choices of τ. This however does not work directly as the decodability for a given deletion pattern τ depends on the decodability of other deletion patterns. For example, if $p > \frac{1}{2}$ and we happen to choose $c = \psi(11\ldots1) = 0101\ldots01$ as a codeword, then for any deletion pattern τ with pN deletions, $c' \sqsubseteq c$ for *all* candidate codewords c'. From this example alone, the probability of many codewords confusable with c is at least K^{-n} and there are many more examples of such *easily disguised* candidate codewords. Fortunately, we can prove that the number of easily disguised candidate codewords is small. In particular, we show that the majority of elements of $\psi([K]^n)$ are not easily disguised in *all* deletion patterns τ. This intuitively makes sense because, as we have shown, in any deletion pattern τ, on average, words are disguised as an exponentially small fraction of codewords, and because the easily disguised words tend to be easily disguised in every deletion pattern τ. For example, $\psi(11\ldots1_K) = 0101\ldots01$ is easily disguised for any deletion pattern τ.

After throwing out the easily disguised candidate codewords, we randomly choose a constant rate code from the remaining candidate codewords. Careful bookkeeping confirms that with positive probability we obtain a code that decodes pN-deletions in the average case. The bookkeeping is nontrivial, because just as there are a handful of words like $\psi(11\ldots1)$ that are easily disguised as other codewords with deletions, there are also a handful of *easily confused* words like $\psi(KK\ldots K)$ that can be confused with many other words when deletions are applied to it. Furthermore, unlike easily disguised codewords, these easily confused words vary over the different deletion patterns, so we cannot simply throw them out. However, like for easily disguised codewords, we show the number of easily confused words is an exponentially small fraction of the codebook size in expectation, so such words do not contribute significantly to the number of incorrectly decoded codewords. Note the subtle difference between easily disguised and easily confused words: a single easily disguised word like $\psi(11\ldots1)$ causes *many* candidate codewords to fail to decode under our unique decoding, but any easily confused codeword adds *at most one* failed decoding.

We model managing easily disguised and easily confusable codewords via a directed graph, where, roughly, for each deletion pattern, we consider a graph on $[K]^n$ where $\overrightarrow{YX}$ is an edge if and only if $\tau(\psi(X)) \sqsubseteq$

$\psi(Y)$. In our proof, we replace the subsequence relation with the matching relation (see §4.4). In this graph language, the easily disguised codewords correspond to vertices with high outdegree, and the easily confusable codewords correspond to vertices with high indegree.

Our construction illustrates the subtle nature of the oblivious deletion channel and average case errors. These settings share much of the behavior of the adversarial deletion channel such as the fact that for $p > \frac{1}{2}$, every codeword has a deletion pattern destroying all of its information. Consequently, our approach tackles the oblivious and average case errors using a combinatorial argument just as the best adversarial deletion results do [1]. Yet, the relaxed decoding requirement allows us to exploit it to correct a fraction of deletions approaching 1.

3 Preliminaries

General Notation. For a boolean statement P, let $\mathbb{1}[P]$ be 1 if P is true and 0 otherwise.

Throughout the paper, $\log x$ refers to the base-2 logarithm.

We use interval notation $[a, b] = \{a, a+1, \ldots, b\}$ to denote intervals of integers, and we use $[a] = [1, a] = \{1, 2, \ldots, a\}$.

For a set S and an integer a, let $\binom{S}{a}$ denote the family subsets of S of size a. Let $U(S)$ denote the uniform distribution on S.

Words. A *word* is a sequence of symbols from some *alphabet*. We denote string concatenation of two words w and w' with ww'. We denote $w^k = ww\cdots w$ where there are k concatenated copies of w. We also denote a concatenation of a sequence of words as $w_1 w_2 \cdots w_k = \prod_{i=1}^{k} w_i$. We denote words from binary alphabets with lowercase letters c, s, w and words from non-binary alphabets with capital letters X, Y, Z.

A *subsequence* of a word w is a word obtained by removing some (possibly none) of the symbols in w.

A *subword* or *interval* of a word w is a contiguous subsequence of characters from w. We identify intervals of words with intervals of integers corresponding to the indices of the subsequence. For example, the interval $\{1, 2, \ldots, |w|\} = [1, |w|]$ is identified with the entire word w.

Let $w' \sqsubseteq w$ denote "w' is a subsequence of w".

Define a *run* of a word w to be a maximal single-symbol subword. That is, a subword w' in w consisting of a single symbol such that any longer subword containing w' has at least two different symbols. Note the runs of a word partition the word. For example, 110001 has 3 runs: one run of 0s and two runs of 1s.

Deletion Patterns. A *deletion pattern* is a function τ that removes a fixed subset of symbols from words of a fixed length. Let $\mathcal{D}(n, m)$ denote the set of deletion patterns τ that operate on length n words and apply exactly m deletions. For example $\tau : x_1x_2x_3 \mapsto x_1x_3$ is a member of $\mathcal{D}(3, 1)$. Let $\mathcal{D}(n) = \cup_{m=0}^{n} \mathcal{D}(n, m)$.

We identify each deletion pattern $\tau \in \mathcal{D}(n, m)$ with a size m subset of $[n]$ corresponding to the deleted bits. We often use sets to describe deletion patterns when the length of the codeword is understood. For example $[n]$ refers to the single element of $\mathcal{D}(n, n)$. Accordingly, let $\subseteq$ be a partial order on deletion patterns corresponding to set inclusion, and let $|\tau|$ denote the number of bits deleted by τ. As such, we have $\tau \subseteq \tau'$ implies $|\tau| \leq |\tau'|$.

For a word w and $\tau \in \mathcal{D}(|w|)$, let $\tau(w)$ and $w \setminus \tau$ both denote the result of applying τ to w. We use the second notation when we identify sets with deletion patterns, as in the above paragraph where the set elements correspond to the deleted positions.

In a concatenated code with outer code length n and inner code length L, we can identify a deletion pattern on the entire codeword $\tau \in \mathcal{D}(nL)$ as the "concatenation" of n deletion patterns $\tau_1 \frown \cdots \frown \tau_n$, one for each inner codeword. To be precise, for all $\tau \in \mathcal{D}(nL)$, there exists $\tau_i \in \mathcal{D}(L)$ such that $\tau = \cup_{i=1}^{n} \{j + (i-1)L : j \in \tau_i\}$, and we denote this by $\tau = \tau_1 \frown \cdots \frown \tau_n$. Using this notation, we refer to τ_i as

an *inner code deletion pattern*.

Graphs. In a (directed or undirected) graph G, let $V(G)$ and $E(G)$ denote the vertex set and edge set of G respectively. For a subset $W \subseteq V(G)$ of the vertices, let $G \restriction W$ denote the subgraph induced by W. We use $G \restriction W$ instead of the more standard $G|_W$ to avoid too many subscripts. For a vertex $v \in V(G)$, let $\deg_G(v)$ denote the degree of v in G when G is undirected, and let $\mathrm{indeg}_G(v), \mathrm{outdeg}_G(v)$ denote the indegree and outdegree of v, respectively, in G when G is directed. We drop the subscript of G in deg, indeg and outdeg notations when the graph G is understood.

Concentration Bounds. We use the following forms of Chernoff bound.

Lemma 3.1 (Chernoff)**.** *Let $A_1, \ldots, A_n$ be independent random variables taking values in $[0,1]$. Let $A = \sum_{i=1}^{n} A_i$ and $\delta \in [0,1]$. Then*

$$\mathbf{Pr}[A \leq (1-\delta)\,\mathbb{E}[A]] \;\leq\; \exp\left(-\delta^2\,\mathbb{E}[A]/2\right). \tag{6}$$

Furthermore, if $A_1, \ldots, A_n$ are Bernoulli random variables, then

$$\mathbf{Pr}[A \geq (1+\delta)\,\mathbb{E}[A]] \;\leq\; \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mathbb{E}[A]} \tag{7}$$

We also use the submartingale form of Azuma's Inequality.

Lemma 3.2 (Azuma)**.** *Let c be a constant. Let $X_1, X_2, \ldots$ be a submartingale such that $|X_i - X_{i-1}| \leq c$ for all i. Then for all positive reals t, we have*

$$\mathbf{Pr}[X_k - X_1 \leq -t] \leq \exp\left(\frac{-t^2}{2c^2(k-1)}\right). \tag{8}$$

4 Deletion code decoding p-fraction of oblivious deletions

4.1 Overview of proof

In §2 we gave a high level overview. We now begin with a brief snapshot of the proof structure and how it is organized.

We present our general code construction in §4.2. The construction uses the "clean construction" in [1] and an outer code that we choose randomly. We analyze properties of the concatenated construction in §4.3. We begin by extracting the useful behavior of deletion patterns with respect to our codewords. This deletion pattern analysis culminates in Lemma 4.7, allowing us to define the *signature* (Definition 4.8) of a deletion pattern. The key result of §4.3 is Proposition 4.18. It states that for any deletion pattern τ, the probability that two random candidate codewords c, c' are confusable (in the sense that $\tau(c) \sqsubseteq c'$) is exponentially small in the code length. However, because working with deletion patterns directly is messy, the proposition is written in the language of *matchings*, a technical notion defined in Definition 4.11. In short, because the inner codewords are nicely behaved, we do not need to know the exact details of the behavior of a given deletion pattern τ, but rather only need certain properties of it, given by its signature. We thus define a matching to approximate the subsequence relation using only the signature of τ, so that "$\sigma(X)$ is matchable in Y" (where σ is the outer code deletion pattern given by τ's signature) holds roughly when "$\tau(\psi(X)) \sqsubseteq \psi(Y)$" holds.

Combinatorially, Proposition 4.18 allows us to finish the proof. As stated in §2, for our outer code we consider $[K]^n$ minus a small set of easily disguised candidate codewords. The notion of a easily disguised

candidate codeword is well defined by Lemma 4.23. For a sufficiently small constant γ, we randomly choose a size $2^{\gamma n}$ outer code over the remaining outer codewords. In §4.4, we use the graph language described in §2 and prove Lemma 4.27, which roughly states that in a sparse directed graph, if we randomly sample a small subset of vertices, the induced subgraph is also sparse (for some appropriate definition of sparse) with high probability. Finally, in §4.5, we piece together these results, showing that Lemma 4.27 guarantees, with positive probability, that our random code combinatorially decodes against pN deletions in the average case.

4.2 Construction

Let $p \in (0,1)$, and let $\lambda = \lambda(p)$ be the smallest integer such that $(1+p)/2 < 1 - 2^{-\lambda}$. For our argument any λ such that $p < 1 - 2^{-\lambda}$ suffices. In particular, for $p < \frac{1}{2}$, we can choose $\lambda = 1$, slightly simplifying the argument as described in §2. However, we choose λ to be the smallest λ such that $(1+p)/2 < 1 - 2^{-\lambda}$ to ensure a clean formula for the rate.

Let δ be such that $p = 1 - 2^{-\lambda} - \delta$. Let n be a positive integer. With hindsight, choose

$$K = 2^{\lceil 2^{\lambda+5}/\delta \rceil}, \qquad R = 4K^4, \qquad L = 2R^K, \qquad N = nL. \tag{9}$$

Note that R is even. For the remainder of this section, the variables p, λ, δ, K, R and L are fixed.

In this way we have $1 - 2^{-\lambda} - \frac{1}{\sqrt{R}} - \frac{\delta}{2} > p$. For $i = 1, \ldots, K$, let g_i be the length L word

$$g_i = \left(0^{R^{i-1}} 1^{R^{i-1}}\right)^{L/(2R^{i-1})}. \tag{10}$$

Consider the encoding $\psi : [K]^* \to \{0,1\}^*$ where $\psi(X_1 \cdots X_k) = g_{X_1} g_{X_2} \cdots g_{X_k}$. We construct a concatenated code where the outer code is a length n code over $[K]$ and the inner code is $\{g_1, g_2, \ldots, g_K\} \subseteq \{0,1\}^L$. For the outer code, we choose a random code C_{out} where each codeword is chosen uniformly at random from $[K]^n$ minus a small undesirable subset that we specify later. We choose our decoding function to be *unique decoding*. That is, our decoder iterates over all codewords $c \in C$ and checks if the received word s is a subsequence of c. If it is a subsequence of exactly one c, the decoder returns that c, otherwise it fails. While this decoder is not optimal in terms of the fraction of correctly decoded codewords (it could try to break ties instead of just giving up), it is enough for this proof. Furthermore, since we are showing combinatorial decodability, we do not need the decoder to be efficient.

If, for some $p' > p$, a code can decode $p'nL$ average case or oblivious deletions, then it can decode pnL deletions. Thus, we can decrease δ until δn is an even integer, so may assume without loss of generality that δn is an even integer.

4.3 Analyzing construction behavior

Definition 4.1. A *g_i-segment* is an interval in $[L]$ corresponding to a run of g_i. Note that the g_i-segments partition $[L]$ and are of the form $[1 + aR^{i-1}, (a+1)R^{i-1}]$ for $a \in \{0, \ldots, L/R^{i-1} - 1\}$.

Note that g_i has $2R^{K+1-i}$ runs. In particular, the number of runs greatly varies between inner codewords. This property makes the concatenated construction powerful because it is difficult to find common subsequences of different inner codewords. The following definition allows us to reason about the inner codewords in terms of their run counts.

Definition 4.2. We say an inner code deletion pattern σ *preserves* g_i if $\sigma(g_i)$ has at least $2R^{K+1-i}/\sqrt{R} = 2R^{K+\frac{1}{2}-i}$ runs. Otherwise we say σ *corrupts* g_i.

We start with a basic but useful fact about deletion patterns and runs.

Lemma 4.3. *Suppose w is a word with r runs $I_1, \ldots, I_r$ and τ is a deletion pattern such that $\tau(w)$ has r' runs. We can think of these runs I_k as subsets of consecutive indices in $\{1, \ldots, |w|\}$. Then the number of runs I_k of w completely deleted by τ, i.e. satisfying $I \subseteq \tau$ when τ is thought of as a subset of $\{1, \ldots, |w|\}$, is at least $\frac{r-r'}{2}$.*

Proof. Deleting any run reduces the number of runs in a word by at most 2, and τ reduces the number of runs by $r - r'$, so the claim follows. □

The next lemma establishes the usefulness of widely varying runs in our construction. It says that even when an inner code deletion pattern has a large number of deletions, most of the inner codewords still look the same in terms of the number of runs. The intuition for the lemma is as follows. Consider the extreme example of an inner code deletion pattern σ that "completely corrupts" the inner codewords $g_1, \ldots, g_\lambda$. That is, σ deletes all the zeros of each of $g_1, \ldots, g_\lambda$. Since σ deletes all the zeros of g_λ, it must delete every other run of $R^{\lambda-1}$ bits, thus deleting $L/2$ bits. Applying these deletions alone to $g_{\lambda-1}$ leaves it with half as many runs of length exactly $R^{\lambda-2}$. However, since σ also deletes all zeros of $g_{\lambda-1}$, it must delete every other run of $R^{\lambda-2}$ bits of the remaining $L/2$ bits, thus deleting $L/4$ more bits. Similarly, since σ deletes all zeros of $g_{\lambda-2}$, it must delete an additional $L/8$ bits. Continuing this logic, we have σ must delete $L(1-2^{-\lambda})$ bits total. This tells us that if an inner code deletion pattern completely corrupts the inner codewords $g_1, \ldots, g_\lambda$, it needs $L(1-2^{-\lambda})$ deletions. This logic works even if we chose to corrupt any subset of λ inner codewords other than $\{g_1, \ldots, g_\lambda\}$. One can imagine that corrupting (according to Definition 4.2) inner codewords is almost as hard as completely corrupting them, so adding some slack gives the lemma.

Lemma 4.4. *If σ is an inner code deletion pattern with $|\sigma| \leq L\left(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}}\right)$, then σ preserves all but at most $\lambda - 1$ choices of g_i.*

Proof. Suppose λ is a positive integer such that σ corrupts g_i for λ different values of i, say $i_1 > i_2 > \cdots > i_\lambda$. We wish to show $|\sigma| > L\left(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}}\right)$.

Recall that a g_i segment is an interval of the form $[1+aR^{i-1}, (a+1)R^{i-1}]$. Inductively define the collections of intervals $\mathcal{I}_1, \ldots, \mathcal{I}_\lambda$ and the sets of indices $I_1, \ldots, I_\lambda$ as follows. For $1 \leq a \leq \lambda$, set

$$\mathcal{I}_a \;=\; \left\{ J : J \text{ is } g_{i_a}\text{-segment, } J \subseteq \sigma \setminus \bigcup_{b=1}^{a-1} I_b \right\} \quad \text{and} \quad I_a \;=\; \bigcup_{J \in \mathcal{I}_a} J. \tag{11}$$

Intuitively, $\mathcal{I}_1$ as the set of runs in g_{i_1} that are entirely deleted by σ, and I_1 is the set of those deleted indices. Then, $\mathcal{I}_2$ is the set of runs of g_{i_2} deleted by σ but not already accounted for by $\mathcal{I}_1$, and I_2 is the set of bits in the runs of $\mathcal{I}_2$. We can interpret $\mathcal{I}_3, \ldots, \mathcal{I}_\lambda$ and $I_3, \ldots, I_\lambda$ similarly. By construction, $I_1, \ldots, I_\lambda$ are disjoint, and their union is a subset of σ (thought of as a subset of $[L]$), so $\sum_{b=1}^{\lambda} |I_b| \leq |\sigma|$. It thus suffices to prove

$$\sum_{b=1}^{\lambda} |I_b| \;>\; L\left(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}}\right). \tag{12}$$

Note that for any $j < j'$, every $g_{j'}$-segment is the disjoint union of $R^{j'-j}$ many g_j-segments. We thus have $[L]$ is the disjoint union of g_{i_a}-segments and I_b is also the disjoint union of g_{i_a}-segments when $b < a$ (and thus $i_b > i_a$). Hence, $[L] \setminus \cup_{b=1}^{a-1} I_b$ is the disjoint union of g_{i_a}-segments. Furthermore, as R is even, each I_b covers an even number of g_{i_a}-segments, so the segments of $[L] \setminus \cup_{b=1}^{a-1} I_b$ alternate between segments corresponding

to runs of 0s in g_{i_a} and segments corresponding to runs of 1s in g_{i_a}. It follows that all runs of $g_{i_a} \setminus \cup_{b=1}^{a-1} I_b$ have length exactly R^{i_a-1}, so the number of runs in the string $g_{i_a} \setminus \cup_{b=1}^{a-1} I_b$ is

$$\frac{L}{R^{i_a-1}} - \sum_{b=1}^{a-1} R^{i_b-i_a}|\mathcal{I}_b|. \tag{13}$$

By construction, the only g_{i_a}-segments that are deleted by σ are the intervals covered by $I_1 \cup \cdots \cup I_a$. Since σ corrupts g_{i_a}, we know $\sigma(g_{i_a})$ has less than $L/(R^{i_a-1}\sqrt{R})$ runs. By Lemma 4.3, we have

$$|\mathcal{I}_a| \;>\; \frac{1}{2}\left(\left(\frac{L}{R^{i_a-1}} - \sum_{b=1}^{a-1} R^{i_b-i_a}|\mathcal{I}_b|\right) - \frac{L}{R^{i_a-1}\sqrt{R}}\right). \tag{14}$$

Simplifying and using $|I_b| = R^{i_b-1}|\mathcal{I}_b|$ for all b, we obtain

$$|I_a| \;>\; L\left(\frac{1}{2} - \frac{1}{2\sqrt{R}}\right) - \frac{1}{2}\sum_{b=1}^{a-1}|I_b|. \tag{15}$$

From here it is easy to verify by induction that, for all $1 \le a \le \lambda$, we have

$$\sum_{b=1}^{a}|I_b| \;>\; L\left(1-\frac{1}{2^a}\right)\left(1-\frac{1}{\sqrt{R}}\right). \tag{16}$$

Indeed, (15) for $a = 1$ provides the base case, and if we know (16) for some $a-1$, then by (15) we have

$$\begin{aligned}\sum_{b=1}^{a}|I_b| \;&>\; L\left(\frac{1}{2} - \frac{1}{2\sqrt{R}}\right) + \frac{1}{2}\sum_{b=1}^{a-1}|I_b| \\ &>\; L\left(\frac{1}{2} - \frac{1}{2\sqrt{R}}\right) + \frac{L}{2}\left(1-\frac{1}{2^{a-1}}\right)\left(1-\frac{1}{\sqrt{R}}\right) \\ &=\; L\left(1-\frac{1}{2^a}\right)\left(1-\frac{1}{\sqrt{R}}\right),\end{aligned} \tag{17}$$

completing the induction. The induction proves (12), from which we have

$$|\sigma| \;\ge\; \sum_{b=1}^{\lambda}|I_b| \;>\; L\left(1-\frac{1}{2^\lambda}-\frac{1}{\sqrt{R}}\right), \tag{18}$$

as desired. □

Lemma 4.4 motivates the following definition.

Definition 4.5. We say an inner code deletion pattern $|\sigma|$ is *ℓ-admissible* if $|\sigma| \le L(1 - 1/2^{\ell+1} - \frac{1}{\sqrt{R}})$.

If, for some ℓ, σ is ℓ-admissible, then Lemma 4.4 tells us σ corrupts at most ℓ different g_i. However, note that ℓ-admissibility is stronger that corrupting at most ℓ different g_i as ℓ-admissibility gives a stronger upper bound on the number of deletions in σ, which is necessary in Lemma 4.12.

Lemma 4.6. *Let $\delta > 0$. Let $\tau = \tau_1 \frown \cdots \frown \tau_n$ be a deletion pattern with at most $(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}} - \frac{\delta}{2})N$ deletions. There are at least δn indices i such that τ_i is $(\lambda-1)$-admissible.*

Proof. By a simple counting argument, we have $|\tau_i| > L(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}})$ for at most

$$\frac{n \cdot L\left(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}} - \frac{\delta}{2}\right)}{L\left(1 - \frac{1}{2^\lambda} - \frac{1}{\sqrt{R}}\right)} \leq n\left(1 - \frac{\delta/2}{1 - 2^{-\lambda}}\right) \leq n(1 - \delta) \tag{19}$$

values of i. For the remaining at least δn values of i, we have τ_i is $(\lambda - 1)$-admissible. □

The following corollary allows us to reduce our analysis of a deletion pattern τ to analyzing positions where τ's inner code deletion pattern is $(\lambda - 1)$-admissible. We effectively assume that our deletion pattern completely deletes all inner codewords with a non-admissible index.

Lemma 4.7. *Let $\tau \in \mathcal{D}(nL, pnL)$. There exists $\tau' \in \mathcal{D}(\delta nL)$, $\sigma \in \mathcal{D}(n, (1-\delta)n)$ and sets $S_1, \ldots, S_{\delta n}$ such that*

1. *$|S_i| = \lambda - 1$ for all i,*
2. *for all $X \in [K]^n$, we have $\tau'(\psi(\sigma(X))) \sqsubseteq \tau(\psi(X))$, and*
3. *when we write $\tau' = \tau'_1 \frown \cdots \frown \tau'_{\delta n}$ as the concatenation of δn inner code deletion patterns, we have, for all i and all $j \notin S_i$, that $\tau'_i \in \mathcal{D}(L)$ preserves g_j.*

Proof. Let $\tau = \tau_1 \frown \cdots \frown \tau_n$. By Lemma 4.6, there exist δn indices $\ell_1 < \cdots < \ell_{\delta n}$ such that, for $i = 1, \ldots, \delta n$, τ_{ℓ_i} is $(\lambda - 1)$-admissible. Choose $\sigma \in \mathcal{D}(n, (1-\delta)n)$ via $\sigma(X_1 \ldots X_n) = X_{\ell_1} X_{\ell_2} \ldots X_{\ell_{\delta n}}$, and choose $\tau' \in \mathcal{D}(\delta nL)$ via $\tau' = \tau_{\ell_1} \frown \cdots \frown \tau_{\ell_{\delta n}}$. We have $\tau' \circ \psi \circ \sigma(X) \sqsubseteq \tau \circ \psi(X)$ for all $X \in [K]^n$ because $\tau' \circ \psi \circ \sigma(X)$ is simply the result of deleting the remaining bits in inner codewords of non-admissible indices in $\tau \circ \psi(X)$. By construction, each $\tau_{\ell_i} \in \mathcal{D}(L)$ is $(\lambda - 1)$-admissible, so we can choose $S_1, \ldots, S_{\delta n}$ by setting S_ℓ to be the set that τ_{ℓ_i} corrupts. Note that some S_i may have size less than $\lambda - 1$, but we can arbitrarily add elements of $[K]$ to S_i until it has $\lambda - 1$ elements. This is okay as item 3 in the corollary statement remains true if we add elements to S_i. □

In our analysis, for a deletion pattern $\tau = \tau_1 \frown \cdots \frown \tau_n$, we only care about the behavior of a given inner code deletion pattern τ_i as far as the set S_i of inner codewords g_j that it corrupts; instead of considering all possible deletion patterns $\tau \in \mathcal{D}(nL)$, it suffices to only consider all possible $\sigma, S_1, \ldots, S_{\delta n}$. This motivates the following definition.

Definition 4.8. The *signature* of a deletion pattern τ is $(\sigma, S_1, S_2, \ldots, S_{\delta n})$, where $\sigma, S_1, \ldots, S_{\delta n}$ are given by Lemma 4.7. If the choice of $\sigma, S_1, \ldots, S_{\delta n}$ satisfying the conditions of Lemma 4.7 are not unique, then choose one such collection of $\sigma, S_1, \ldots, S_{\delta n}$ arbitrarily and assign this collection to be the signature of τ.

Below we define the matchability relation $\prec$. Definition 4.11 allows us to worry only about the signature of a deletion pattern τ rather than τ itself. Intuitively, we can think of the matchability relation $\prec$ as an approximation of the subsequence relation $\sqsubseteq$. Proposition 4.12 establishes this relationship formally. Specifically, it states that if τ is a deletion pattern with signature $(\sigma, S_1, \ldots, S_{\delta n})$, then for $X, Y \in [K]^n$, we have $\sigma(X) \sqsubseteq Y$ implies that $\sigma(X)$ has a matching in Y with appropriate parameters. This means that if we want to show there are few incorrectly decoded codewords in a given code, it suffices to show that few codewords have an appropriately parameterized matching in some other codeword.

We first define type-A and type-B pairs of indices $(i, j) \in \{1, \ldots, |X|\} \times \{1, \ldots, |Y|\}$. Intuitively, pairs (i, j) are type-B only if $\tau_i(\psi(X_i))$ has many more runs than $\psi(Y_j)$, i.e. it is difficult to find (continguous) subwords of $\tau_i(\psi(X_i))$ as subsequences of $\psi(Y_j)$.

Definition 4.9. Let $X, Y \in [K]^*$ be words over the alphabet $[K]$ and let $S_1, \ldots, S_{|X|}$ be subsets of K. Given a pair $(i, j) \in \{1, \ldots, |X|\} \times \{1, \ldots, |Y|\}$, we say (i, j) is *type-A* with respect to $X, Y, S_1, \ldots, S_{|X|}$ (or simply *type-A* if the parameters are understood) if $X_i \in S_i$ or $X_i \geq Y_j$. Call a pair (i, j) *type-B* with respect to $X, Y, S_1, \ldots, S_{|X|}$ otherwise.

Definition 4.10. Let $X, Y \in [K]^*$ be words over the alphabet $[K]$, let $S_1, \ldots, S_{|X|}$ be subsets of K, and let s and t be positive integers. The following algorithm constructs the $(s, t, S_1, \ldots, S_{|X|})$ *matching* of X with Y. Begin with a pair $(a, b) = (1, 1)$. The first and second coordinates correspond to indices of the strings X and Y, respectively. Define an *A-move* to be incrementing the first coordinate, a, by 1, and a *B-move* to be incrementing of the second coordinate, b, by 1.

1. If $a = |X|$ or $b = |Y|$, stop.
2. Do one of the following
   (a) If the last s moves were A-moves, make a B-move.
   (b) Else if the last t moves were B-moves, make an A-move.
   (c) Else if (a, b) is type-A, make an A-move.
   (d) Else, (a, b) must be type-B, in which case make a B-move.
3. Repeat from step 1.

Note that at the end of this algorithm, exactly one of $a = |X|$ and $b = |Y|$ is true. We say this matching is a *success* if we ended with $a = |X|$, otherwise it is a *failure*.

Definition 4.11. Note also that the matching is uniquely determined by $X, Y, S_1, \ldots, S_{|X|}, s, t$. If this matching is a success, we say X is $(s, t, S_1, \ldots, S_{|X|})$-*matachable* (or has a $(s, t, S_1, \ldots, S_{|X|})$-*matching*) in Y, denoted

$$X \prec_{(s,t,S_1,\ldots,S_{|X|})} Y. \tag{20}$$

Proposition 4.12. *Let $S_1, \ldots, S_{\delta n}$ be subsets of $[K]$ of size exactly $\lambda - 1$. Let $\tau = \tau_1 \frown \cdots \frown \tau_{\delta n} \in \mathcal{D}(nL)$ be a deletion pattern such that for all i, $\tau_i \in \mathcal{D}(L)$ is $(\lambda - 1)$-admissible and in particular preserves g_j for all $j \notin S_i$. Suppose we have $X \in [K]^{\delta n}$ and $Y \in [K]^n$ such that $\tau(\psi(X)) \sqsubseteq \psi(Y)$ (recall $\sqsubseteq$ is the subsequence relation). Then $X \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y$.*

Proof. Let $s = 2^\lambda, t = \sqrt{R}$. Run the matching algorithm defined above to obtain a matching of X and Y. Let $\mathcal{M}$ be the set of all (a, b) reached by some step of the algorithm. We wish to show this matching is a success, i.e. that there exists some b such that $(|X|, b) \in \mathcal{M}$, or, equivalently, there does not exist a such that $(a, |Y|) \in \mathcal{M}$.

Since $\tau(\psi(X)) \sqsubseteq \psi(Y)$, we can find $\tau(\psi(X))$ as a subsequence of $\psi(Y)$ by greedily matching the bits of $\tau(\psi(X))$ with the bits of $\psi(Y)$. Let $\mathcal{N}$ be the set of (i, k) such that some bit of $\tau_i(\psi(X_i))$ is matched with some bit in $\psi(Y_k)$. We first establish some basic facts about $\mathcal{M}, \mathcal{N}$.

Fact 4.13. *1.* $(|X|, |Y|) \notin \mathcal{M}$.

2. $\mathcal{M}, \mathcal{N} \subseteq \{1, \ldots, |X|\} \times \{1, \ldots, |Y|\}$.

3. For all $a^* \in \{1, \ldots, |X|\}, b^* \in \{1, \ldots, |Y|\}$, we have $\{b : (a^*, b) \in \mathcal{M}\}$ and $\{a : (a, b^*) \in \mathcal{M}\}$ are intervals of consecutive integers of lengths at most $t + 1$ and $s + 1$, respectively.

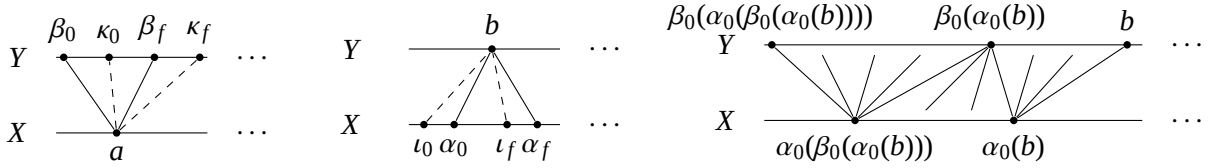


Figure 1: Illustrations of $\alpha_0(b), \alpha_f(b), \beta_0(a), \beta_f(a), \iota_0(b), \iota_f(b), \kappa_0(a), \kappa_f(a)$

4. *For all $i^* \in \{1, \dots, |X|\}, k^* \in \{1, \dots, |Y|\}$, we have $\{k : (i^*, k) \in \mathcal{N}\}$ and $\{i : (i, k^*) \in \mathcal{N}\}$ are intervals of consecutive integers.*

5. *Let $\leq_{\mathcal{M}}$ be a relation on $\mathcal{M}$ such that $(a, b) \leq_{\mathcal{M}} (a', b')$ iff $a \leq a'$ and $b \leq b'$. Then $\mathcal{M}$ is totally ordered under $\leq_{\mathcal{M}}$. As such, we can define $\text{next}_{\mathcal{M}}(a, b)$ and $\text{prev}_{\mathcal{M}}(a, b)$ to be the next larger and next smaller element after (a, b) under $\leq_{\mathcal{M}}$, respectively. Then $\text{next}_{\mathcal{M}}(a, b) \in \{(a + 1, b), (a, b + 1)\}$ and $\text{prev}_{\mathcal{M}}(a, b) \in \{(a - 1, b), (a, b - 1)\}$.*

6. *Let $\leq_{\mathcal{N}}$ be a relation on $\mathcal{N}$ such that $(i, k) \leq_{\mathcal{N}} (i', k')$ iff $i \leq i'$ and $k \leq k'$. Then $\mathcal{N}$ is totally ordered under $\leq_{\mathcal{N}}$. As such, we can define $\text{next}_{\mathcal{N}}(i, k)$ and $\text{prev}_{\mathcal{N}}(i, k)$ to be the next larger and next smaller element after (i, k) under $\leq_{\mathcal{N}}$, respectively. Then $\text{next}_{\mathcal{N}}(i, k) \in \{(i + 1, k), (i, k + 1), (i + 1, k + 1)\}$ and $\text{prev}_{\mathcal{N}}(i, k) \in \{(i - 1, k), (i, k - 1), (i - 1, k - 1)\}$.*

7. *If $(a, b), (a', b') \in \mathcal{M}$, we never have both $a < a'$ and $b' < b$.*

8. *If $(i, k), (i', k') \in \mathcal{N}$, we never have both $i < i'$ and $k' < k$.* □

For $a \in \{1, \dots, |X|\}$ and $b \in \{1, \dots, |Y|\}$, define

$$\begin{aligned} \alpha_0(b) &= \min\{a : (a, b) \in \mathcal{M}\} & \alpha_f(b) &= \max\{a : (a, b) \in \mathcal{M}\} \\ \beta_0(a) &= \min\{b : (a, b) \in \mathcal{M}\} & \beta_f(a) &= \max\{b : (a, b) \in \mathcal{M}\} \\ \iota_0(b) &= \min\{i : (i, b) \in \mathcal{N}\} & \iota_f(b) &= \max\{i : (i, b) \in \mathcal{N}\} \\ \kappa_0(a) &= \min\{k : (a, k) \in \mathcal{N}\} & \kappa_f(a) &= \max\{k : (a, k) \in \mathcal{N}\}. \end{aligned} \tag{21}$$

See Figure 1 for illustrations of the behavior of these eight functions. We first establish a few facts about the notation $\alpha_0, \beta_0, \dots$ that are helpful for developing intuition and are also useful later. These proofs are more involved than Fact 4.13 and are provided.

Lemma 4.14. 1. *For all $a \in \{1, \dots, |X|\}$, if $\kappa_f(a) - \kappa_0(a) < \sqrt{R}$, then there exists $b' \in [\kappa_0(a), \kappa_f(a)]$ such that (a, b') is type-A.*

2. *For all $a \in \{1, \dots, |X|\}$ we have $\beta_f(a) - \beta_0(a) \leq \sqrt{R}$ and for all $\beta_0(a) \leq b' < \beta_f(a)$ we have (a, b') is type-B Furthermore, if $\beta_f(a) - \beta_0(a) < \sqrt{R}$, then $(a, \beta_f(a))$ is type-A.*

3. *For all $b \in \{1, \dots, |Y|\}$, we have $\iota_f(b) - \iota_0(b) \leq 2^\lambda$.*

4. *For all $b \in \{1, \dots, |Y|\}$, we have (i', b) is type-A for all $i' \in [\iota_0(b) + 1, \iota_f(b) - 1]$.*

5. *For all $b \in \{1, \dots, |Y|\}$, we have $\alpha_f(b) - \alpha_0(b) \leq 2^\lambda$ and for all $a' \in [\alpha_0(b), \alpha_f(b) - 1]$ we have (a', b) is type-A. Furthermore, if $\alpha_f(b) - \alpha_0(b) < 2^\lambda$, then $(\alpha_f(b), b)$ is type-B.*

Proof. Parts 2 and 5 follow from Definition 4.10.

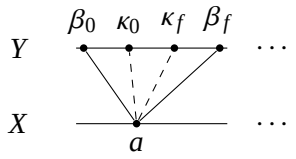


Figure 2: Step 2, Assume FSOC $\beta_0(a) \le \kappa_0(a) \le \kappa_f(a) < \beta_f(a)$

For part 1, suppose for contradiction that (a, b') is type-B for all $\kappa_0(a) \le b' \le \kappa_f(a)$. Then $X_a \notin S_a$ and $X_a < Y_{b'}$ for all such b'. This means $\tau_a(\psi(X_a))$ has at least $2R^{K+\frac{1}{2}-X_a}$ runs while $\psi(Y_{b'})$ has at most $2R^{K+1-(X_a+1)}$ runs for $\kappa_0(a) \le b' \le \kappa_f(a)$. On the other hand, we have

$$\tau_a(\psi(X_a)) \sqsubseteq \psi\left(Y_{\kappa_0(a)} \dots Y_{\kappa_f(a)}\right). \tag{22}$$

As $\kappa_f(a) - \kappa_0(a) < \sqrt{R}$ this means the right side of (22) has less than $\sqrt{R} \cdot 2R^{K-X_a} = 2R^{K+\frac{1}{2}-X_a}$ runs while the left side has at least that many runs, a contradiction.

For part 3, suppose for contradiction that $\iota_f(b) - \iota_0(b) - 1 \ge 2^\lambda$. Since $\psi(Y_b)$ contains $\prod_{i'=\iota_0(b)+1}^{\iota_f(b)-1} \tau_{i'}(\psi(X_{i'}))$ as a strict subsequence ($\psi(Y_b)$ additionally contains at least one bit from each of $\tau_{\iota_0}(\psi(X_{\iota_0}))$ and $\tau_{\iota_f}(\psi(X_{\iota_f}))$), we have

$$L + 2 \;\le\; (\iota_f(b) - \iota_0(b) - 1) \cdot \frac{L}{2^\lambda} + 2 \;\le\; \left(\sum_{i'=\iota_0+1}^{\iota_f-1} |\tau_{i'}(\psi(X_{i'}))| \right) + 2 \;\le\; |\psi(Y_b)| \;=\; L, \tag{23}$$

a contradiction.

For part 4, suppose for contradiction that (i', b) is type-B for some $\iota_0(b) < i' < \iota_f(b)$. Thus $X_{i'} \notin S_{i'}$ and $X_{i'} < Y_b$. In particular, $\tau_{i'}(\psi(X_{i'}))$ has at least $2R^{K+\frac{1}{2}-X_{i'}}$ runs, which is more than the at-most-$2R^{K-X_{i'}}$ runs of $\psi(Y_b)$. However, $\iota_0(b) < i' < \iota_f(b)$, so $(i', b) \in \mathcal{N}$ and in particular $\tau_{i'}(\psi(X_{i'})) \sqsubseteq \psi(Y_b)$, which is a contradiction. Note that $i' = \iota_0(b)$ and $i' = \iota_f(b)$ do not guarantee a contradiction because for such i', some bits of $\tau_{i'}(\psi(X_{i'}))$ might be matched with other inner codewords in $\psi(Y)$. □

The following definition of proper indices is introduced for convenience. Intuitively, indices of Y are Y-proper if the bit-matching $\mathcal{N}$ consumes corresponding indices of X "slower" than in the algorithmic matching $\mathcal{M}$, and indices of X are X-proper if the bit-matching $\mathcal{N}$ consumes corresponding indices of Y "faster" than in the algorithmic matching $\mathcal{M}$.

Definition 4.15. We say an index in $a \in \{1, \dots, |X|\}$ is *X-proper* if

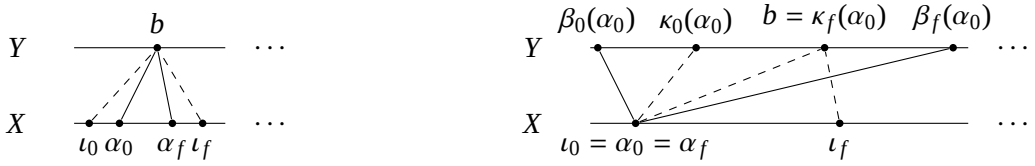
$$\beta_0(a) \;\le\; \kappa_0(a), \qquad \beta_f(a) \;\le\; \kappa_f(a) \tag{24}$$

and we say an index $b \in \{1, \dots, |Y|\}$ is *Y-proper* if

$$\iota_0(b) \;\le\; \alpha_0(b), \qquad \iota_f(b) \;\le\; \alpha_f(b). \tag{25}$$

Claim 4.16. *For all $a \in \{1, \dots, |X|\}$ and all $b \in \{1, \dots, |Y|\}$, a is X-proper and b is Y-proper.*

Remark 4.17. First we illustrate how the Claim 4.16 implies Proposition 4.12. Suppose for contradiction that Proposition 4.12 is false. Then there exists some a such that $(a, |Y|) \in \mathcal{M}$ and for all b we have $(|X|, b) \notin \mathcal{M}$. In particular, $\alpha_f(b) < |X|$ for all $b \in \{1, \dots, |Y|\}$. By the claim, $\iota_f(|Y|) \le \alpha_f(|Y|) < |X|$, implying that no bits from $\tau_{|X|}(\psi(X_{|X|}))$ are matched with bits of $\psi(Y)$, a contradiction of $\tau(\psi(X)) \sqsubseteq \psi(Y)$.



(a) Assume FSOC $\iota_0(b) \le \alpha_0(b) \le \alpha_f(b) < \iota_f(b)$

(b) $\iota_0(b) = \alpha_0(b) = \alpha_f(b) < \iota_f(b)$

Figure 3: Step 3

(a) $b < \kappa_f(\alpha_0(b))$

(b) $b = \kappa_f(\alpha_0(b))$

Figure 4: Step 4

Proof. **Step 1.** First, note that, as $(1,1) \in \mathcal{M}, \mathcal{N}$, we have $\alpha_0(1) = \beta_0(1) = \iota_0(1) = \kappa_0(1) = 1$.

Step 2. Next, we show that for all a, if $\beta_0(a) \le \kappa_0(a)$ then a is X-proper. That is, we show $\beta_f(a) \le \kappa_f(a)$. Suppose for contradiction we have $\kappa_f(a) < \beta_f(a)$ so that $\beta_0(a) \le \kappa_0(a) \le \kappa_f(a) < \beta_f(a)$ (see Figure 2). By Lemma 4.14 part 2, we have $\beta_f(a) - \beta_0(a) \le \sqrt{R}$ and (a, b') is type-B for all $b' \in [\beta_0(a), \beta_f(a) - 1]$. In particular, (a, b') is type-B for all $b' \in [\kappa_0(a), \kappa_f(a)]$. As $\kappa_f(a) - \kappa_0(a) < \beta_f(a) - \beta_0(a)$, we have $\kappa_f(a) - \kappa_0(a) < \sqrt{R}$, so we can apply Lemma 4.14 part 1 to obtain that (a, b') is type-A for some $b' \in [\kappa_0(a), \kappa_f(a)]$. This is a contradiction as all such b' must be type-B.

Step 3. Next, we show that for all b, if $\iota_0(b) \le \alpha_0(b)$ and $\alpha_0(b)$ is X-proper, then b is Y-proper. That is, we show $\iota_f(b) \le \alpha_f(b)$. Suppose for contradiction that $\alpha_f(b) < \iota_f(b)$ so that $\iota_0(b) \le \alpha_0(b) \le \alpha_f(b) < \iota_f(b)$ (see Figure 3a). We have $\alpha_f - \alpha_0 < \iota_f - \iota_0 \le 2^\lambda$ by Lemma 4.14 part 3. Thus, by Lemma 4.14 part 5, $(\alpha_f(b), b)$ is type-B. By Lemma 4.14 part 4, (i', b) is type-A for $i' \in [\iota_0(b) + 1, \iota_f(b) - 1]$. Since $\alpha_f(b) \in [\iota_0(b), \iota_f(b) - 1]$ we must have $\alpha_f(b) = \iota_0(b)$, so $\iota_0(b) = \alpha_0(b) = \alpha_f(b)$ (See Figure 3b). By definition of $\alpha_f(b)$, we have $\text{next}_{\mathcal{M}}(\alpha_f(b), b) = (\alpha_f(b), b + 1)$ so $\beta_f(\alpha_f(b)) \ge b + 1$. However, since we assumed $\alpha_f < \iota_f$, we have $\text{next}_{\mathcal{N}}(\alpha_f(b), b) = (\alpha_f(b) + 1, b)$, so $\kappa_f(\alpha_f(b)) = b$. Thus

$$\beta_f(\alpha_0(b)) = \beta_f(\alpha_f(b)) \ge b + 1 > b = \kappa_f(\alpha_f(b)) = \kappa_f(\alpha_0(b)). \tag{26}$$

On the other hand, $\beta_f(\alpha_0(b)) \le \kappa_f(\alpha_0(b))$ by assumption that $\alpha_0(b)$ is X-proper, which is a contradiction. This covers all possible cases, completing Step 3.

Step 4. We prove that, for all $b \in \{1, \dots, |Y|\}$, if $\alpha_0(b)$ is X-proper, then b is Y-proper. By Step 3, it suffices to prove $\iota_0(b) \le \alpha_0(b)$. Suppose for contradiction that $\alpha_0(b) < \iota_0(b)$. Since $(\alpha_0(b), b) \in \mathcal{M}$, we have $b \le \beta_f(\alpha_0(b))$. By assumption, $\alpha_0(b)$ is X-proper, so $\beta_f(\alpha_0(b)) \le \kappa_f(\alpha_0(b))$, which means $b \le \kappa_f(\alpha_0(b))$. If $b < \kappa_f(\alpha_0)$, then we have $(\alpha_0(b), \kappa_f(\alpha_0(b))) \in \mathcal{N}$. However, $(\iota_0(b), b) \in \mathcal{N}$, contradicting Fact 4.13 part 8. Thus, $\kappa_f(\alpha_0(b)) = b$. But then $(\alpha_0(b), b) = (\alpha_0(b), \kappa_f(\alpha_0(b))) \in \mathcal{N}$ with $\alpha_0(b) < \iota_0(b)$, contradicting the minimality of $\iota_0(b)$.

Step 5. By the same argument as Step 4, we have that, for all $a \in \{1, \dots, |X|\}$, if $\beta_0(a)$ is Y-proper, then a is X-proper.

Step 6. We prove by strong induction that for pairs (a, b), ordered by $<_{\mathcal{M}}$, we have a is X-proper and b

is Y-proper. Combining Steps 1,2, and 3, we have

$$\beta_0(1) \le \kappa_0(1), \qquad \beta_f(1) \le \kappa_f(1), \qquad \iota_0(1) \le \alpha_0(1), \qquad \iota_f(1) \le \alpha_f(1), \tag{27}$$

where the first and third inequalities are actually equalities arising from Step 1, the second inequality is established by Step 2, and the fourth is established by Step 3. Thus, 1 is both X-proper and Y-proper.

Now suppose we have some pair $(a,b) \in \mathcal{M}$ with $(1,1) <_{\mathcal{M}} (a,b)$ and (22) has been established for all smaller pairs. If $\text{prev}_{\mathcal{M}}(a,b) = (a-1,b)$, then by the inductive hypothesis, we have b is Y-proper. However, as $(a-1,b) \in \mathcal{M}$, we have $(a,b-1) \notin \mathcal{M}$, so we have $\beta_0(a) = b$. Thus $\beta_0(a)$ is Y-proper, so a is X-proper by Step 5. Similarly, if $\text{prev}_{\mathcal{M}}(a,b) = (a,b-1)$, then by the inductive hypothesis, we have a is X-proper. Thus $\alpha_0(b) = a$ is X-proper, so b is Y-proper by Step 4. This completes the proof of Claim 4.16, proving Proposition 4.12. □

The following proposition is the key result of this subsection. Following our approach, it should be possible to prove this proposition for any choice of $S_1, \dots, S_{\delta n}$, not just $[\lambda-1], \dots, [\lambda-1]$. However, because of Lemma 4.23, which tells us that $[\lambda-1], \dots, [\lambda-1]$ is the "worst" possible choice of $S_1, \dots, S_{\delta n}$, it suffices to prove the proposition as stated.

Proposition 4.18. *There exists a constant $\beta > 0$ such that for any fixed deletion pattern $\sigma \in \mathcal{D}(n, (1-\delta)n)$ we have*

$$\mathop{\mathbf{Pr}}_{X,Y \sim U([K]^n)}[\sigma(X) \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y] < 2^{-\beta n}. \tag{28}$$

Proof. Let $s = 2^\lambda$. With hindsight, let $\beta = \frac{\log K}{16R}$. It suffices to prove

$$\mathop{\mathbf{Pr}}_{\substack{X \sim U([K]^{\delta n}) \\ Y \sim U([K]^n)}} \left[X \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \dots, [\lambda-1])} Y\right] < 2^{-\beta n}. \tag{29}$$

This suffices because for any σ, the distribution of $\sigma(X)$ for $X \sim U([K]^n)$ is the same as $X \sim U([K]^{\delta n})$.

Let $X_1, \dots, X_{\delta n}, Y_1, Y_2, \dots$ be independently chosen from $[K]$. Let $X = X_1 \dots X_{\delta n}$, $Y = Y_1 \dots Y_n$, and $Y_\infty = Y_1 Y_2 \dots$ so that $X \sim U([K]^{\delta n})$ and $Y \sim U([K]^n)$. Construct a $(2^\lambda, \sqrt{R}, S_1, \dots, S_{\delta n})$-matching of X in Y_∞. Note that, as Y_∞ is an infinite random string, $\mathcal{M}$ succeeds almost surely, i.e. ends in $(|X|, b)$ for some integer b.

Let $\mathcal{M}$ be the set of all reached states (a,b) in the matching, and let $\alpha_f(b) = \max\{a : (a,b) \in \mathcal{M}\}$ and $\beta_f(a) = \max\{b : (a,b) \in \mathcal{M}\}$ as in Proposition 4.12. Let $A_0 = 1, B_0 = 1$. For $i \ge 1$, set $A_i = \alpha_f(B_{i-1})$ and $B_i = \beta_f(A_{i-1})$. As $A_i - A_{i-1} \le s$ for all $i \ge 1$, we have A_i is well defined for $i \le \delta n/s - 1$. By the definition of matching, $\mathcal{M}'$ succeeds if and only if $\mathcal{M}$ succeeds and the final position $(|X|, \beta_f(|X|))$ satisfies $\beta_f(|X|) < |Y|$. It thus suffices to prove

$$\mathbf{Pr}\left[B_{\lfloor \delta n/s \rfloor - 1} < |Y|\right] < 2^{-\beta n}. \tag{30}$$

The key idea of this proof is that the B_i's grow much faster than the A_i's, so that the B_i's "run out of indices in Y" faster than the A_i's "run out of indices in X", even though Y is longer than X. In particular, by definition of a matching, we have $A_{i+1} - A_i \le s = 2^\lambda$, but, on the other hand, we show that $B_{i+1} - B_i$ is, in expectation, $\Omega(\log K)$ (see Figure 5).

We have two technical lemmas. The proofs are straightforward, and we include them in Appendix C for completness.

Lemma 4.19. *Let J be chosen uniformly from $[K]$. Let D be a random variable that is 1 if $J \in [\lambda-1]$ and, conditioned on a fixed $J \ge \lambda$, is distributed as* $\min(\text{Geometric}(J/K), \sqrt{R})$. *Then $\mathbb{E}[D] \ge (\log K)/4$.*

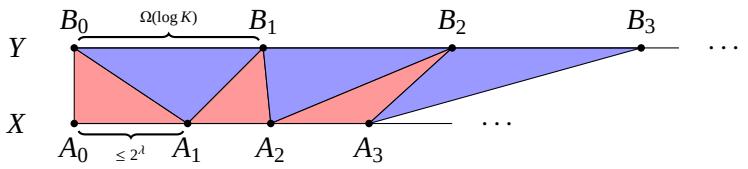


Figure 5: A_i's and B_i's behavior

Lemma 4.20. *Let $\lambda' \in [\lambda, K]$ and let J be chosen uniformly from $\{\lambda, \lambda+1, \ldots, \lambda'\}$. Let D be the random variable that, conditioned on a fixed J, is distributed as* $\min(\text{Geometric}(J/K), \sqrt{R})$. *Then $\mathbb{E}[D] \geq (\log K)/4$.*

Claim 4.21. *Let $i \geq 1$. For any fixed $A_0, \ldots, A_i, B_0, \ldots, B_i, X_1, \ldots, X_{A_i}, Y_1, \ldots, Y_{B_i}$, we have*

$$\mathbb{E}\left[B_{i+1} - B_i | A_1, \ldots, A_i, B_1, \ldots, B_i, X_1, \ldots, X_{A_i}, Y_1, \ldots, Y_{B_i}\right] > \frac{\log K}{4}. \tag{31}$$

Proof. It suffices to prove that if we additionally condition on $A_{i+1} - A_i < s$, then the expectation is at least $(\log K)/4$, and that the same is true if we condition on $A_{i+1} - A_i = s$.

First, note that, for $1 \leq b < \sqrt{R}$, we have $B_{i+1} = B_i + b$ if and only if $(A_{i+1}, B_i + j)$ is type-B for $j \in \{1, \ldots, B_i + b - 1\}$ and $(A_{i+1}, B_i + b)$ is type-A. If no such b exists, we have $B_{i+1} - B_i = \sqrt{R}$. Thus, conditioned on fixed $A_0, \ldots, A_{i+1}, B_0, \ldots, B_i, X_1, \ldots, X_{A_{i+1}}, Y_1, \ldots, Y_{B_i}$, we have $B_{i+1} - B_i$ is distributed as $\min(\text{Geometric}(X_{A_{i+1}}/K), \sqrt{R})$.

Suppose we condition on $A_{i+1} - A_i = s$. This is equivalent to saying $(A_i + \ell, B_i)$ is type-A for $\ell = 1, \ldots, s-1$. However, this assertion depends only on $X_{A_i}, X_{A_i+1}, \ldots, X_{A_i+s-1}$, which are independent of $X_{A_{i+1}}$, so we have $X_{A_{i+1}}$ is still distributed uniformly on $[K]$. If $X_{A_{i+1}} \in [\lambda - 1]$, then $B_{i+1} - B_i = 1$, otherwise $B_{i+1} - B_i$ is distributed as $\min(\text{Geometric}(X_{A_{i+1}}/K), \sqrt{R})$, so by Lemma 4.19 on $D = B_{i+1} - B_i$, we have

$$\mathbb{E}[B_{i+1} - B_i | A_{i+1} - A_i = s, A_1, \ldots, A_i, B_1, \ldots, B_i, X_1, \ldots, X_{A_i}, Y_1, \ldots, Y_{B_i}] > \frac{\log K}{4}. \tag{32}$$

Suppose we condition on $A_{i+1} - A_i = s' < s$. This is equivalent to saying $(A_i + \ell, B_i)$ is type-A for $\ell = 1, \ldots, s' - 1$ and $(A_i + s', B_i)$ is type-B. This implies $X_{A_{i+1}} \geq \lambda$ (i.e. $X_{A_{i+1}} \notin [\lambda - 1]$) and $X_{A_{i+1}} < Y_{B_i}$, Since Y_{B_i} is fixed and, without any conditioning, $X_{A_{i+1}}$ is distributed uniformly in $[K]$, we have $X_{A_{i+1}}$ is distributed uniformly on $[\lambda, \ldots, Y_{B_i}]$. By the previous argument, we have $B_{i+1} - B_i$ is distributed as $\min(\text{Geometric}(X_{A_{i+1}}/K), \sqrt{R})$, so by Lemma 4.20 on $\lambda' = Y_{B_i}$ and $D = B_{i+1} - B_i$, we have

$$\mathbb{E}[B_{i+1} - B_i | A_{i+1} - A_i = s' < s, A_1, \ldots, A_i, B_1, \ldots, B_i, X_1, \ldots, X_{A_i}, Y_1, \ldots, Y_{B_i}] > \frac{\log K}{4}. \tag{33}$$

□

Corollary 4.22. *We have, for any fixed $B_1, \ldots, B_i$,*

$$\mathbb{E}[B_{i+1} - B_i | B_1, \ldots, B_i] > \frac{1}{4} \log K. \tag{34}$$

Continuing the proof of Proposition 4.18, let $\alpha = \frac{1}{4} \log K$ so that

$$\alpha(k-1) \;>\; \frac{1}{4} \log K \cdot \left(\frac{\delta n}{2^\lambda} - 1\right) \;>\; 2n. \tag{35}$$

Define the random variable $B'_i := B_i - i \cdot \alpha$. As B'_i and B_i uniquely determine each other, we have, by Corollary 4.22,

$$\mathbb{E}[B'_{i+1} - B'_i | B'_1, \ldots, B'_i] = \mathbb{E}[B_{i+1} - B_i | B_1, \ldots, B_i] - \alpha > 0. \tag{36}$$

Thus, B'_i form a submartingale with $|B'_{i+1} - B'_i| < \sqrt{R}$, so by Azuma's Inequality (Lemma 3.2), we have

$$\mathbf{Pr}\left[B'_k - B'_1 \le -\frac{\alpha(k-1)}{2}\right] \le \exp\left(-\frac{(\alpha(k-1)/2)^2}{2(k-1)\cdot(\sqrt{R})^2}\right) = \exp\left(-\frac{\alpha^2(k-1)}{8R}\right) < \exp\left(-\frac{n\log K}{16R}\right). \tag{37}$$

Combining with (35) gives

$$\mathbf{Pr}[B_k \le n] \le \mathbf{Pr}[B'_k - B'_1 \le -n] \le \mathbf{Pr}\left[B'_k - B'_1 \le -\frac{\alpha(k-1)}{2}\right] < \exp\left(-\frac{n\log K}{16R}\right). \tag{38}$$

We conclude

$$\mathop{\mathbf{Pr}}_{\substack{X\sim U([K]^{\delta n})\\ Y\sim U([K]^n)}}[X \prec_{(\sqrt{R},2^\lambda,[\lambda-1],\ldots,[\lambda-1])} Y] \le \mathbf{Pr}[m_{\delta n} \le n] \le \mathbf{Pr}[B_k \le n] < \exp\left(-\frac{n\log K}{16R}\right). \tag{39}$$

As $\exp(-\frac{n\log K}{16R}) < 2^{-\frac{n\log K}{16R}}$, we have (29) is true, completing the proof of Proposition 4.18. □

To conclude this section, we formalize the intuition that the "worst possible deletion pattern", that is, the deletion pattern that makes decoding most difficult in some sense, is the deletion pattern such that each $(\lambda-1)$-admissible inner deletion pattern corrupts $g_1, \ldots, g_{\lambda-1}$. This fact is intuitive, as it is hard, for example, to find g_1 as a subsequence of a random $\psi(Y)$ because g_1 has many runs. Thus if our deletion pattern τ corrupts the inner codewords with the most runs, there is a greater probability that over random X, Y we have $\tau(\psi(X))$ is a subsequence of $\psi(Y)$. However, note that, to make a clean assertion, we continue to argue using the matching relation rather than subsequence relation.

Lemma 4.23. *Let $r, s \in \mathbb{N}$ and $S_1, \ldots, S_{\delta n}$ be subsets of $[K]$ of size exactly $\lambda - 1$. Then for all $Y \in [K]^n$ we have*

$$\#\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y\right\} \le \#\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \ldots, [\lambda-1])} Y\right\} \tag{40}$$

Proof. We start with a claim.

Claim 4.24. *For every set $A \subseteq [K]$ with size exactly $\lambda - 1$, there exists a bijection $h_A : [K] \to [K]$ such that $h_A(x) \in [\lambda - 1]$ for $x \in A$ and $h_A(x) \ge x$ for $x \notin A$.*

Proof. Pair each element $x \in A \setminus [\lambda - 1]$ with an element $y \in [\lambda - 1] \setminus A$ arbitrarily and for each pair (x, y) set $h_A(x) = y, h_A(y) = x$. Note this always gives $x > y$. This is possible as $A \setminus [\lambda - 1]$ and $[\lambda - 1] \setminus A$ have the same size. Then set $h_A(x) = x$ for all other x. It is easy to check this function satisfies $h_A(x) \ge x$ for $x \notin A$. □

Fix $Y \in [K]^n$. For all i, let $h_i : S_i \to [\lambda - 1]$ be a bijection such that $h_i(x) \in [\lambda - 1]$ for $x \in S_i$ and $f(x) \ge x$ for all other x. This exists by Claim 4.24. Let $h : [K]^{\delta n} \to [K]^{\delta n}$ be such that $h(X_1, \ldots, X_{\delta n}) = h_1(X_1)h_2(X_2)\cdots h_{\delta n}(X_{\delta n})$. Since each of $h_1, \ldots, h_{\delta n}$ are bijections, h is a bijection as well.

Let X be such that $X \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y$. We claim $h(X) \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \ldots, [\lambda-1])} Y$. Let $\mathcal{M}$ be a $(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})$-matching of X in Y and let $\mathcal{M}'$ be a $(2^\lambda, \sqrt{R}, [\lambda-1], \ldots, [\lambda-1])$-matching of $h(X)$ in Y. We know $\mathcal{M}$ is a successful matching, and we wish to show $\mathcal{M}'$ is a successful matching.

If (a, b) is type-A with respect to $X, Y, S_1, \ldots, S_{|X|}$, then $Y_b \le X_a$ or $X_a \in S_a$, so, by definition of h, either $h(X_a) \in [\lambda - 1]$ or $Y_b \le X_a \le h(X_a)$, so (a, b) is type-A with respect to $h(X), Y, [\lambda - 1], \ldots, [\lambda - 1]$. Let

$\ell \geq 0$ be an integer and let $(a, b) \in \mathcal{M}$ and $(a', b') \in \mathcal{M}'$ be the state of the matchings after the ℓth step of the matchings of X in Y and $h(X)$ in Y, respectively. It is easy to see by induction that $a \leq a'$ and $b' \leq b$; if $a < a'$ then after one move we still have $a \leq a'$, and if $a = a'$ and $\mathcal{M}$s next move is an A-move, then $\mathcal{M}'$s next move must also be an A-move. Since $\mathcal{M}$ succeeds, we have $(|X|, b) \in \mathcal{M}$ for some $b < |Y|$, so we conclude that $(|X|, b') \in \mathcal{M}'$ for some $b' \leq b < |Y|$. Thus $\mathcal{M}'$ succeeds as well.

Since h is a bijection such that

$$h\left(\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y\right\}\right) \subseteq \left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \ldots, [\lambda-1])} Y\right\}, \tag{41}$$

we have

$$\#\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y\right\} \leq \#\left\{X \in [K]^{\delta n} : X \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \ldots, [\lambda-1])} Y\right\} \tag{42}$$

as desired. □

4.4 Technical combinatorial lemmas

Lemma 4.25. *Let n be a positive integer and suppose we have $0 < \beta < 1$. Suppose that $\gamma = \beta/3$, $M = 2^{\gamma n}$, and $\epsilon \geq 2^{-(\gamma - o(1))n/2}$. Suppose G is a graph on $N = K^{(1-o(1))n}$ vertices such that each vertex has degree at most $d = Nr$ for some $r = r(n) = 2^{-(\beta - o(1))n}$. Suppose C_{out} is chosen as a random subset of $V(G)$ by including each vertex of $V(G)$ in C_{out} with probability M/N. Then for sufficiently large n, we have $\mathbf{Pr}_{C_{out}}[|E(G \upharpoonright C_{out})| > \epsilon M] < 2^{-\omega(n)}$.*

Remark 4.26. Essentially this lemma is saying that when the edge density is extremely small, around $2^{-\beta n}$, then for all but an extremely small set of choices for C_{out}, C_{out} is extremely sparse.

Proof. Let $E = E(G)$. We know E satisfies $|E| \leq dN/2 < N^2 r$.

Enumerate the edges $1, \ldots, |E|$ arbitrarily. Let $Y_1, \ldots, Y_{|E|}$ be Bernoulli random variables denoting whether the ith edge is in C_{out}. Let $Z = Y_1 + \cdots + Y_{|E|}$. We would like to show

$$\mathop{\mathbf{Pr}}_{C_{out}}[Z > \epsilon M] < 2^{-n}. \tag{43}$$

We do this by computing a sufficiently large moment of Z and using

$$\mathop{\mathbf{Pr}}_{C_{out}}[Z > \epsilon M] \leq \frac{\mathbb{E}[Z^k]}{(\epsilon M)^k}. \tag{44}$$

For a tuple of (not necessarily distinct) edges $(e_1, \ldots, e_k)$, denote $V(e_1, \ldots, e_k)$ to be the set of vertices on at least one of the edges $e_1, \ldots, e_k$. Alternatively, we say $V(e_1, \ldots, e_k)$ is the set of vertices *covered* by edges $e_1, \ldots, e_k$. Note that for all $e_1, \ldots, e_k$, we have $2 \leq V(e_1, \ldots, e_k) \leq 2k$. Let $P_{k,\ell} = \#\{(e_1, \ldots, e_k) : |V(e_1, \ldots, e_k)| = \ell\}$ denote the number of ordered tuples of edges from E that cover exactly ℓ vertices.

Claim.

$$P_{k,\ell} < N^\ell \cdot r^{\ell/2} \cdot \ell^{2k+\ell-1} \cdot k^k \tag{45}$$

Proof. We bound $P_{k,\ell}$ by first bounding the number of sets (unordered tuples) of edges covering exactly ℓ vertices and then multiply by $k!$. We first compute the number of ways to construct a forest of trees covering ℓ vertices using only edges from E. We do this by casework on the number of connected components. Let $1 \leq c \leq \lfloor \ell/2 \rfloor$ be an integer. To construct a forest with c connected components, we first choose c disjoint edges $e_1, \ldots, e_c$. This can be done in at most $|E|^c$ ways. We have $|V(e_1, \ldots, e_c)| = 2c$. For $c + 1 \leq i \leq \ell - c$, choose a vertex $v \in V(e_1, \ldots, e_{i-1})$ and an edge vw such that $w \notin V(e_1, \ldots, e_{i-1})$. By construction, for

$c+1 \le i \le \ell - c$, we have $|V(e_1, \dots, e_i)| = i + c$. Thus the i^{th} edge can be added in at most $(i+c-1)d$ ways. The $i = \ell - c^{\text{th}}$ edge completes a forest covering ℓ vertices. Recalling that $|E| < N^2 r$ and $d = Nr$, we have that the total number of ways to construct a forest in this fashion is at most

$$|E|^c \cdot 2cd \cdot (2c+1)d \cdots (\ell-1)d \;\le\; |E|^c \cdot d^{\ell-2c} \cdot \ell^{\ell-2c} \;<\; N^\ell \cdot r^{\ell-c} \cdot \ell^{\ell-2c}. \tag{46}$$

We have used $\ell - c$ edges thus far. There are $k - \ell + c$ remaining edges. As $|V(e_1, \dots, e_k)| = \ell$, these remaining edges must connect one of the $\binom{\ell}{2}$ pairs of vertices in $V(e_1, \dots, e_{k-\ell+c})$. As $c \le \lfloor \ell/2 \rfloor < \ell$, we have $k - \ell + c < k$. The remaining edges can thus be chosen in at most $\binom{\ell}{2}^{k-\ell+c} < \ell^{2k}$ ways. Using (46) and multiplying by $k!$ we have

$$P_{k,\ell} \;<\; k! \cdot \sum_{c=1}^{\ell/2} N^\ell \cdot r^{\ell-c} \cdot \ell^{\ell-2c} \cdot \ell^{2k} \;<\; k^k \cdot (\ell/2) \cdot N^\ell \cdot r^{\ell/2} \cdot \ell^{\ell-2} \cdot \ell^{2k} \;<\; N^\ell \cdot r^{\ell/2} \cdot \ell^{2k+\ell-1} \cdot k^k. \tag{47}$$

□

Note that $M\sqrt{r} \ll 1$. With this claim, we have

$$\begin{aligned}
\mathbb{E}[Z^k] &= \mathbb{E}[(Y_1 + \cdots + Y_{|E|})^k] \\
&= \sum_{(e_1,\dots,e_k) \in \{1,\dots,|E|\}^k} \mathbb{E}[Y_1 \cdots Y_k] \\
&= \sum_{\ell=2}^{2k} \sum_{|V(e_1,\dots,e_k)|=\ell} \mathbb{E}[Y_{e_1} \cdots Y_{e_k}] \\
&= \sum_{\ell=2}^{2k} \sum_{|V(e_1,\dots,e_k)|=\ell} \left(\frac{M}{N}\right)^\ell \\
&= \sum_{\ell=2}^{2k} \left(\frac{M}{N}\right)^\ell \cdot P_{k,\ell} \\
&< \sum_{\ell=2}^{2k} \left(\frac{M}{N}\right)^\ell \cdot N^\ell \cdot r^{\ell/2} \cdot \ell^{2k+\ell-1} \cdot k^k \\
&< \sum_{\ell=2}^{2k} \ell^{2k+\ell-1} \cdot k^k \\
&< 2k \cdot (2k)^{4k-1} \cdot k^k = 2^{4k} \cdot k^{5k}.
\end{aligned} \tag{48}$$

Finally, choosing $k = \log n$, we have

$$\mathbf{Pr}[Z \ge \epsilon M] \;\le\; \frac{\mathbb{E}[Z^k]}{(\epsilon M)^k} \;<\; \frac{2^{4k} k^{5k}}{\epsilon^k M^k} \;<\; \left(\frac{16k^5}{\epsilon 2^{\gamma n}}\right)^k \;\le\; \left(\frac{16k^5}{2^{\gamma n/2}}\right)^k \;<\; 2^{-\omega(n)} \tag{49}$$

as desired. □

Lemma 4.27. *Let n be a positive integer, and suppose $0 < \beta < 1$ and $\gamma = \beta/4$. Suppose $M = 2^{\gamma n}$ and $\epsilon \ge 2^{-\gamma n/2}$. Suppose G is a directed graph on $N = K^{n(1-o(1))}$ vertices such that each vertex has outdegree at most Nr for some $r = 2^{-(\beta - o(1))n}$. Choose a subset $C_{out} \subseteq V(G)$ at random by including each vertex of $V(G)$ in C_{out} with probability M/N so that $\mathbb{E}[|C_{out}|] = M$. Then, for sufficiently large n, we have*

$$\mathop{\mathbf{Pr}}_{C_{out}}\left[\#\left\{X \in C_{out} : \exists Y \in C_{out} \text{ s.t. } \overrightarrow{YX} \in E(G)\right\} > \epsilon |C_{out}|\right] \;<\; 2^{-\omega(n)}. \tag{50}$$

Proof. As, by assumption, Y has outdegree at most $N \cdot r$ for all $Y \in V(G)$, the average indegree of G is at most $N \cdot r$. Thus at most $\epsilon/8$ fraction of all words in $V(G)$ have indegree larger than $\frac{8}{\epsilon} \cdot N \cdot r$. Call this set of vertices W. We have

$$\mathop{\mathbb{E}}_{C_{out}}\left[\#\left\{X \in C_{out} : \mathrm{indeg}_G(X) > \frac{8}{\epsilon} \cdot N \cdot r\right\}\right] \;=\; \mathbb{E}[|C_{out} \cap W|] \;\leq\; \frac{\epsilon}{8}M. \tag{51}$$

Since $|C_{out} \cap W|$ is the sum of i.i.d Bernoulli random variables, we have by Lemma 3.1

$$\mathop{\mathbf{Pr}}_{C_{out}}\left[\#\left\{X \in C_{out} : \mathrm{indeg}_G(X) > \frac{8}{\epsilon} \cdot N \cdot r\right\} > \frac{\epsilon}{4}M\right] \;=\; \mathbf{Pr}\left[|C_{out} \cap W| > \frac{\epsilon}{4}M\right] \;<\; e^{-\frac{1}{2}\cdot\frac{\epsilon}{8}\cdot M} < 2^{-\omega(n)}. \tag{52}$$

Now consider the undirected graph H on $V(G)$ such that $XY \in E(H)$ if $\overrightarrow{XY} \in E(G)$ and $Y \notin W$ or $\overrightarrow{YX} \in E(G)$ and $X \notin W$. For every vertex v, the in-edges of v in G correspond to edges in H only if the indegree is at most $\frac{8}{\epsilon} \cdot N \cdot r$, and the outdegree of v in G is always at most Nr. Therefore the degree of every vertex in H is at most $r'N$ where $r' = \left(\frac{8}{\epsilon} + 1\right) \cdot r < 2^{-(\beta-\gamma-o(1))n}$.

We are including each vertex of $V(G)$ (and thus each vertex of $V(H)$) in C_{out} independently with probability M/N. Let $\epsilon' = \epsilon/4, \beta' = \frac{3}{4}\beta, \gamma' = \gamma$ so that $\gamma' = \beta'/3, \epsilon \geq 2^{-(\gamma+o(1))n/2}$, $M = 2^{\gamma n}$, and $r' = 2^{-(\beta'-o(1))n}$. By Lemma 4.25 for $\beta', \gamma', M, \epsilon'$, and r', and H, we have for sufficiently large n that

$$\mathop{\mathbf{Pr}}_{C_{out}}\left(E(H \upharpoonright C_{out}) \;>\; \frac{\epsilon}{4}M\right) < 2^{-\omega(n)}. \tag{53}$$

Also, by Chernoff bounds, we have

$$\mathop{\mathbf{Pr}}_{C_{out}}\left[|C_{out}| \;<\; \frac{3}{4}M\right] < 2^{-\Omega(M)} \implies \mathop{\mathbf{Pr}}_{C_{out}}\left[|C_{out}| \;<\; \frac{3}{4}M\right] < 2^{-\omega(n)}. \tag{54}$$

Note that if the number of X such that $\mathrm{indeg}_{G\upharpoonright C_{out}}(X) > 0$ at least $\epsilon|C_{out}|$, that is,

$$\#\left\{X \in C_{out} : \exists Y \in C_{out}\,\text{s.t.}\,\overrightarrow{YX} \in E(G)\right\} > \epsilon|C_{out}|, \tag{55}$$

then one of the following must be false.

1. $|C_{out}| \geq \frac{3}{4}M$
2. $\#\{X \in C_{out} : \deg_{H\upharpoonright C_{out}}(X) > 0\} \leq \frac{\epsilon}{2}M$
3. $|C_{out} \cap W| \leq \frac{\epsilon}{4}M$

Indeed, suppose to the contrary all of these were true. The number of vertices in $C_{out} \cap W$ with positive indegree in $G \upharpoonright C_{out}$ is at most $|C_{out} \cap W| \leq \frac{\epsilon}{4}M$. If a vertex $X \in C_{out} \setminus W$ has positive indegree in $G \upharpoonright C_{out}$ then there exists $Y \in C_{out}$ such that $\overrightarrow{YX} \in E(G)$, so $XY \in E(H)$ by definition of H and thus $\deg_{H\upharpoonright C_{out}}(X) > 0$. Thus the number of vertices in $C_{out} \setminus W$ with positive indegree is at most $\frac{\epsilon}{2}M$ by property 2. Hence the total number of vertices in C_{out} with positive indegree in $G \upharpoonright C_{out}$ is at most $\frac{3}{4}\epsilon M \leq \epsilon|C_{out}|$.

Each of items 1, 2, and 3 is false with probability $2^{-\omega(n)}$ by (52), (53), and (54), so the probability any of them occur is at most $2^{-\omega(n)}$, as desired. □

4.5 Proof of Construction (Theorem 2.2)

Proof of Theorem 2.2. We would like to show there exists a code C and an $\epsilon = o_N(1)$ such that, for any deletion pattern τ deleting pN bits,

$$\#\{x \in C : \exists y \in C \text{ s.t. } x \neq y \text{ and } \tau(x) \leq y\} \;\leq\; \epsilon|C|. \tag{56}$$

For $Y \in [K]^n$, define

$$f(Y) \;=\; \mathop{\mathbf{Pr}}_{Z \sim U([K]^{\delta n})}\left[Z \prec_{(2^\lambda,\, \sqrt{R}, [\lambda-1], \ldots, [\lambda-1])} Y\right]. \tag{57}$$

Let $\sigma \in \mathcal{D}(n, (1-\delta)n)$ be a deletion pattern for our outer code $[K]^n$. Let $S_1, \ldots, S_{\delta n}$ be subsets of $[K]$ of size exactly $\lambda - 1$. Let $G^*_{\sigma, S_1, \ldots, S_{\delta n}}$ be the graph on the vertex set $[K]^n$ such that $\overrightarrow{YX}$ is an edge if and only if $\sigma(X) \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y$. Note that for all $\sigma \in \mathcal{D}(n, (1-\delta)n)$ and all $Y \in [K]^n$ we have, by Lemma 4.23,

$$\begin{aligned}
\#\{X \in [K]^n : \sigma(X) \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y\} \;&=\; K^{(1-\delta)n} \cdot \#\{Z \in [K]^{\delta n} : Z \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y\} \\
&\leq\; K^{(1-\delta)n} \cdot \#\{Z \in [K]^{\delta n} : Z \prec_{(2^\lambda, \sqrt{R}, [\lambda-1], \ldots, [\lambda-1])} Y\} \\
&=\; K^n \cdot f(Y).
\end{aligned} \tag{58}$$

In the graph language, this means every $Y \in [K]^n$ has outdegree at most $K^n \cdot f(Y)$ in every $G^*_{\sigma, S_1, \ldots, S_{\delta n}}$.

We remove all Y with large $f(Y)$. Let $\beta = \log K/16R$. By Proposition 4.18, we know the average value of $f(Y)$ is $2^{-\beta n}$. Hence at most $K^n/2$ such Y satisfy $f(Y) \geq 2 \cdot 2^{-\beta n}$. These are the *easily disguised* candidate codewords mentioned in §2. There is thus a set $W \subseteq [K]^n$ such that $|W| = K^n/2$ and each $Y \in W$ satisfies $f(Y) < 2 \cdot 2^{-\beta n} = 2^{-(\beta - o(1))n}$. For all $\sigma \in \mathcal{D}(n, (1-\delta)n), S_1, \ldots, S_{\delta n}$, consider the subgraph $G_{\sigma, S_1, \ldots, S_{\delta n}} := G^*_{\sigma, S_1, \ldots, S_{\delta n}} \restriction W$. By construction, for all $\sigma, S_1, \ldots, S_{\delta n}$, the outdegree of every vertex of $G_{\sigma, S_1, \ldots, S_{\delta n}}$ is at most $(K^n/2) \cdot r$ for some $r = 2^{-(\beta - o(1))n}$.

Let $\gamma = \beta/4$. Let $M = 2^{\gamma n}$ and $\epsilon = 2^{-\gamma n/2}$. Choose a subset $C_{out} \subseteq W$ by including each vertex of W in C_{out} independently with probability $M/|W|$ so that $\mathbb{E}[|C_{out}|] = M$.

By Lemma 4.27 for $N = K^n/2, \beta, \gamma, M, \epsilon, r$, and $G_{\sigma, S_1, \ldots, S_{\delta n}}$, we have that, for all σ and sufficiently large n,

$$\mathop{\mathbf{Pr}}_{C_{out}}\left[\#\{X \in C_{out} : \exists Y \in C_{out} \text{ s.t. } \overrightarrow{YX} \in E(G_{\sigma, S_1, \ldots, S_{\delta n}})\} > \epsilon|C_{out}|\right] \;<\; 2^{-\omega(n)}. \tag{59}$$

This is equivalently

$$\mathop{\mathbf{Pr}}_{C_{out}}\left[\#\{X \in C_{out} : \exists Y \in C_{out} \text{ s.t. } \sigma(X) \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})} Y\} > \epsilon|C_{out}|\right] \;<\; 2^{-\omega(n)}. \tag{60}$$

Union bounding over the at-most-2^n possible values of σ and $\binom{K}{\lambda-1}^{\delta n}$ values of $S_1, \ldots, S_{\delta n}$ gives

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{C_{out}}\left[\exists \sigma, S_1, \ldots, S_{\delta n} \text{ s.t. } \#\{X \in C_{out} : \exists Y \in C_{out} \text{ s.t. } \sigma(X) \prec_{(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\sigma n})} Y\} > \epsilon|C_{out}|\right] \\
&\qquad<\; 2^n \cdot \binom{K}{\lambda - 1}^{\delta n} \cdot 2^{-\omega(n)} \\
&\qquad<\; 2^n \cdot 2^{n\delta(\lambda-1)\log K} \cdot 2^{-\omega(n)} = 2^{-\omega(n)}.
\end{aligned} \tag{61}$$

Additionally, with probability approaching 1, $|C_{out}| \geq \frac{3}{4}M$. Thus, there exists a C_{out} with $|C_{out}| \geq \frac{3}{4}M$ such that the following holds for all $\sigma \in \mathcal{D}(n, (1-\delta)n)$ and all $S_1, \ldots, S_{\delta n} \in \binom{[K]}{\lambda-1}$: at most $\epsilon|C_{out}|$ of the codewords X are $(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})$ matchable in some other codeword $Y \in C_{out}$.

By Lemma 4.7, there exists $\sigma \in D(n,(1-\delta)n)$ and $\tau' \in \mathcal{D}(\delta nL)$ such that the following holds: If we write $\tau' = \tau'_1 \frown \cdots \frown \tau'_{\delta n}$ then each τ'_i is $(\lambda-1)$-admissible and preserves all g_j for all j not in some size-$\lambda-1$ set S_i, and furthermore we have $\tau'(\psi(\sigma(X))) \sqsubseteq \tau(\psi(X))$ for all $X \in [K]^n$. By choice of C_{out}, for at least $(1-\epsilon)|C_{out}|$ choices of $X \in C_{out}$, $\sigma(X)$ is not $(2^\lambda, \sqrt{R}, S_1, \ldots, S_{\delta n})$ matchable in all $Y \in C_{out}$. Thus, for these X, we have $\tau'(\psi(\sigma(X))) \not\sqsubseteq \psi(Y)$ for all $Y \in C_{out}$ by Lemma 4.12. Since $\tau'(\psi(\sigma(X))) \sqsubseteq \tau(\psi(X))$ for all $X \in [K]^n$, we have for these $(1-\epsilon)|C_{out}|$ choices of X that $\tau(\psi(X)) \not\sqsubseteq \psi(Y)$ for all $Y \in C_{out}$. Thus, choosing $C = \psi(C_{out})$ gives our desired average deletion code.

Recall λ is the smallest integer such that $(1+p)/2 < 1-2^{-\lambda}$ and $\delta = 1-2^{-\lambda}-p$ so that $2^\lambda = \Theta(1/(1-p))$ and $\delta = \Theta(1-p)$. Recall $K = 2^{\lceil 2^{\lambda+5}/\delta \rceil}$ and $R = 4K^4$ so that $K = 2^{\Theta(1/(1-p)^2)}$. The rate of the outer code is $\log 2^{\gamma n}/n \log K$ and the rate of the inner code is $\log K/L$. The total rate is thus

$$\mathcal{R} = \frac{\log K}{48R \cdot 2R^K} = 2^{-2^{\Theta(1/(1-p)^2)}}. \tag{62}$$

This completes the proof of Theorem 2.2. □

5 Efficiently decodable codes for random deletions with p approaching 1

5.1 Construction

We present a family of constant rate codes that decodes with high probability in a binary deletion channel with deletion fraction p (BDC_p). These codes have rate $c_0(1-p)$ for a constant c_0, which is within a constant of the upper bound $(1-p)$, which even holds for the erasure channel. By Drinea and Mitzenmacher [9] the maximum known rate of a non-efficiently correctable binary deletion channel code is $(1-p)/9$.

The construction is based on the intuition that deterministic codes are better than random codes for the deletion channel. Indeed, for adversarial deletions, random codes correct 0.17 deletions [17], while explicitly constructed codes correct $\frac{1}{3}$ and $\sqrt{2}-1$ deletions [1].

We begin by borrowing a result from [14].

Lemma 5.1 (Corollary of Lemma 2.3 of [14])**.** *For every binary string $s \in \{0,1\}^m$, there are at most $\delta m\binom{m}{(1-\delta)m}^2$ strings $s' \in \{0,1\}^m$ such that it is possible to apply δm deletions to s and s' and obtain the same result.*

Lemma 5.2. *Let $\delta > 0$. There exists a length m binary code of rate $\mathcal{R} = 0.6942(1-2h(\delta)-O(\log(\delta m)/m))$ correcting a δ fraction of insertions and deletions such that each codeword contains only runs of size 1 and 2. Furthermore this code is constructible in time $\tilde{O}(2^{(0.6942+\mathcal{R})m})$.*

Proof. It is easy to show that the number of codewords with only runs of 1 and 2 is F_m, the mth Fibonacci number, and it is well known that $F_m = \varphi^m + o(1) \approx 2^{0.6942m}$ where φ is the golden mean. Now we construct the code by choosing it greedily. Each codeword is confusable with at most $\delta m\binom{m}{(1-\delta)m}^2$ other codewords, so we can choose at least

$$\frac{2^{0.6942m}}{\delta m\binom{m}{(1-\delta)m}^2} = 2^{0.6942m(1-2h(\delta)-O(\log(\delta m)/m))} \tag{63}$$

codewords.

The following gives an algorithm for constructing the code in this greedy fashion. Find all words of length m whose run lengths are only 1 and 2. This can be done by recursion in time $O(F_m) = O(2^{0.6942m})$. Running the greedy algorithm, we need to, for at most $(F_m \cdot 2^{\mathcal{R}})$ pairs of such words, determine whether the

pair is confusable (we only need to check confusability with words already added to the code). Checking confusability of two words under adversarial deletions reduces to checking whether the longest common subsequence is at least $(1-\delta)m$, which can be done in time $O(m^2)$. This gives an overall runtime of $O(m^2 \cdot F_m \cdot 2^{\mathcal{R}m}) = \tilde{O}(2^{(0.6942+\mathcal{R})m})$. □

Corollary 5.3. *There exists a length m binary code of rate 0.23 correcting a 0.06 fraction of insertions and deletions such that each codeword contains runs of size 1 and 2 only and each codeword starts and ends with a 1. Furthermore this code is constructible in time $O(2^m)$.*

Now we follow the approach of [14] and [12].

The code.

- *Outer code* The outer code C_{out} is a Reed Solomon code of length n over $\mathbb{F}_{q^h}$ (where $n = q$) correcting $\epsilon = \frac{1}{1000}$ fraction of errors/erasures. Unlike normal Reed Solomon codes that only store the field symbol, we encode both the symbol and the index as a pair (i, c_i) in the inner code. Our inner code thus has nq^h codewords. For sake of having nice numbers, we take $h = 1$ so that there are $nq = n^2$ codewords. As a remark, we can improve the rate by a factor of 2 by taking larger values of h, at a cost of the decoding time.

- *Inner code* The inner code $C_{in} : n \times \mathbb{F}_q \to \{0,1\}^m$ is given by Lemma 5.3 having rate 0.23 and correcting 0.06 fraction of insertions and deletions. Each codeword has runs of length 1 and 2 only, and each codeword starts and ends with a 1. This code is constructed via brute force and has length $m = 5 \log n$ (in general $\frac{5}{2}(h+1)\log n$).

- *Buffer* We insert a buffer of ϵm 0s between pairs of inner codewords.

- *Duplication* After concatenating the codes and inserting the buffers, we replace each character c (including characters in the buffers) with $\lceil B/(1-p) \rceil$ ($B = 40$) copies of c to obtain our final code C

Rate. The rate of the outer code is $\frac{h}{h+1}(1-2\epsilon)$, the rate of the inner code is 0.23, the buffer and duplications multiply the rate by $\frac{1}{1+\epsilon}$ and $(1-p)/B = (1-p)/40$ respectively. This gives a total rate of slightly greater than $(1-p)/180$.

Decoding algorithm.

- First identify all runs of 0s in the received word with length at least $B\epsilon m/2$. These are our the *decoding buffers* that divide the word into *decoding windows*.

- Divide each decoding window into runs. For each run, if it has more than $1.4B$ bits, decode it as two copies of that bit, otherwise decode it as one copy. For example, 0^{2B} gets decoded as 00 while 0^B gets decoded as 0. For each decoding window, concatenate these decoded bits in their original order, and decode a pair (i, c_i) from each decoding window using C_{in}.

- Use the polynomial reconstruction Welch Berlekamp algorithm to find a degree-at-most-$(1-2\epsilon)n$ polynomial passing through at least $(1-\epsilon)n$ pairs (i, c_i).

Runtime. The inner code is constructible in time $O(2^m) = O(n^5)$, and constructing the outer Reed Solomon code, adding the buffers, and applying duplications can be done in linear time. The overall construction time is thus $O(n^5)$.

A naive implementation of the Welch Berlekamp algorithm takes $O(n^3)$ time (there are faster algorithms which run in near-linear time but this step is not the bottleneck for the overall algorithm). The decoding

with C_{in} can be done in time $\tilde{O}(n^3)$ by brute force search over the n^2 possible codewords. Checking each of the n^2 codewords takes time $O(m^2)$ because we are computing longest common subsequence, giving a total time of $O(n^2m^2) = \tilde{O}(n^2)$ for each inner codeword. We need to run this inner decoding n times, so the total time is $\tilde{O}(n^3)$. The decoding thus takes time $\tilde{O}(n^3)$. Note if we take larger values of h the decoding time becomes $\tilde{O}(n^{h+2})$.

Correctness. We claim that each inner codeword is decoded correctly with probability at least $1-n^{-\Omega(1)}$. To do this, we show that each of the following occurs with probability at most $n^{-\Omega(1)}$.

1. Some inner codeword is decoded incorrectly.
2. A spurious decoding buffer is created.
3. A decoding buffer is deleted.

For 1, suppose an inner codeword $r = r_1 \dots r_k \in C_{in}$ has k runs r_i each of size 1 or 2, so that $m/2 \le k \le m$, suppose our received word after duplicating r and passing through the deletion channel is s, and suppose that applying the decoding algorithm to s produces a codeword r'.

Definition 5.4. A run of α identical bits in the received word s is

- *type-0* if $\alpha = 0$
- *type-1* if $\alpha \in [1, 1.4B]$
- *type-2* if $\alpha \in [1.4B + 1, \infty)$. Note that a run in s can have more than $2B/(1-p)$ bits if it is merged with neighboring runs of the same bit.

By abuse of notation, we say that a length 1 or 2 run r_i of an inner codeword has *type-t_i* if, after duplication and sending through the deletion channel, the resulting string is type-t_i.

By definition, type-1 runs are decoded as single bits, and type-2 runs are decoded as a pair of identical bits.

Let $t_1, \dots, t_k$ be the types of the runs $r_1, \dots, r_k$, respectively. The probability that a run r_i of length 1 is type-2 is the probability that a string of length $B/(1-p)$ has at least $1.4B$ bits after passing through the deletion channel, BDC_p. By Chernoff bounds in Lemma 3.1, this is at most (using the multiplicative form with $\delta = 0.4$ and $\mu = (1-p) \cdot B/(1-p) = B$)

$$\mathbf{Pr}[\hat{X} > (1+0.4)(1-p) \cdot B/(1-p)] \le (e^{0.4}/1.4^{1.4})^{40} \approx 0.058 < 0.06. \tag{64}$$

Similarly, the probability that a run of length-2 is type-1 is at most

$$\mathbf{Pr}[\hat{X} < (1-0.3) \cdot 2(1-p) \cdot B/(1-p)] \le e^{-0.09 \cdot 80/2} \approx 0.027 < 0.06. \tag{65}$$

The probability any run is type-0 is at most $p^{B/(1-p)} \le \left(\frac{1}{e}\right)^B < 1/2^{40}$.

We now have established that runs in r keep their length with probability at least 0.941 after the three steps of duplication, deletion, and decoding. If the only kinds of errors in the runs of r were runs of length 1 becoming runs of length 2 and runs of length 2 become runs of length 1, then we have that, with high probability, the number of insertions deletions needed to transform r' back into r is at most $0.06|r|$ by concentration bounds. However, we must also account for the fact that some runs of r may become deleted completely after passing through the deletion channel, in which case the two neighboring runs become merged together. For example, if a run of 1s was deleted completely after duplication and deletion, its

neighboring runs of 0s would be interpreted by the decoder as a single run. Fortunately, as we saw, the probability that a run is type-0 is extremely small ($< 10^{-12}$), and each type-0 run only locally increases the number of insertions and deletions in r' by at most a constant number. We show this constant is at most 8, and while we could improve the constant, this would have a negligible effect on our analysis: the probability of having a type-0 run is so small that the constant does not substantially affect the expected number of insertions and deletions between r and r', which we ultimately would like to bound away from $0.06|r|$.

Let Y_i be a random variable that is 0 if $|r_i| = t_i$, 1 if $\{|r_i|, t_i\} = \{1, 2\}$, and 8 if $t_i = 0$. We claim $\sum_{i=1}^{k} Y_i$ is an upper bound on the number of insertions and deletions need to manipulate r into r'. To see this, first note that if $t_i \neq 0$ for all i, then the number of runs of r and r' are equal, so we can transform r into r' by adding a bit to each length-1 type-2 run of r and deleting a bit from each length-2 type-1 run of r'. Now if some number ℓ of the t_i are 0, then at most 2ℓ of the runs in r become merged with some other run after duplication and deletion. Each run of r that is merged effectively gets deleted and each set of merged runs gets replaced with 1 or 2 copies of the corresponding bit. For example, if $r_1 = 11, r_2 = 0, r_3 = 11$, r_1 and r_3 are both type-2 where, after duplication and deletion, $2B$ bits remain in each, and r_2 is type-0, then our received word is 1^{4B} which gets decoded as 11 because it is type-2. To account for the type-0 runs in transforming r into r', we delete at most two bits from r for each type-0 run, we delete at most two bits for each of at most 2ℓ merged runs in r, and we insert back at most two bits for each run in the received word s resulting from a merge. The total number of additional insertions and deletions required to account for type-0 runs of r is thus at most 8ℓ, so we need at most 8 insertions and deletions to account for each type-0 run.

Note that if r_i has length 1, then by (64) we have

$$\mathbb{E}[Y_i] \;=\; 1 \cdot \mathbf{Pr}[r_i \text{ is type-2}] + 8 \cdot \mathbf{Pr}[r_i \text{ is type-0}] \;<\; 1 \cdot 0.058 + 8 \cdot (1/2^{40}) \;<\; 0.059. \tag{66}$$

Similarly, if r_i has length 2, then by (65) we have

$$\mathbb{E}[Y_i] \;=\; 1 \cdot \mathbf{Pr}[r_i \text{ is type-1}] + 8 \cdot \mathbf{Pr}[r_i \text{ is type-0}] \;<\; 1 \cdot 0.027 + 8 \cdot (1/2^{40}) \;<\; 0.059. \tag{67}$$

Thus $\mathbb{E}[Y_i] < 0.059$ for all i. The Y_i are independent so the Chernoff bound gives

$$\begin{aligned}
\mathbf{Pr}[r \text{ decoded incorrectly}] &\leq \mathbf{Pr}[Y_1 + Y_2 + \cdots + Y_k \geq 0.06m] \\
&\leq \mathbf{Pr}[Y_1 + Y_2 + \cdots + Y_k \geq 0.06k] \\
&\leq \mathbf{Pr}[Y_1 + Y_2 + \cdots + Y_k \geq (1 + 0.001)(0.059)k] \\
&\leq e^{-\text{const} \cdot \log n} = n^{-\Omega(1)}
\end{aligned} \tag{68}$$

A spurious buffer appears inside an inner codeword if many consecutive runs of 1s are type-0. To be precise, a set of ℓ consecutive type-0 runs of 1s occurs with probability at most $2^{-40\ell}$. Thus the probability an inner codeword has a sequence of $\epsilon m/5$ consecutive type-0 runs of 1s is at most $m^2 \cdot 2^{-40\epsilon m/5} = n^{-\Omega(1)}$. Now assume that in an inner codeword, each set of consecutive type-0 runs of 1s has size at most $\epsilon m/5$. Each set of ℓ consecutive type-0 runs of 1s merges $\ell + 1$ consecutive runs of 0s in r, so that they appear as a single longer run in s. The sum of the length of these $\ell + 1$ runs is at most $2\ell + 2$, so after duplication these runs have total length at most $B(2\ell + 2)/(1 - p) < 2.1B\epsilon m/5(1 - p)$. Thus the length of their merged run in s is the sum of at most $2.1B\epsilon m/5(1 - p)$ i.i.d random variables with mean $1 - p$, so the probability the sum is at least $B\epsilon m/2$ is at most $2^{-O(\epsilon m)} = n^{-\Omega(1)}$. Hence, conditioned on each set of consecutive type-0 runs of 1s having size at most $\epsilon m/5$, the probability of having no spurious buffers is at least $1 - n^{-\Omega(1)}$. Thus the overall probability there are no spurious buffers in the inner codeword is at least $(1 - n^{-\Omega(1)})(1 - n^{-\Omega(1)}) = 1 - n^{-\Omega(1)}$. Since each inner codeword contains at most m spurious buffers, the expected number of spurious buffers is $m \cdot n^{-\Omega(1)} = n^{-\Omega(1)}$.

A deleted buffer occurs only when the sum of $B\epsilon m/(1 - p)$ i.i.d Bernoulli random variables with mean $1 - p$ sum to less than $B\epsilon m/2$, which by Chernoff bounds happens with probability $2^{-\Omega(m)} = n^{-\Omega(1)}$.

Each spurious buffer, deleted buffer, and incorrectly decoded inner codeword increases the number of errors and or erasures by at most $O(1)$. The expected number of incorrect inner codewords plus spurious buffers plus deleted buffers is therefore $n^{1-\Omega(1)}$, so the probability that this is at least δn is at most $n^{-\Omega(1)}$ by Markov's inequality, so the RS decoding succeeds with probability $1 - n^{-\Omega(1)}$ as desired. We believe that, with a more careful analysis, the decoding failure probability can be made $2^{-\Omega(n)}$. This concludes the proof of Theorem 1.2.

5.2 Possible Alternative Constructions

In a binary deletion channel with deletion probability p (BDC_p), each bit is deleted independently with probability p. Drinea and Mitzenmacher [9, 10] proved that the capacity of the BDC_p is at least $(1-p)/9$. However, their proof is nonconstructive and they do not provide an efficient decoding algorithm.

One might think it is possible to use Drinea and Mitzenmacher's construction as a black box. We could follow the approach in §5.1, [14] and [12] by concatenating a rate $(1-p)/9$ random-deletion-correcting binary code with an outer Reed Solomon code, so that the outer code has length n and the inner code has length $m = O(\log n)$. We can try to prevent synchronization errors in the same way by (1) encoding a Reed Solomon codeword with both its position and symbol $(i, c_i) \in [n] \times \mathbb{F}_n$ and (2) inserting buffers of 0s of length ϵm in between inner codewords. Even though Drinea and Mitzenmacher's so-called *jigsaw decoding* does not run efficiently, it is not hard to show (Lemma 5.5) that a jigsaw-decodable code, after possibly throwing out a small fraction of codewords, is maximum-likelihood decodable.

For maximum likelihood decoding on the inner code, we only need to compare the received string s with every codeword c to determine the probability s was received given c and choose the c with highest probability. Each comparison can be done in $\operatorname{polylog} n$ time by dynamic programming, and there are $2^{O(\log n)} = \operatorname{poly}(n)$ inner codewords, so maximum likelihood decoding on the inner code runs efficiently. Then, as the inner code decodes $(1-p)/9$ random deletions with high probability, the outer code has exponentially small numbers of errors, so the Reed-Solomon code decodes correctly.

Lemma 5.5. *Suppose a code C is decodable under random deletions with probability p using some kind of decoder, not necessary the maximum likelihood decoder. That is, for each codeword $c \in C$, after sending through the random deletion channel, the decoder returns c with probability $1 - o(1)$. Then, if we replace the decoder with the maximum likelihood decoder, we can find a code C' such that $|C'| \geq |C|/2$ such that each codeword $c \in C'$ is decoded corrected with probability $1 - o(1)$.*

However, this approach runs into several issues

1. First, the inner code is still nonconstructive, and constructing the inner code by brute force takes $2^{2^{O(\log n)}} = 2^{\operatorname{poly} n}$ time. Thus, for this idea to work, one needs multiple layers of concatenation, which would be nontrivial to analyze.

2. Second, while the inner code can tolerate random deletions with probability p, bits are *not* deleted in the concatenated construction according to a BDC_p; the 0 bits closer to the buffers between the inner codewords are deleted with higher probability because they might be "merged" with a buffer. For example, if an inner codeword is 101111, then because the codeword is surrounded by buffers of 0s, deleting the leftmost 1 morally deletes two bits because the 0 is interpreted as part of the buffer. While this may not be a significant issue because the distributions of deletions in this deletion process and BDC_p are quite similar, much more care would be needed to prove correctness.

6 Conclusion

6.1 Decoding deletions vs. insertions and deletions

A lemma due to Levenshtein [22] states that a code C can decode against pn adversarial deletions if and only if it can decode against pn adversarial insertions and deletions.

In oblivious and random error models, decoding insertions vs insertions and deletions are not the same. In decoding against oblivious deletions, we not only need to worry about *whether* two codewords are by insertions and deletions, but also *the number of ways* in which they are confusable. In particular, it is okay (and necessary when the error fraction is greater than $\frac{1}{2}$) in the oblivious model for two codewords c and c' to be confusable, but we need the number of deletion patterns τ that confuse them, i.e. with $\tau(c) \sqsubseteq c'$ or $\tau(c') \sqsubseteq c$, to be small. It may be possible to make an assertion that if the number of ways c and c' are confusable by deletion patterns is small, then the number of ways they are confusable by insertion and deletion patterns is also small. This would show that our oblivious deletion code could also decode against oblivious insertions and deletions.

For the random model, it is not even clear how to define random insertions. One could define insertions and deletions via the Poisson repeat channel where each bit is replaced with a Poisson many copies of itself (see [9, 25]). Alternatively one can consider a model of random insertions and deletions where, for every bit, the bit is deleted with a fixed probability p_1, a bit is inserted after it with a fixed probability p_2, or it is transmitted unmodified with probability $1 - p_1 - p_2$ [27]. One could also investigate settings involving memoryless insertions, deletions, and substitutions [24].

It may be possible to extend our oblivious and random deletion construction to oblivious and random insertion and deletions by tweaking the parameters. This was the case for bridging the gap between efficiently decoding adversarial deletions [14] and efficiently decoding adversarial insertions and deletions [12]. Our analysis for deletions analyzes runs of 0s and 1s, but perhaps it would be possible to analyze insertions and deletions by relaxing the definition of a run to be "a subword with 0.9 density of 0s and 1s." For now, however, we leave the case of correcting oblivious insertions and deletions approaching 1 as an open problem.

6.2 Open problems

These are a number of open questions concerning deletion codes. Our work brings to the fore a few more.

1. Can we close the gap between $\sqrt{2} - 1$ and $\frac{1}{2}$ for adversarial deletions?

2. We showed the existence of codes that decode against oblivious and average case deletions. Can we modify the proof to show existence of codes decoding against oblivious and average case insertions and deletions?

3. Can we adapt the construction for oblivious and average case deletions to obtain explicit codes that are constructable, encodable, and decodable in polynomial time?

4. For erasures, the capacity of the random error channel and the capacity of the oblivious error channel are both $1 - p$. For bit flips the random and oblivious error channels also have the same capacity at $1 - h(p)$. Can we construct codes that decode oblivious deletions with capacity equal to the capacity of the random deletion channel? On the other end, could we upper bound the capacity by $o(1 - p)$ as $p \to 1$? Such an upper bound would fundamentally distinguish the behavior of the deletions from errors and erasures.

5. Can we find efficiently decodable codes for random deletions with better rate, perhaps reaching or beating the best known existential capacity lower bound of $(1 - p)/9$?

6. Can find codes correcting p fraction of *online* deletions when p approaches $\frac{1}{2}$? (see §1.2 for discussion)

7 Acknolwedgements

The authors would like to thank Joshua Brakensiek and Boris Bukh for helpful discussions.

References

[1] Boris Bukh, Venkatesan Guruswami, and Johan Håstad. An improved bound on the fraction of correctable deletions. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1893–1901, 2016.

[2] Zitan Chen, Sidharth Jaggi, and Michael Langberg. A characterization of the capacity of online (causal) binary channels. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 287–296, New York, NY, USA, 2015. ACM.

[3] Imre Csiszár and Prakash Narayan. Arbitrarily varying channels with constrained inputs and states. *IEEE Transactions on Information Theory*, 34(1):27–34, 1988.

[4] Imre Csiszár and Prakash Narayan. Capacity and decoding rules for classes of arbitrarily varying channels. *IEEE Transactions on Information Theory*, 35(4):752–769, 1989.

[5] Daniel Cullina and Negar Kiyavash. An improvement to Levenshtein's upper bound on the cardinality of deletion correcting codes. *IEEE Trans. Information Theory*, 60(7):3862–3870, 2014.

[6] Marco Dalai. A new bound on the capacity of the binary deletion channel with high deletion probabilities. In *2011 IEEE International Symposium on Information Theory Proceedings, ISIT 2011, St. Petersburg, Russia, July 31 - August 5, 2011*, pages 499–502, 2011.

[7] Bikash Kumar Dey, Sidharth Jaggi, Michael Langberg, and Anand D. Sarwate. A bit of delay is sufficient and stochastic encoding is necessary to overcome online adversarial erasures. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 880–884, 2016.

[8] Suhas Diggavi and Matthias Grossglauser. On transmission over deletion channels. In *Proceedings of the 39th Annual Allerton Conference on Com- munication, Control, and Computing*, pages 573–582, 2001.

[9] Eleni Drinea and Michael Mitzenmacher. On lower bounds for the capacity of deletion channels. *IEEE Transactions on Information Theory*, 52(10):46484657, 2006.

[10] Eleni Drinea and Michael Mitzenmacher. Improved lower bounds for the capacity of i.i.d. deletion and duplication channels. *IEEE Trans. Information Theory*, 53(8):26932714, 2007.

[11] Dario Fertonani, Tolga M. Duman, and M. Fatih Erden. Bounds on the capacity of channels with insertions, deletions and substitutions. *IEEE Trans. Communications*, 59(1):2–6, 2011.

[12] Venkatesan Guruswami and Ray Li. Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 620–624, 2016.

[13] Venkatesan Guruswami and Adam Smith. Optimal rate code constructions for computationally simple channels. *Journal of the ACM*, 2016. To appear. Preliminary version in FOCS 2010.

[14] Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, pages 867–880, 2015.

[15] Adam Kalai, Michael Mitzenmacher, and Madhu Sudan. Tight asymptotic bounds for the deletion channel with small deletion probabilities. In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 997–1001, 2010.

[16] Yashodhan Kanoria and Andrea Montanari. Optimal coding for the binary deletion channel with small deletion probability. *IEEE Trans. Information Theory*, 59(10):6192–6219, 2013.

[17] Ian Kash, Michael Mitzenmacher, Justin Thaler, and John Ullman. On the zero-error capacity threshold for deletion channels. In *Information Theory and Applications Workshop (ITA)*, pages 1–5, January 2011.

[18] Adam Kirsch and Eleni Drinea. Directly lower bounding the information capacity for channels with i.i.d.deletions and duplications. *IEEE Trans. Information Theory*, 56(1):86–102, 2010.

[19] Ankur A. Kulkarni and Negar Kiyavash. Nonasymptotic upper bounds for deletion correcting codes. *IEEE Trans. Information Theory*, 59(8):5115–5130, 2013.

[20] Michael Langberg. Oblivious communication channels and their capacity. *IEEE Trans. Information Theory*, 54(1):424–429, 2008.

[21] Amos Lapidoth and P. Narayan. Reliable communication under channel uncertainty. *IEEE Trans. Information Theory*, 44(6):2148–2177, 1998.

[22] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Dokl. Akad. Nauk*, 163(4):845–848, 1965 (Russian). English translation in Soviet Physics Doklady, 10(8):707-710, 1966.

[23] Vladimir I. Levenshtein. Bounds for deletion/insertion correcting codes. In *Proceedings of the IEEE International Symposium on Information Theory*, 2002.

[24] Hugues Mercier, Vahid Tarokh, and Fabrice Labeau. Bounds on the capacity of discrete memoryless channels corrupted by synchronization and substitution errors. *IEEE Trans. Information Theory*, 58(7):4306–4330, 2012.

[25] Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009.

[26] Mojtaba Rahmati and Tolga M. Duman. An improvement of the deletion channel capacity upper bound. In *51st Annual Allerton Conference on Communication, Control, and Computing, Allerton 2013, Allerton Park & Retreat Center, Monticello, IL, USA, October 2-4, 2013*, pages 1221–1225, 2013.

[27] Ramji Venkataramanan, Sekhar Tatikonda, and Kannan Ramchandran. Achievable rates for channels with deletions and insertions. *IEEE Trans. Information Theory*, 59(11):6990–7013, 2013.

[28] S. M. Sadegh Tabatabaei Yazdi and Lara Dolecek. A deterministic polynomial-time protocol for synchronizing from deletions. *IEEE Trans. Information Theory*, 60(1):397–409, 2014.

A Relationships between error models

The following theorem establishes relationships between the capacities of the adversarial, oblivious, average case, and random deletion channels.

Theorem A.1. *We have the following.*

1. *The capacity of the adversarial deletion channel with deletion probability p is at most the capacity of each of the oblivious, average case, and random deletion channels.*
2. *The capacity of average case decoding of deletions is at most the capacity of the oblivious deletion channel.*
3. *The capacity of average case decoding of deletions is at most the capacity of the random deletion channel.*

In this appendix, we prove the following lemma, which establishes item 2 of Theorem A.1.

Lemma A.2. *Let $p \in (0,1)$. Suppose we have a family of codes C of length n and rate $\mathcal{R}$ such that, for any deletion pattern τ with at most pn deletions, at most $\epsilon = o_n(1)$ of the codewords are decoded incorrectly. Then, for any $\delta < \frac{1}{2}$, provided n is sufficiently large, we can find a family of stochastic codes C' of length n and rate $\mathcal{R}(1-\delta) - o(1)$ that corrects pn oblivious deletions with probability $1 - 3\epsilon$.*

Proof. For set of words C^* a pair (c, τ) (where $c \in C^*$ and τ is a deletion pattern) C^*-bad if $\tau(c) \subseteq c'$ for some $c' \in C^* \setminus \{c\}$, and C^*-good otherwise.

Let $M = |C| = 2^{\mathcal{R}n}$. For any τ, let A_τ denote the set of c such that (c, τ) is C-bad. Then $|A_\tau| \leq \epsilon|C|$ by assumption that C decodes against pn deletions in the average case. Let $d = \lceil h(p)/\mathcal{R} \rceil$ be a constant over varying n, so that there are at most $M^d > \binom{n}{pn}$ choices of τ.

Let $t = M^\delta$, and $N = \lfloor M^{1-\delta}/2 \rfloor$. We construct C' iteratively. For $1 \leq k \leq N$, chose a random subset of exactly t codewords from $C \setminus \cup_{i=1}^{k-1} E_i$ to form E_k. With these sets of codewords, we associate each message m_i with a set E_i. We encode each message m_i by choosing uniformly at random one of t codewords in some set E_i.

Note that $C' \subseteq C$ and $|C'| = Nt = M/2$. It is easy to see the rate of C' is $\mathcal{R}(1-\delta) - o(1)$.

We claim that, with positive probability over the choice of C', C' corrects pn oblivious deletions with probability $1 - 3\epsilon$. Define B_τ to be all c such that (c, τ) is C'-bad. As $B_\tau \subseteq A_\tau$, we have $|B_\tau| \leq \epsilon M$. We wish to show that the probability there exists an B_τ and a message m_i with encoding set E_i such that $|B_\tau \cap E_i| \geq 3\epsilon|E_i|$ is less than 1. Fix a τ. We show that the probability $|B_\tau \cap E_i| \geq 3\epsilon|E_i|$ is tiny. When we chose E_i, we can imagine we picked the t elements of E_i one after the other. Each one was chosen from at least $M/2$ codewords, so each codeword lands in $|B_\tau \cap E_i|$ with probability at most $\epsilon M/(M/2) = 2\epsilon$. By Chernoff bounds, $\mathbf{Pr}\left[|B_\tau \cap E_i| \geq 3\epsilon|E_i|\right] \leq e^{-t/3}$. By union bound over all E_i and all B_τ, we have

$$\mathbf{Pr}\left[\exists E_i \exists \tau : |B_\tau \cap E_i| \geq 3\epsilon|E_i|\right] \;\leq\; N \cdot M^d \cdot e^{-t/3} \;<\; M^{d+1}/e^{M^\delta/3} \tag{69}$$

This is less than 1 for $n > \frac{2}{\delta\mathcal{R}} \log(3(d+1))$, so our code corrects pn oblivious deletions with probability $1 - 3\epsilon$, as desired. □

B Comparison of oblivious deletions with oblivious bit flips

First, for completeness, we outline the proof that that there exist codes achieving oblivious bit-flip capacity. After this, we show that it is not possible to follow the same approach to construct oblivious deletion codes.

Theorem B.1 ([20, 3])**.** *The capacity of oblivious bit-flip channels is* $1 - h(p)$*.*

Proof that capacity $1 - h(p)$ *is achievable.* Note that $1 - h(p)$ is an upper bound for the capacity as the capacity of the binary symmetric channel is $1 - h(p)$ and the adversary can always just choose the error vector randomly.

We consider a stochastic code C where each message m is mapped uniformly into a set $E(m)$ of size t. Choose $t = n$ and $\epsilon = 1/\log n$. This code decodes successfully if, for all m,

$$\mathop{\mathbf{Pr}}_{c\in E(m)}\left[c + e \in \bigcup_{c\in C\setminus E(m)} B_{pn}(c')\right] \geq 1 - \epsilon \tag{70}$$

where $B_{pn}(x) \subseteq \{0,1\}^n$ is the set of all words within Hamming distance pn of x.

Choose C to have $2^{\mathcal{R}n}$ codewords at random. Fix a message m, the rest of the codeword $C \setminus E(m)$, and the error vector e. We have

$$\mathop{\mathbf{Pr}}_{c\sim\{0,1\}^n}\left[c \in \bigcup_{c\in C\setminus E(m)} B_{pn}(c' + e)\right] \leq \frac{1}{2^n}|C| \cdot |B_{pn}(0)| \approx \frac{2^{\mathcal{R}n} \cdot 2^{h(p)n}}{2^n}, \tag{71}$$

which is $2^{-\Omega(n)}$ if $\mathcal{R} < 1 - h(p)$. In this case

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{E(m)}\left[\mathop{\mathbf{Pr}}_{c\in E(m)}\left[c + e \notin \bigcup_{c\in C\setminus E(m)} B_{pn}(c')\right] < 1 - \epsilon\right] \\
&= \mathop{\mathbf{Pr}}_{E(m)}\left[\mathop{\mathbf{Pr}}_{c\in E(m)}\left[c \notin \bigcup_{c\in C\setminus E(m)} B_{pn}(c' + e)\right] < 1 - \epsilon\right] \\
&= \mathop{\mathbf{Pr}}_{E(m)}\left[\left|E(m) \cap \bigcup_{c\in C\setminus E(m)} B_{pn}(c' + e)\right| > \epsilon t\right] \\
&\leq 2^{-\Omega(\epsilon n t)} = 2^{-\tilde{\Omega}(n^2)}.
\end{aligned} \tag{72}$$

Thus, applying a union bound over all m and error vectors e, we get

$$\mathop{\mathbf{Pr}}_{C}\left[\exists m : \mathop{\mathbf{Pr}}_{c\in E(m)}\left[c + e \in \bigcup_{c\in C\setminus E(m)} B_{pn}(c')\right] > \epsilon\right] \leq 2^{\mathcal{R}n} \cdot 2^n \cdot 2^{-\tilde{\Omega}(n^2)} \leq 2^{-\tilde{\Omega}(n^2)}. \tag{73}$$

Thus when $\mathcal{R} < 1 - h(p)$, we can construct a rate $\mathcal{R}$ stochastic code correcting w.h.p. p oblivious bit flips. □

For oblivious deletions the above technique does not work. This is established by the following lemma.

Lemma B.2. *Let C be a random length n rate $\mathcal{R}$ stochastic binary code such that each message is encoded uniformly at random in one of t codewords. If $p \geq 0.4$ and $0 < \epsilon < 1$, then for any message m and deletion pattern τ we have*

$$\mathop{\mathbf{Pr}}_{C}\left[\mathop{\mathbf{Pr}}_{c\in E(m)}\left[\exists c' \in C \setminus E(m) : \tau(c) \sqsubseteq c'\right] > \epsilon\right] > 2^{-h(p)n} \tag{74}$$

where $D_k(s) = \{s' : |s'| = |s| - k, s' \leq s\}$.

In plain English, this lemma says that for a fixed message and deletion pattern, the probability that message decodes incorrectly too many times ($> \epsilon$ fraction of the time) is too large ($> 2^{-h(p)n}$). In short, the reason the lemma is true is because the probability the entire code C contains a "really bad" word (e.g. a word with at least $0.91n$ runs of 1s) is too large. The following remark shows us, using Lemma B.2, why we cannot follow the same randomized approach as Theorem B.1.

Remark B.3. Following the approach of Theorem B.1, we would like to conclude, using union bound,

$$\begin{aligned}
&\Pr_C\left[\forall m \forall \tau \Pr_{c\in E(m)}\left[\exists c' \in C \setminus E(m) : \tau(c) \sqsubseteq c'\right] > \epsilon\right] \\
&\leq \sum_m \sum_\tau \Pr_C\left[\Pr_{c\in E(m)}\left[\exists c' \in C \setminus E(m) : \tau(c) \sqsubseteq c'\right] > \epsilon\right] \\
&\leq 2^{\mathcal{R}n}\binom{n}{pn}\Pr_C\left[\Pr_{c\in E(m)}\left[\exists c' \in C \setminus E(m) : \tau(c) \sqsubseteq c'\right] > \epsilon\right] \\
&< 2^{\mathcal{R}n}2^{h(p)n}2^{-h(p)n-\Omega(n)} = 2^{-\Omega(n)}
\end{aligned} \tag{75}$$

Unfortunately the last inequality (indicated in red) is false by Lemma B.2.

Proof. It suffices to prove for $p = 0.4$. Indeed, as p increases,

$$\Pr_{c\in E(m)}\left[\tau(c) \in \bigcup_{c'\in C\setminus E(m)} D_{pn}(c')\right] \tag{76}$$

increases as $\tau(c)$ contains fewer symbols.

If $p = 0.4$, then $\tau(c)$ has $0.6n$ characters. Note that if c is uniform on $\{0,1\}^n$, then $\tau(c)$ is uniform on $\{0,1\}^{0.6n}$. It is easy to check that for a uniformly chosen $0.6n$ string, the probability it is a subsequence of the length $0.91n$ string $a_{0.91n} = 0101\ldots01$ is $1 - 2^{-\Omega(n)}$: the position in the longest string increases by 1.5 each time, so in expectation the string spans $0.9n$ characters of $a_{0.91n}$, so it is not a subsequence with exponentially small probability by Chernoff bounds.

There are $2^{0.09n}$ length n strings that start with $a_{0.91n}$. Call this set of strings A. Thus, the probability that C contains an element of A is at least $2^{-0.91n}$ since that is the probability the first element is an element of A. Conditioned on C containing an element of A, the probability that, for some $c \in E(m)$, $\tau(c)$ is a subsequence of some element of A is at least the probability that c is a subsequence of $a_{0.91n}$, which is $1 - 2^{-\Omega(n)}$. Thus, conditioned on C containing an element of A, we have

$$\mathbb{E}\left[\#\{c : \exists c' \in C \setminus E(m) \text{ s. t. } \tau(c) \leq c'\}\right] = t(1 - 2^{-\Omega(n)}) \tag{77}$$

Conditioned on C being fixed, consider the indicator random variables X_i for whether each $c_i \in E(m)$ satisfies $\tau(c_i)$ is a subsequence of some $c' \in C \setminus E(m)$. The $X_1, \ldots, X_t$ are i.i.d, so the probability $\sum X_i/t > \epsilon$ is approximately 1, (for sure, it is $1 - 2^{-\Omega(n)}$).

Thus we conclude

$$\Pr_C[\#\{c : \exists c' \in C \setminus E(m) \text{ s. t. } \tau(c) \leq c'\} > \epsilon t] \geq (1 - 2^{-\Omega(n)})\Pr_C[C \cap A \neq \emptyset] > 2^{(0.91-o(1))n} > 2^{-H(0.4)n} \tag{78}$$

as $H(0.4) \approx 0.97$. □

Intuitively B.2 should be true as deletion codes behave poorly when chosen completely randomly. In the adversarial deletion channel, random codes correct only $0.17n$ deletions [17], while the best known constructions correct $0.41n$ deletions [1].

C Proofs of Lemma 4.19 and Lemma 4.20

Lemma C.1. *For any $j \in [1, K]$, if Z is a random variable distributed as* $\min(\text{Geometric}(j/K), \sqrt{R} - 1)$, *then* $\mathbb{E}[Z] > \frac{K}{2j} - 1$.

Proof. We have

$$\begin{aligned}
\mathbb{E}[D] &= \left(\frac{j}{K}\right) \cdot 1 + \left(\frac{j}{K}\right)\left(1 - \frac{j}{K}\right) \cdot 2 + \cdots + \left(\frac{j}{K}\right)\left(1 - \frac{j}{K}\right)^{\sqrt{R}-2} \cdot (\sqrt{R} - 1) \\
&= \frac{1 - \left(1 - \frac{j}{K}\right)^{\sqrt{R}-1}}{j/K} - (\sqrt{R} - 1)\left(1 - \frac{j}{K}\right)^{\sqrt{R}-1} \\
&\geq \frac{1 - \left(1 - \frac{1}{K}\right)^{\sqrt{R}-1}}{j/K} - (\sqrt{R} - 1)\left(1 - \frac{1}{K}\right)^{\sqrt{R}-1} > \frac{K}{2j} - 1 \qquad (79)
\end{aligned}$$

The last inequality follows from recalling $R = 4K^4$ and twice applying

$$\left(\sqrt{R} - 1\right)\left(\frac{K-1}{K}\right)^{\sqrt{R}-1} < 2K^2 \cdot \left(\frac{K-1}{K}\right)^{K^2} < 2K^2 \cdot \left(\frac{1}{2}\right)^{K} < 1, \qquad (80)$$

which is true since $K > 8$. □

Lemma (Lemma 4.19)**.** *Let J be chosen uniformly from $[K]$. Let D be a random variable that is 1 if $J \in [\lambda - 1]$ and, conditioned on a fixed $J \geq \lambda$, is distributed as* $\min(\text{Geometric}(J/K), \sqrt{R})$. *Then* $\mathbb{E}[D] \geq (\log K)/4$.

Proof. Applying Lemma C.1,

$$\begin{aligned}
\mathbb{E}[D] &= \frac{1}{K}\sum_{j=1}^{K} \mathbb{E}[D|J = j] > \sum_{j=1}^{\lambda-1} \frac{1}{K} \cdot 1 + \sum_{j=\lambda}^{K} \frac{1}{K}\left(\frac{K}{2j} - 1\right) \\
&\geq -1 + \sum_{j=\lambda}^{K}\left(\frac{1}{2j}\right) > \frac{1}{2}(\ln K - \ln \lambda - 1) - 1 > \frac{1}{4}\log K. \qquad (81)
\end{aligned}$$

□

Lemma (Lemma 4.20)**.** *Let $\lambda' \in [\lambda, K]$ and let J be chosen uniformly from $\{\lambda, \lambda + 1, \ldots, \lambda'\}$. Let D be the random variable that, conditioned on a fixed J, is distributed as* $\min(\text{Geometric}(J/K), \sqrt{R})$. *Then* $\mathbb{E}[D] \geq (\log K)/4$.

Proof. Applying Lemma C.1,

$$\begin{aligned}
\mathbb{E}[D] &= \frac{1}{\lambda' - \lambda + 1}\sum_{j=\lambda}^{\lambda'} \mathbb{E}[D|J = j] > \frac{1}{\lambda' - \lambda + 1}\sum_{j=\lambda}^{\lambda'}\left(\frac{K}{2j} - 1\right) \\
&> \frac{1}{K - \lambda + 1}\sum_{j=\lambda}^{K}\left(\frac{K}{2j} - 1\right) > -1 + \sum_{j=\lambda}^{K}\left(\frac{1}{2j}\right) > \frac{1}{4}\log K. \qquad (82)
\end{aligned}$$

□

D Proof of Lemma 5.5

Suppose C has block length n and that each codeword decodes correctly with probability $1-\epsilon$ using the some initial decoder. Let X_c be the indicator random variable that c decodes correctly under the initial decoder, and Y_c be the random variable that c decodes correctly under maximum likelihood. By assumption, $\mathbb{E}[X_c] \geq 1-\epsilon$ for all $c \in C$. By definition of maximum likelihood decoder, we have

$$\sum_{c \in C} \mathbb{E}[Y_c] \geq \sum_{c \in C} \mathbb{E}[X_c] \geq (1-\epsilon)|C| \tag{83}$$

Since $\mathbb{E}[Y_c] \leq 1$ for all c, it follows that at most $|C|/2$ codewords of C have $\mathbb{E}[Y_c] \leq 1-2\epsilon$. Take the remaining at least $|C|/2$ codewords to form a new code C'. Note having less codewords only increases the probability any codeword is decoded correctly. Under the maximum likelihood decoder, each of these codewords decodes correctly with probability at least $1-2\epsilon$, as desired.