Electronic Colloquium on Computational Complexity, Report No. 172 (2023)

Constant Query Local Decoding Against Deletions Is Impossible

Meghal Gupta

UC Berkeley

meghal@berkeley.edu

November 14, 2023

Abstract

Locally decodable codes (LDC's) are error-correcting codes that allow recovery of individual message indices by accessing only a constant number of codeword indices. For substitution errors, it is evident that LDC's exist – Hadamard codes are examples of 2-query LDC's. Research on this front has focused on finding the optimal encoding length for LDC's, for which there is a nearly exponential gap between the best lower bounds and constructions.

Ostrovsky and Paskin-Cherniavsky (ICITS 2015) introduced the notion of local decoding to the insertion and deletion setting. In this context, it is not clear whether constant query LDC's exist at all. Indeed, in contrast to the classical setting, Block et al. conjecture that they do not exist. Blocki et al. (FOCS 2021) make progress towards this conjecture, proving that any potential code must have at least exponential encoding length.

Our work definitively resolves the conjecture and shows that constant query LDC's do not exist in the insertion/deletion (or even deletion-only) setting. Using a reduction shown by Blocki et al., this also implies that constant query locally correctable codes do not exist in this setting.

ISSN 1433-8092

Contents

1 Introduction

The study of classical error-correcting codes dates back to Shannon and Hamming in the mid-1900's [Sha48, Ham50]. While the initial focus was on substitution errors, the field expanded to address *synchronization errors* with the seminal work of [Lev66]. Synchronization errors involve inserting or deleting symbols in a transmitted message rather than substituting symbols. Such errors are prevalent in applications such as text/speech processing [CS08], DNA storage technologies [GYM16, LSWZY19], and communication complexity [BGMO17].

Within the study of classical (substitution) error-correcting codes, the concept of *locally decodable codes* (LDC's) has garnered considerable attention. In a locally decodable encoding C of a message $x \in \Sigma^n$ for some alphabet Σ, a receiver can decode any individual symbol $x[i]$ by making a constant number of queries[1] to $\mathsf{C}(x)$, even in the presence of a small constant fraction of adversarial corruption. It is evident that such codes exist in the classical setting, for example Hadamard codes, and research has focused on finding the optimal encoding length. Despite considerable effort, there remains a nearly exponential gap between the best lower bounds and constructions [Yek12].

Recently, the work of Ostrovsky and Paskin-Cherniavsky [OPC15] introduced the notion of local decoding to insertion/deletion codes. Unlike in the setting of substitutions, here it is not clear whether constant query LDC's exist at all: For a constant-sized alphabet Σ, is there an encoding function $\mathsf{C} : \Sigma^n \to \Sigma^M$ such that any one symbol $x[i]$ of the original message can be recovered with high probability using only a constant number of queries to the corrupted transmission?

Block et al. [BBG+20] conjecture that in contrast to the substitution setting, constant query insertion/deletion LDC's do not exist. That is, even when the encoding can be arbitrarily long in terms of n, constant-query LDC's do not exist in the insertion/deletion setting. Subsequent research by Blocki et al. [BCG+22] makes progress toward this conjecture, demonstrating that any candidate LDC must have at least exponential length relative to n, but it does not definitively rule out their existence. Blocki et al.'s negative result holds even when the adversary can only perform deletions (without insertions).

In this work, we fully resolve Block et al.'s conjecture, establishing that constant-query LDC's of any length do not exist in the insertion/deletion setting. Like [BCG+22], our result holds when the adversary can only perform deletions. Moreover, due to a reduction shown by [BCG+22], this implies that locally correctable codes (LCC's) do not exist in the deletion setting.

1.1 Our Results

Our main result is that constant query locally decodable codes of any length do not exist in the deletion setting. We refer the reader to Definition 3.1 for a formal definition of locally decodable codes for the deletion channel.

Theorem 1.1. *For any $\varepsilon > 0$, $k \in \mathbb{N}$, and alphabet Σ, there exists a constant $C := C(\varepsilon, k, |\Sigma|)$ such that for all $n > C$, there is no k-query deletion LDC $\mathsf{C} : \Sigma^n \to \Sigma^M$ for any M that is resilient to ε-fraction of deletions.*

We conjecture a stronger version of Theorem 4.1, which we hope to see proven or disproven in future work.

[1] The number of queries is known as the locality, and although it is sometimes useful to explore parameter regimes where the locality is super-constant (for example polylog(n)), in this paper we focus on codes with constant locality.

Conjecture 1.2. *For any $\varepsilon > 0$ and $k \in \mathbb{N}$, and alphabet Σ, there exists a constant $C := C(\varepsilon, |\Sigma|)$ such that for all $n > Ck$, there is no non-adaptive k-query deletion LDC $\mathsf{C} : \Sigma^n \to \Sigma^M$ that is resilient to ε-fraction of adversarial deletions.*

In other words, we conjecture that any LDC for deletions requires a number of queries directly proportional to the length of the original message to recover a single message symbol. We emphasize that this conjectured lower bound only applies when all queries to a candidate LDC must be specified non-adaptively. When the queries can be submitted adaptively, prior results [OPC15, BBG+20] demonstrate that polylog(n) queries is achievable.

Theorem 4.1 also implies that locally correctable codes do not exist in the deletion setting. The implication follows from a result in [BCG+22], which states that the existence of LCC's for insertions/deletions would imply LDC's for insertions/deletions. We remark that they present this reduction for adversarial codes allowing both insertions and deletions, but their proof also works in the deletion-only setting.

Corollary 1.3. *For any $\varepsilon > 0$, $k \in \mathbb{N}$, and alphabet Σ, there exists a constant $C := C(\varepsilon, k, |\Sigma|)$ such that for all $n > C$, there is no k-query deletion LCC $\mathsf{C} : \Sigma^n \to \Sigma^M$ for any M that is resilient to ε-fraction of adversarial deletions.*

1.2 Related Works

Locally decodable codes for insertion/deletions. Locally decodable codes for insertions and deletions were introduced by Ostrovsky and Paskin-Cherniavsky in [OPC15]. They present deletion LDC's of length $\tilde{O}(n)$, resilient against a constant fraction of adversarial deletions with polylog(n) queries. Cheng, Li, and Zheng [CLZ20] introduce locally decodable codes with randomized encoding and demonstrate improved rate-query tradeoffs in this setting. [BBG+20] replicate [OPC15]'s results using different techniques and conjecture the non-existence of deletion LDCs of any length in the constant query regime.

As previously mentioned, Blocki et al. [BCG+22] make progress on this conjecture, proving that any candidate LDC must have exponential length relative to n. More specifically, they eliminate the possibility of 2-query linear deletion LDCs and establish an exponential lower bound for the size of any 2-query deletion LDC. In general, they prove a lower bound of $\exp(n^{1/O(k)})$ for a candidate k-query LDC.

[BBC+22] introduces the relaxed local decoding model (RLDC's) to the insertion/deletion setting, which allows limited decoding failure answers $\perp$. They show lower bounds for constant query "strong" RLDC's and constant query constructions for "weak" RLDC's. A few other models for insertion/deletion LDC's have been explored, including the secret key setting in [BB21, BCG+22].

Insertion/deletion codes. The study of codes that correct insertions and deletions was initiated by [Lev66]. Correcting for insertions and deletions is generally more challenging than substitutions, and our understanding of the field remains limited. Nevertheless, efficient constructions of constant rate codes that correct a constant fraction of insertion/deletion errors have been shown, initially in [SZ99]. More recently, the work of [HS17] introduces synchronization strings, which transform substitution error-correcting codes into insertion/deletion error-correcting codes in an efficient black-box manner. The optimal error resilience of constant rate insertion/deletion codes was studied by [GHL22].

Many other aspects of insertion/deletion codes have been explored. For more details, we refer to surveys on this topic, including [Mit09, HS21].

LDC's and LCC's for substitutions. Locally decodable codes and locally correctable codes were initially introduced in the classical setting of substitution errors. The ideas behind LDC's and LCC's arose as early as Reed-Muller codes [Ree54, Mul54] and were first formalized by Katz and Trevisan [KT00]. They have applications in many areas, including in PCP's [BFLS91], private information retrieval [CKGS98], and average case complexity [Tre04].

The best constructions of LDC's and LCC's for constant queries are barely sub-exponential in the length of the message [Efr09], while lower bounds are polynomial for $k > 3$ queries [KT00, WDW05, Woo07, AGKM23], except in the case of 2-query LDC's and LCC's [GKST02, KDW03] and linear 3-query LCC's [KM23]. The question of whether constant query LDC's with polynomial message length exist remains open. We refer the reader to [Yek12] for a survey of the work on LDC's.

2 Technical Overview

In this section, we'll overview our main result that constant query deletion LDC's do not exist. We'll restrict our attention to the case where the alphabet $\Sigma = \{0, 1\}$ as the proof is the same for any constant-sized alphabet.

Before delving into the details, let's set up the problem. Given a sufficiently small $\varepsilon > 0$ and a constant number of queries k, our goal is to show the existence of a constant $C(\varepsilon, k)$ such that for all $n > C(\varepsilon, k)$, there is no function $\mathsf{C} : \{0,1\}^n \to \{0,1\}^M$ for any value of M, which satisfies local decoding for deletions. That is, the adversary has a randomized corruption process $\mathsf{C}(x) \to \widehat{\mathsf{C}(x)}$ using at most ε-fraction corruption with the following guarantee. For any candidate local decoding algorithm that non-adaptively queries k indices of $\widehat{\mathsf{C}(x)}$ to guess $x[i]$, there exists an index i and some x for which it fails with probability greater than $\frac{1}{2} - \varepsilon$.

Throughout the overview, we use $\widehat{q}$ to denote the index of a query in the corrupted codeword $\widehat{\mathsf{C}(x)}$, while q represents its corresponding index in the original uncorrupted codeword $\mathsf{C}(x)$. For convenience, we'll use τ to represent a generic constant much smaller than ε (around $\mathrm{poly}(\varepsilon)$).

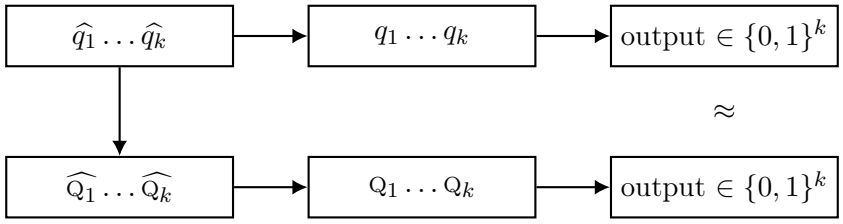
2.1 Reduction to Simulating All Queries with a Fixed Constant-Sized Set

Our main goal will be to show the following: the adversary can perform deletions in such a way that there is a *constant-sized* set $\widehat{\mathcal{Q}}$ of lists of queries $(\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k})$ such that any pair of queries $(\widehat{q_1} \dots \widehat{q_k})$ can be simulated by a corresponding pair of queries $(\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k}) \in \widehat{\mathcal{Q}}$. That is, for most x, the distribution over $\{0,1\}^k$ of the output of queries $(\widehat{q_1} \dots \widehat{q_k})$ is approximately the same (TV distance $< \tau$) as the output of $(\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k})$ over the randomness of the corruption process.

Pictorially, we want to construct $\widehat{\mathcal{Q}}$ so that for any list of queries $(\widehat{q_1} \dots \widehat{q_k})$, the following two processes yield roughly the same distribution of outputs for most $\mathsf{C}(x)$.

(1) Perform the randomized deletion pattern, then query $(\widehat{q_1} \dots \widehat{q_k})$ so that the induced indices of the queries in the original codeword are $(q_1 \dots q_k)$, and receive as output the values $\mathsf{C}(x)[q_1] \dots \mathsf{C}(x)[q_k]$.

(2) Choose the representative query list $(\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k}) \in \widehat{\mathcal{Q}}$ that approximates $(\widehat{q_1} \dots \widehat{q_k})$. Then, perform the randomized deletion pattern, query $(\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k})$ so that the induced indices of the queries in the original codeword are $(\mathsf{Q}_1 \dots \mathsf{Q}_k)$, and receive as output the values $\mathsf{C}(x)[\mathsf{Q}_1] \dots \mathsf{C}(x)[\mathsf{Q}_k]$.



If we can successfully establish this statement, then any local decoding algorithm can be emulated by one that only queries lists in $\widehat{\mathcal{Q}}$. Consequently, we can conclude that constant query deletion LDCs do not exist, as querying a fixed constant-sized list provides only a constant amount of information about x, while the number of choices for the index i is super-constant. A similar information-theoretic argument also shows that constant query deletion LCC's do not exist.

For the remainder of this overview, we will show how to design the adversary's corruption process and find this constant-sized $\widehat{\mathcal{Q}}$. We will begin with the 2-query case which illustrates many of the ideas, and then we will describe how to modify the argument for the general k-query case.

2.2 The 2-Query Case

2.2.1 The Corruption Process.

We'll begin by defining the corruption process used by the adversary. This is the same process used in the work [BCG+22]. The adversary performs two types of corruptions to $\mathsf{C}(x)$.[2]

- **Type 1:** Choose $\varepsilon_1 \in [0, \frac{\varepsilon}{2}]$ uniformly at random and delete the first $\varepsilon_1 M$ bits of $\mathsf{C}(x)$.
- **Type 2:** Choose $\varepsilon_2 \in [0, \frac{\varepsilon}{2}]$ uniformly at random and delete every $\frac{1}{\varepsilon_2}$'th remaining bit.

2.2.2 Constructing $\widehat{\mathcal{Q}}^{(\ell)}$ for Each Layer ℓ.

We'll first attempt to construct $\widehat{\mathcal{Q}}$ so that it satisfies an even stronger guarantee than that the output in $\{0,1\}^2$ of any pair $(\widehat{q_1}, \widehat{q_2})$ can be approximated by some pair $(\widehat{\mathsf{Q}_1}, \widehat{\mathsf{Q}_2}) \in \widehat{\mathcal{Q}}$. We'll have $\widehat{\mathcal{Q}}$ contain, for every possible pair of queries $(\widehat{q_1}, \widehat{q_2})$, a corresponding pair $(\widehat{\mathsf{Q}_1}, \widehat{\mathsf{Q}_2}) \in \widehat{\mathcal{Q}}$ such that the distribution of (q_1, q_2) is similar to $(\mathsf{Q}_1, \mathsf{Q}_2)$. This is essentially what [BCG+22] do, and the argument we outline below is a reformulation of theirs.

We will show that if $\widehat{\mathcal{Q}}$ contains $\widehat{\mathsf{Q}_1} \in \left[\widehat{q_1} \pm \tau M\right]$ and $\widehat{\mathsf{Q}_2} \in \left[\widehat{q_2} \pm \tau(\widehat{q_2} - \widehat{q_1})\right]$, then the statement holds true. With just Type 1 deletions, shifting a query pair $(\widehat{q_1}, \widehat{q_2})$ by $(\tau M, \tau M)$ maintains the distribution of the q_1 and q_2. This is because the small difference in the indices queried is counteracted by the unpredictability of ε_1. Similarly, Type 2 deletions counteract a small change in the distance between $\widehat{q_1}$ and $\widehat{q_2}$, so they can be shifted by $(0, (\widehat{q_2} - \widehat{q_1})\tau)$ without changing the output by much.

[2] We remark that the formal deletion process described in Section 4.2 even when restricted to the $k = 2$ case is different and more complex, but that this simpler process used in the overview permits the same analysis specifically for $k = 2$.

Notice that a "random" deletion process, where every index is deleted with probability $\varepsilon/2$, doesn't work here, because the unpredictability of the index q_1 is proportional only to $\sqrt{q_1}$ and not to M. Even if the deletion fraction is chosen randomly instead of being $\varepsilon/2$, the unpredictability of q_2 conditioned on q_1 is only proportional to $\sqrt{\widehat{q_2} - \widehat{q_1}}$ and not to $(\widehat{q_2} - \widehat{q_1})$.

Now, we want to construct $\widehat{\mathcal{Q}}$ so that it has a pair $\widehat{\mathsf{Q}_1} \in \left[\widehat{q_1} \pm \tau M\right]$ and $\widehat{\mathsf{Q}_2} \in \left[\widehat{q_2} \pm \tau(\widehat{q_2} - \widehat{q_1})\right]$ for every $(\widehat{q_1}, \widehat{q_2})$. For a fixed "layer" ℓ of queries – that is queries satisfying $\ell := \lfloor \log(\widehat{q_2} - \widehat{q_1}) \rfloor$, only a constant number of queries need to be added to $\widehat{\mathcal{Q}}$ to approximate all queries in this layer. This is because only one option for $\widehat{q_1}$ is needed in each τM-length range, and only one option for $\widehat{q_2} - \widehat{q_1}$ in each $\tau\ell$-length range. To be explicit, any such pair of queries can be approximated by one of the form $(\tau M i, \tau M i + \tau 2^a j)$ for $i \in [\tau^{-1}], j \in [\tau^{-1}]$. We'll add all queries of this form into the set $\widehat{\mathcal{Q}}^{(\ell)}$.

If there were only a constant number of layers, that is if $\log M$ were a constant, then we would be done, as all queries can be approximated by queries in $\bigcup_{\ell \in [\log M]} \widehat{\mathcal{Q}}^{(\ell)}$. Unfortunately, $\log M$ need not be a constant, and in fact will certainly depend on n. The main goal of the rest of our argument is to show that we only need to include $\widehat{\mathcal{Q}}^{(\ell)}$ for a *constant number* of layers ℓ, and these alone will approximate *all* queries. This constant-sized set of layers will be represented as $\mathcal{L}$.

We remark that this argument suffices for the bound in [BCG+22]. It is actually enough that $|\widehat{\mathcal{Q}}| = o(n)$ to achieve the information-theoretic contradiction in the previous section, so they establish a contradiction whenever $\log(M) = o(n)$, which yields their exponential lower bound.

2.2.3 A Constant Number of Layers ℓ Suffice.

If we let $\widehat{\mathcal{Q}} = \bigcup_{\ell \in [\log M]} \widehat{\mathcal{Q}}^{(\ell)}$, it is not only true that $(\widehat{q_1}, \widehat{q_2})$ shares the same distribution of outputs in $\{0,1\}^2$ as their representative queries in $\widehat{\mathcal{Q}}$, but they also correspond to a similar distribution on indices of the uncorrupted codeword $\mathsf{C}(x)$. As explained earlier, this is stronger than necessary. We'll identify layers ℓ that cen be excluded because they induce the same output distribution as other layers, even if not the same distribution on indices of the uncorrupted codeword $\mathsf{C}(x)$.

An illustrative example. Let's with an example. Look at the following string:

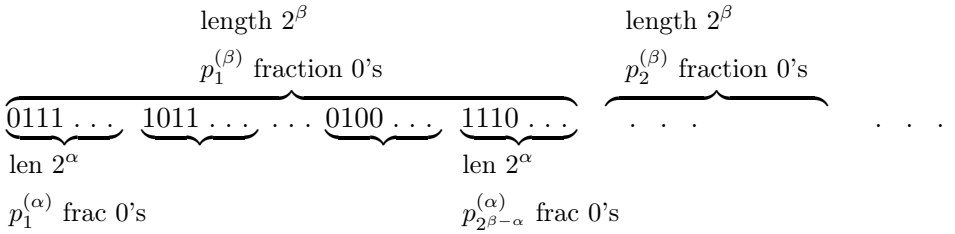
$$\overbrace{\underbrace{00\ldots0}_{\sqrt{n}\text{ 0's}}\ \underbrace{11\ldots1}_{\sqrt{n}\text{ 1's}}\ \cdots\ \underbrace{00\ldots0}_{\sqrt{n}\text{ 0's}}\ \underbrace{11\ldots1}_{\sqrt{n}\text{ 1's}}}^{\sqrt{n}\text{ blocks}}$$

In layers where $\widehat{q_2} - \widehat{q_1} \ll \sqrt{n}$, any two queries $(\widehat{q_1}, \widehat{q_2})$ have a very high probability of being identical. This is because the random shift from the Type 1 deletions mean that q_1 (the induced index of $\widehat{q_1}$ in the original codeword) is fairly uniform within the block it falls into, making it likely that q_2 is in the same block. Thus, slightly changing the gap between the queries does not significantly affect the output distribution, and hence, the set $\widehat{\mathcal{Q}}^{(1)}$ accounts for all $\widehat{\mathcal{Q}}^{(\ell)}$ with $2^\ell \ll \sqrt{n}$.

A similar argument works for queries where $\widehat{q_2} - \widehat{q_1} \gg \sqrt{n}$. In this case, q_1 and q_2 will be in different blocks with many blocks between them. In particular, Type 2 errors make it approximately equally likely for there to be an odd or even number of blocks between $\widehat{q_1}$ and $\widehat{q_2}$, and so their values are uncorrelated. Since this is true for all layers where $2^\ell \gg \sqrt{n}$, any single $\widehat{\mathcal{Q}}^{(\ell)}$ correctly models all the queries in these layers.

Hence, to model all queries, $\mathcal{L}$ would need only to include $\ell = 1$, one value of ℓ much larger than $\log \sqrt{n}$, and a few values of ℓ near $\log \sqrt{n}$. A more detailed analysis shows that only constantly many layers of the third type are necessary. This results in a constant sized union $\widehat{\mathcal{Q}}$.

Surprisingly, this phenomenon is not limited to the specific example shown; most strings exhibit a similar behavior. For a general string, consider the fraction of 0's and 1's in each block of size 2^α and 2^β for some $\beta \gg \alpha$.



The main claim is as follows. Assume that most blocks of size 2^α within the same 2^β chunk i have approximately the same fraction of 0's (i.e. $p_{2^\beta i+j}^{(\alpha)}$ is similar for all $j \in [2^{\beta-\alpha}]$, which also implies all $p_{2^\beta i+j}^{(\alpha)}$ are approximately equal to the mean $p_i^{(\beta)}$). Then only a single layer $\ell \in [\alpha + \tau^{-1}, \beta - \tau^{-1}]$ needs to be included in $\mathcal{L}$ (within this layer, it holds that $2^\alpha \ll \widehat{q}_2 - \widehat{q}_1 \ll 2^\beta$).

To illustrate this, consider a pair of queries $(\widehat{q}_1, \widehat{q}_2)$. Conditioned on q_1 falling within a specific size 2^β chunk i (and q_2 will likely fall in the same chunk since $\widehat{q}_2 - \widehat{q}_1 \ll 2^\beta$), we'll show that the distribution of the output of queries $(\widehat{q}_1, \widehat{q}_2)$ is $\mathrm{Bern}(p_i^{(\beta)})^2$.[3] Because $\widehat{q}_2 - \widehat{q}_1 \gg 2^\alpha$, the Type 2 errors cause enough unpredictability in $q_2 - q_1$ that both q_1 and q_2 can be considered essentially independent and uniformly distributed within the size 2^α chunk that each falls into (even though which size 2^a each query falls into can be correlated). We've assumed that each 2^α block has distribution $\mathrm{Bern}(p_i^{(\beta)})$ for a query made randomly to it, so the joint distribution on $(\widehat{q}_1, \widehat{q}_2)$ is $[\mathrm{Bern}(p_i^{(\beta)})]^2$.

Consequently, the output is essentially the same for all $\ell \in [\alpha + \tau^{-1}, \beta - \tau^{-1}]$ within the 2^β-sized block of the original codeword that the queries fall into. In particular, any query $(\widehat{q}_1, \widehat{q}_1 + D)$ exhibits almost an identical same output distribution as long as $\alpha + \tau^{-1} < \log D < \beta - \tau^{-1}$. Therefore, it suffices to represent all of the queries in a single one of the layers ℓ rather than all of them, which is adequately done by one $\widehat{\mathcal{Q}}^{(\ell)}$.

A structure lemma. It may seem like a strong assumption that most blocks of size 2^α within the same 2^β chunk have approximately the same fraction of 0's, but it is indeed the case that any string has only constantly many layers at which this fraction changes. Specifically, we'll identify a list $\alpha_1 < \ldots < \alpha_{\tau^{-1}}$ such that most blocks of size 2^{α_i} within the same $2^{\alpha_{i+1}}$ chunk have approximately the same fraction of 0's. Then, only one layer between $\alpha_i + 1$ and α_{i+1} needs to be included in $\mathcal{L}$ since they all then $\widehat{\mathcal{Q}}^{(\ell)}$ induce the same distribution of outputs.[4]

Consider the variance of $p^{(\alpha)}$ within a size 2^β chunk (for simplicity of notation let's focus on the first size 2^β chunk).

$$\operatorname*{Var}_{i \sim [2^{\beta-\alpha}]} \left(p_i^{(\alpha)} \right) = \operatorname*{\mathbb{E}}_{j \sim [2^{\beta-\alpha}]} \left[(p_j^{(\alpha)})^2 \right] - \left(p_1^{(\beta)} \right)^2.$$

[3] We use $\mathrm{Bern}(p)$ for the distribution on $\{0, 1\}$ with probability p of being 0 and probability $1 - p$ of being 1.

[4] This is not quite true; actually a constant number of extra layers right around $\alpha_i + 1$ and α_{i+1} need to be included.

We want to characterize the ranges in which this variance is large. For any $\theta \in [\log M]$, define the weight $\mathcal{W}_\theta := \mathbb{E}_{i\sim[M/2^\theta]}\left[p_i^{(\theta)}\right]$. Then, averaging the above variance (of 2^α sized blocks) over all the sized 2^β chunks gives

$$\begin{aligned}\mathop{\mathbb{E}}_{i\in[M/2^\beta]}\left[\mathop{\mathrm{Var}}_{j\sim[2^\beta i+1,2^\beta i+2^{\beta-\alpha}]}\left(p_j^{(\alpha)}\right)\right] &= \mathop{\mathbb{E}}_{i\in[M/2^\beta]}\left[\mathop{\mathbb{E}}_{j\sim[2^\beta i+1,2^\beta i+2^{\beta-\alpha}]}\left[(p_j^{(\alpha)})^2\right]-(p_i^{(\beta)})^2\right].\\ &= \mathcal{W}_\alpha - \mathcal{W}_\beta\end{aligned}$$

In simpler terms, we have found a weight function on layers $\mathcal{W}_\theta$ such that the variance of the fraction of 0's in a sized 2^α block in a 2^β chunk is $W_\alpha - W_\beta$. Moreover, this weight $\mathcal{W}_\theta$ decreases as θ increases and always falls between 0 and 1. Therefore, the string can be partitioned into a constant length list $\alpha_1 \dots \alpha_{\tau^{-1}}$ such that for all i, $\mathcal{W}_{\alpha_{i+1}} - \mathcal{W}_{\alpha_i+1} \ll \varepsilon$ and so only a single layer in each interval needs to be included.

Reconciling with C. Recall that $\widehat{\mathcal{Q}}$ needs to be a function of the code C, not each individual string the sender could send. To deal with this, one can instead use $\mathcal{W}_\theta(\mathsf{C}) := \mathbb{E}_{z\in\mathsf{C}}\left[\mathcal{W}_\theta(z)\right]$, and find the relevant $\alpha_1 \dots \alpha_{\tau^{-1}}$ according to this weighting. Then, on most strings $z \in \mathrm{Im}(\mathsf{C})$, layers between adjacent elements of the sequence will induce the same output distribution, which is enough for the argument to work.

2.3 The k-Query Case

Next, we'll extend this argument to the k-query case.

2.3.1 Naive Attempt to Extend 2-Query Argument.

Ideally, the same corruption pattern and a similar analysis would still apply. We had identified a sequence of critical transition layers $\alpha_1 \dots \alpha_{\tau^{-1}}$ using the weight function defined above, and based on this, pinpointed a constant-sized list of layers $\mathcal{L}$ that adequately approximated all pairs of queries. We can attempt to extend this by defining $\widehat{\mathcal{Q}}^{(\ell_1\dots\ell_{k-1})}$ for $\ell_1 \dots \ell_{k-1} \in \mathcal{L}$ as follows: choose $j, i_1 \dots i_{k-1} \in [\tau^{-1}]$ and include each tuple of the form $\left(j\tau M, j\tau M + i_1 2^{\ell_1}, \dots, j\tau M + i_1 2^{\ell_1} + \dots + i_{k-1}2^{\ell_{k-1}}\right)$ in $\widehat{\mathcal{Q}}^{(\ell_1\dots\ell_{k-1})}$.

To carry through the 2-query analysis, we would need to show the following two claims:

(1) Any $(\widehat{q_1}, \dots, \widehat{q_k})$ satisfying $\widehat{q_{i+1}} - \widehat{q_i} \approx 2^{\ell_i}$ induces the same distribution on $(q_1, \dots, q_k)$ as some list in $\widehat{\mathcal{Q}}^{(\ell_1\dots\ell_{k-1})}$.

(2) Any successive difference between queries $(\widehat{q_{i+1}} - \widehat{q_i})$ that does not correspond to a layer in $\mathcal{L}$ is suitably approximated by a layer in $\mathcal{L}$.

Unfortunately, neither of these claims is true as stated.

2.3.2 Addressing the First Issue.

To see this issue with (1), consider the following triples of queries for some fixed q, d where $d \approx 2^\ell$ for some layer $\ell \in \mathcal{L}$: $(\widehat{q_1} := q, \widehat{q_2} := q+d, \widehat{q_3} := q+2d)$ and $(\widehat{q_1'} := q, \widehat{q_2'} := q+d, \widehat{q_3'} := q+2d+10)$. We want the distributions of (q_1, q_2, q_3) and (q_1', q_2', q_3') to be similar. However, since $\widehat{q_3} - \widehat{q_2} = \widehat{q_2} - \widehat{q_1}$,

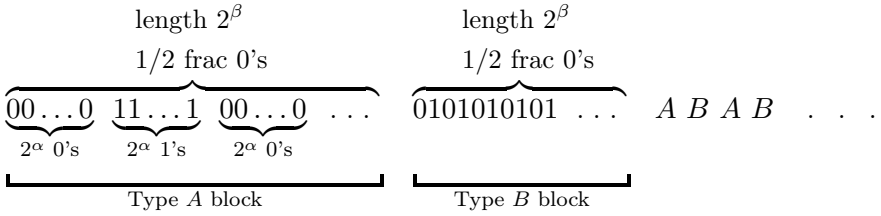
any induced indices (q_1, q_2, q_3) always take on the form $(Q, Q+D, Q+2D)$ for some Q, D since Type 2 deletions delete the same fraction of each equal-length interval. By similar logic (q_1', q_2', q_3') takes on the form $(Q, Q+D, > Q+2D+5)$. Consequently, not only are the induced distributions of (q_1, q_2, q_3) and (q_1', q_2', q_3') different, but they have no overlap, as the triples take on different forms.

The key issue here is that every interval of length D in a codeword has the same number of bits deleted, while the desired behavior is to have a random, uncorrelated number of bits deleted. To achieve this, we'll modify the deletion process as follows: for every layer $\ell \in \mathcal{L}$, in the i'th chunk of length 2^ℓ, choose a random fraction $\varepsilon_i^{(\ell)}$ and delete every $\frac{1}{\varepsilon_i^{(\ell)}}$'th bit. Since there are only a constant number of layers in $\mathcal{L}$, this still results in a constant fraction of corruption overall. Now, each interval of length $\approx 2^\ell$ not only has a random fraction of bits deleted, but these fractions are uncorrelated. Consequently, the example from the previous paragraph would induce the same distribution on (q_1, q_2, q_3) and (q_1', q_2', q_3') as originally intended.

2.3.3 Addressing the Second Issue.

The issue with (2) is that $\mathcal{L}$ doesn't actually encompass all the important layers. We will show that there are important transition layers that are not accounted for by the weight function $\mathcal{W}$ defined previously. In the final argument, the number of transition layers will still be constant, but we need a new weight function to identify additional important layers.

An illustrative example. Let's begin with an example that demonstrates why $\mathcal{W}$ is inadequate.



In this example, the string consists of $M/2^\beta$ alternating Type A and Type B blocks. At each scale $\theta \in [\log M]$, let's examine the weight $\mathcal{W}_\theta := \mathbb{E}_{i \sim [M/2^\theta]}\left[p_i^{(\theta)}\right]$. For all $\theta > \alpha$, this quantity is $1/4$, so if $\mathcal{W}$ were sufficient, differences between queries that are $\gg 2^\alpha$ would need to behave essentially the same. However, it is evident that the structure of the string chanes around the scale 2^β. Two queries weren't sufficient to detect this change, but it turns out that four queries are.

To be concrete, choose D_1 and D_2 satisfying $2^\alpha \ll D_1 \ll 2^\beta$ and $D_2 \gg 2^\beta$. Make the following two sets of queries: $(0, 1, D_1, D_1+1)$ and $(0, 1, D_2, D_2+1)$. The aim is for the distribution on outputs to be the same since D_1 and D_2 correspond to supposedly indistinguishable layers. However, when a pair of queries differing by 1 falls in a type A block, the outputs are typically the same, and when such a pair falls in a type B block, the outputs are typically different. In the query $(0, 1, D_1, D_1+1)$, the first pair is likely in the same length 2^β block as the second pair (so whether the first pair are the same is correlated to whether the second pair are the same). However, in $(0, 1, D_2, D_2+1)$, the first and second pair are likely in different, uncorrelated blocks. Therefore, the distribution of outputs in $\{0,1\}^4$ for the queries $(0, 1, D_1, D_1+1)$ and $(0, 1, D_2, D_2+1)$ are different.

Defining more weight functions. In this way, layer β is a transition point where smaller layers look substantially different than larger ones, and it's not accounted for by $\mathcal{W}$. Our new weighting

needs to say something about what happens when you make *two* queries to this region rather than just considering the fraction of 0's and 1's. Given any pair of queries $(\widehat{q_1}, \widehat{q_2})$, define the weight:

$$\mathcal{W}_\theta^{(00)}(\widehat{q_1}, \widehat{q_2}) := \mathop{\mathbb{E}}_{\substack{i \sim [M/2^\theta] \\ c \sim [2^\theta]}} \left[\left(\Pr\left(\widehat{q_1} + c, \widehat{q_2} + c \right) \text{ outputs } 00 \right)^2 \right].$$

We similarly define $\mathcal{W}^{(01)}$, $\mathcal{W}^{(10)}$, and $\mathcal{W}^{(11)}$, and if the alphabet Σ were larger than binary, we would define $\mathcal{W}^{(\sigma_1\sigma_2)}$ for all $\sigma_1, \sigma_2 \in \Sigma$, but we'll focus the discussion on $\mathcal{W}^{(00)}$. In our earlier example, observe that layers smaller than β have significantly higher weight than layers larger than β for $\widehat{q_1} = 0$ and $\widehat{q_2} = 1$, so this successfully addresses the issue. Furthermore, for a fixed $(\widehat{q_1}, \widehat{q_2})$, this weight is also (approximately) decreasing with respect to θ, so it has a constant number of transition points. Consequently, we can use the constant-sized set of transition points for $\mathcal{W}_\theta^{(00)}(\widehat{q_1}, \widehat{q_2})$ and correspondingly add a constant number more layers into $\mathcal{L}$ for each $(\widehat{q_1}, \widehat{q_2})$. As it turns out, this new $\mathcal{L}$ successfully accounts for all the relevant layers, at least when $k = 3$.

One issue remains: there are super-constantly many pairs $(\widehat{q_1}, \widehat{q_2})$, but we can only add a constant number of layers to $\mathcal{L}$. This is easily addressed by utilizing only a representative set of $(\widehat{q_1}, \widehat{q_2})$ whose elements approximate all possible queries that could be made. This is exactly what the set $\widehat{\mathcal{Q}}$ calculated for the 2-query case represents (so we only need to use $\mathcal{W}_\theta^{(00)}(\widehat{q_1}, \widehat{q_2})$ for pairs of queries in the original $\widehat{\mathcal{Q}}$).

In general, for each of the k queries, we define an additional recursive layer of weightings. In this manner, we construct the set $\mathcal{L}$ of relevant layers based on those relevant to any weighting, and include every corresponding $\widehat{\mathcal{Q}}^{(\ell_1 \ldots \ell_{k-1})}$. We refer the reader to Section 4 for a comprehensive analysis. It is worth noting that the formal analysis in Section 4 does not define $\widehat{\mathcal{Q}}$ in in the same way, but the ideas remain the same.

3 Preliminaries

3.1 Model Definitions

We'll start with the definition of locally decodable codes (LDC's) that are resilient to ε-fraction of adversarial deletions.

Definition 3.1 (Deletion LDC)**.** For a fixed value of n and alphabet Σ, a candidate deletion LDC is an encoding $\mathsf{C} : \Sigma^n \to \Sigma^M$. We say that C is resilient to ε-fraction of adversarial deletions as a q-query deletion LDC (for shorthand, just k-query ε-resilient LDC) if the following holds.

For all $i \in [n]$, there exists an algorithm that makes k queries[5] to a codeword $\mathsf{C}(x)$ adversarially[6] corrupted by an adversary using less than εM deletions and with probability at least $\frac{1}{2} + \varepsilon$ correctly guesses $x[i]$ for all values of $x \in \Sigma^n$.

Next, we'll define locally decodable codes (LCC's) that are resilient to ε-fraction of adversarial deletions.

[5] Arbitrary indices of $\widehat{\mathsf{C}(x)}$ can be queried, and if the index is out of bounds, the query will return 0.

[6] The adversary knows the encoding C and x but not the decoder's algorithm.

Definition 3.2 (Deletion LCC)**.** For a fixed value of n and alphabet Σ, a candidate deletion LCC is an encoding $\mathsf{C} : \Sigma^n \to \Sigma^M$. We say that C is resilient to ε-fraction of adversarial deletions as a q-query deletion LCC (for shorthand, just k-query ε-resilient LCC) if the following holds.

For all $i \in [M]$, there exists an algorithm that makes k queries to a codeword $\mathsf{C}(x)$ adversarially corrupted by an adversary using less than εM deletions and with probability at least $\frac{1}{2} + \varepsilon$ correctly guesses $\mathsf{C}(x)[i]$ for all values of $x \in \Sigma^n$.

3.2 Notation and Definitions

Notation.

- The set $[n]$ denotes the set $\{1 \dots n\}$.
- Logarithms are in base 2 unless otherwise specified.
- In general, messages will be denoted by x, encoded messages by $\mathsf{C}(x)$, and corrupted encoded messages by $\widehat{\mathsf{C}(x)}$.
- By default, if an algorithm makes queries $\widehat{q_1} \dots \widehat{q_r}$, the queries will be non-adaptive and in increasing order (so $\widehat{q_r} > \dots > \widehat{q_1}$). The induced indices of $\widehat{q_1} \dots \widehat{q_r}$ in the uncorrupted codeword are $q_1 \dots q_r$.

First, we define a (randomized) deletion pattern that the adversary can perform on a string with indices $1 \dots n$.

Definition 3.3 (Deletion Pattern)**.** A deletion pattern del on the indices $1 \dots \nu$ is a probability distribution over subsets of $1 \dots n$. Usually, in a string of length ν, the bits at the indices corresponding to the subset will be deleted.

For the context of our next definition, consider the situation after Bob has received a corrupted codeword and queries the indices $\widehat{q_1} \dots \widehat{q_k}$.

Definition 3.4 ($\overline{\mathcal{D}}(S)$)**.** For a measurable space S, the family of probability distributions over S is denoted by $\overline{\mathcal{D}}(S)$.

Definition 3.5 ($\mathcal{D}(\widehat{q_1}, \dots, \widehat{q_k}; \mathsf{del})$)**.** For a deletion pattern del on strings of length n, given that Bob has queried $\widehat{q_1}, \dots, \widehat{q_k}$, let the distribution of $q_1, \dots q_k$ (indices of the original string) be denoted $\mathcal{D} := \mathcal{D}(\widehat{q_1}, \dots, \widehat{q_k}; \mathsf{del})$.

For a distribution $\mathcal{D}$ and string z of length n, the distribution $z[\mathcal{D}]$ denotes the distribution of Σ^* obtained by indexing the tuple $\mathcal{D}$ into the string z. If a queried index in $\mathcal{D}$ are out of bounds for z (i.e. not in the range $1 \dots n$), then instead return err (so the distribution also supports some error rather than only valid outputs in Σ^r.

Next, we'll define an easy way to view a string z as a concatenation of chunks.

Definition 3.6 ($C_{i,\nu}(z)$)**.** Consider a string z whose length is a multiple of ν (specifically, length $\alpha\nu$). For $i \in [\alpha]$, we define $C_{i,\nu}(z)$ to be the i'th length ν chunk of length of the string z.

4 Impossibility of k-Query Deletion LDC's

In this section, we establish the main theorem, that constant query locally decodable codes for deletion errors do not exist.

Theorem 4.1. *For any $\varepsilon > 0$, $k \in \mathbb{N}$, and alphabet Σ, there exists a constant $C := C(\varepsilon, k, |\Sigma|)$ such that for all $n > C$, there is no k-query deletion LDC $\mathsf{C} : \Sigma^n \to \Sigma^M$ for any M that is resilient to ε-fraction of deletions.*

Moreover, we establish that constant query LCC's do not exist.

Corollary 4.2. *For any $\varepsilon > 0$, $k \in \mathbb{N}$, and alphabet Σ, there exists a constant $C := C(\varepsilon, k, |\Sigma|)$ such that for all $n > C$, there is no k-query deletion LCC $\mathsf{C} : \Sigma^n \to \Sigma^M$ for any M that is resilient to ε-fraction of adversarial deletions.*

We'll show these theorems by considering any candidate locally decodable code C and describing a corruption pattern for which some index i cannot be recovered with only k queries. Throughout this section, ε, k and n are fixed, and we'll let κ be the smallest power of 2 greater than $\varepsilon^{-k \cdot |\Sigma|}$. Let C denote the candidate LDC (encoding strings of length n to length $M = \kappa^m$), x denotes a message being sent, and z typically is an arbitrary string of length M.

We remark that the assumption that M is a power of κ is almost without loss of generality. This is because any encoding is effectively padded with infinitely many 0's since out-of-bounds queries return 0. To make the code's length a power of κ, simply increase the length of the code until the next power of 2 larger than M which at most doubles the total length. Then, any adversarial attack on the new padded encoding also works on the original (since deleting indices larger than M doesn't change the string) and requires (at most) the same number of deletions, but since M increased in length by a factor of up to κ, we will require error resilience $\varepsilon\kappa^{-1}$ instead of ε.

4.1 Definitions and Useful Lemmas

We begin with a few useful definitions and simple facts. Throughout, the length of an arbitrary string ν will be assumed to be a power of κ (so that it divides M).

4.1.1 Frequencies

When discussing lists of queries, we'll usually be interested in the list of differences between successive queries rather than the queries themselves. In particular, for any list of queries $\widehat{q_1} \dots \widehat{q_r}$, we'll define the list of differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ as $d_i := \widehat{q_{i+1}} - \widehat{q_i}$. We'll often state lemmas about the lists of query differences rather than the queries themselves.

Our first definition describes the probability of the output being a specific pattern b when queries with differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ are made to a string s on which some probabilistic deletion pattern is performed. The queries will be randomly shifted in the string subject to the successive differences being $\widehat{d_1} \dots \widehat{d_{r-1}}$.

Definition 4.3 ($\mathsf{freq}^{(b)}(s; \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del})$)**.** We define the shift-invariant frequency $\mathsf{freq}^{(b)}(s; \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del})$ of a pattern $b \in \Sigma^r$ in a string $s \in \Sigma^\nu$ on the query differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ with deletion pattern del on strings of length ν as follows.

For a shift $i \in 1 \dots \nu$, let $\mathcal{D}_i := \mathcal{D}[i, i + \widehat{d_1} \dots i + \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}]$ Then,

$$\mathsf{freq}^{(b)}(s; \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}) := \mathop{\mathbb{E}}_{i \in [\nu]} \Big[\mathbb{P}\big[s(\mathcal{D}_i) = b\big] \Big].$$

Next, we define the error probability of querying a list with differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ in a string s. Specifically, there's some chance of outputting err because some of the indices in the queries are out of bounds when the shift is realized.

Definition 4.4 ($\mathsf{err}(s; \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del})$)**.** The error probability $\mathsf{err}(s; \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del})$ when querying $i, i + \widehat{d_1} \dots i + \widehat{d_1} + \dots + \widehat{d_{r-1}}$ for a uniformly random shift $i \in [\nu]$ on a string s with deletion pattern del is the probability that at least one index is out of bounds (and so the query returns err).

4.1.2 Approximation Equivalence of Queries

Next, we'll define what it means for a list of query differences $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}$ to approximate another list of differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ for a deletion pattern del.

Definition 4.5 (δ-Approximate Queries)**.** For a deletion pattern del on a string s of length ν, a list of query differences $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}$ δ-approximates another list of query differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ if $\widehat{\mathrm{D}_1} + \dots + \widehat{\mathrm{D}_{r-1}} \le \widehat{d_1} + \dots + \widehat{d_{r-1}}$ and for all $b \in \Sigma^r$,

$$\begin{aligned} \Big|\mathsf{freq}^{(b)}(s; \widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}; \mathsf{del}) - \mathsf{freq}^{(b)}(s; \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del})\Big| \\ + \mathsf{err}(s; \widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}; \mathsf{del}) + \mathsf{err}(s; \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}) \end{aligned} < \delta.$$

If s is length $\alpha\nu$ and del is a deletion pattern on strings of length ν, then we say that $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}$ δ-approximates $\widehat{d_1} \dots \widehat{d_{r-1}}$ if

$$\mathop{\mathbb{E}}_{i \in [\alpha]} \left[\begin{aligned} \Big|\mathsf{freq}^{(b)}(C_{i,\nu}(s); \widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}; \mathsf{del}) - \mathsf{freq}^{(b)}(C_{i,\nu}(s); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del})\Big| \\ + \mathsf{err}(C_{i,\nu}(s); \widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}; \mathsf{del}) + \mathsf{err}(C_{i,\nu}(s); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}) \end{aligned} \right] < \delta.$$

In other words, on the average chunk of length ν, $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}$ δ-approximates $\widehat{d_1} \dots \widehat{d_{r-1}}$.

4.1.3 Layered Deletion Patterns and Significant Layers

Next, we'll define a specific type of deletion pattern that we call a *layered* deletion pattern.

Definition 4.6 (Layered Deletion Pattern)**.** For a length $\nu = \kappa^a$, we define a layered deletion pattern on the indices $[\nu]$. Each layer $\alpha \in [a]$ is assigned a maximum deletion fraction described by a function $f_{\mathsf{del}} : \mathbb{N} \to \mathbb{R}$.

To perform the (randomized) deletions, for each layer $\alpha \in [a]$, the string is divided into chunks of length κ^α. Then, for each chunk, a uniformly random integer I between 0 and $\lfloor \varepsilon_\alpha \kappa^\alpha \rfloor$ is chosen, and the first I bits of the chunk are marked to be deleted. At the end, traverse the string left to write. Keep a counter that starts at 0, and at every index, if an index is marked as deleted t times, add t to the counter. Moreover, if the counter is greater than 0, delete the index and decrement the counter by 1.

The layered deletion pattern del_a is the pattern del restricted to layers $1 \dots a$. It is on strings of length κ^a instead of κ^M. The layered deletion pattern $\mathsf{del}_{a|A}$ is the pattern del_a extended to strings of length κ^A but still only incorporating the deletions in layers $1 \dots a$.

Next, we'll define significant layers in a code C. Loosely speaking, these are the values of a such that performing some fixed list of queries with differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ with a random shift of about κ^a generates different outputs from performing the queries with a random shift somewhat smaller than κ^a. These are the scales at which we'll need to insert random deletions.

Definition 4.7 (Significant Layers for Threshold ρ)**.** We define as follows a set $\mathcal{L}$ to be the significant layers for some threshold ρ given the code C, a pattern $b \in \Sigma^r$, query differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ and deletion pattern del.

The possible layers are $1 \dots m$, corresponding to partitions of a codeword into exponentially-sized intervals. The first $10\kappa^\kappa$ significant layers will be the one corresponding to layer $\tau = \lfloor(\widehat{d_1} + \dots + \widehat{d_{r-1}})\rfloor, \dots, \lfloor(\widehat{d_1} + \dots + \widehat{d_{r-1}})\rfloor + 10\kappa^\kappa - 1$. Every future layer ℓ_{i+1} will be constructed from ℓ_i as follows. It is the smallest possible $\ell_{i+1} \in [m]$ satisfying

$$\mathop{\mathbb{E}}_{z \in \mathrm{Im}(\mathsf{C}), j \in \left[\frac{M}{\kappa^{\ell_i}}\right]} \left[\left| \mathsf{freq}^{(b)}(C_{j', \kappa^{\ell_{i+1}}}(z); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}}) - \mathsf{freq}^{(b)}(C_{j, \kappa^{\ell_i}}(z); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_i}) \right| \right] > \rho$$

where $j' = \lfloor \frac{i}{\kappa^{\ell_{i+1} - \ell_i}} \rfloor$, so that $C_{j', \kappa^{\ell_i}}(z)$ is the length κ^{ℓ_i} chunk that contains $C_{j, \kappa^{\ell_{i+1}}}(z)$.

Loosely speaking, ℓ_{i+1} is the smallest layer significantly larger than ℓ_i such that queries performed at the scale ℓ_{i+1} look significantly different to those performed at the scale ℓ_i.

4.1.4 Useful Lemmas

Lemma 4.8. *Given a string $s \in \Sigma^*$ and a deletion pattern del on strings of length $\nu|s$, if a list of query differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ δ_1-approximates another such list $\widehat{d'_1} \dots \widehat{d'_{r-1}}$ and $\widehat{d'_1} \dots \widehat{d'_{r-1}}$ δ_2-approximates another list $\widehat{d''_1} \dots \widehat{d''_{r-1}}$, then $\widehat{d_1} \dots \widehat{d_{r-1}}$ $\delta_1 + \delta_2$-approximates $\widehat{d''_1} \dots \widehat{d''_{r-1}}$.*

Proof. This follows from the triangle inequality. Note that the condition is met that $\widehat{d_1} + \dots + \widehat{d_{r-1}} < \widehat{d''_1} + \dots + \widehat{d''_{r-1}}$. We have that

$$\mathop{\mathbb{E}}_{i \in [\alpha]} \left[\begin{array}{r} \left| \mathsf{freq}^{(b)}(C_{i,\nu}(s); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}) - \mathsf{freq}^{(b)}(C_{i,\nu}(s); \widehat{d'_1} \dots \widehat{d'_{r-1}}; \mathsf{del}) \right| \\ + \mathsf{err}(C_{i,\nu}(s); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}) + \mathsf{err}(C_{i,\nu}(s); \widehat{d'_1} \dots \widehat{d'_{r-1}}; \mathsf{del}) \end{array} \right] < \delta_1$$

and also that

$$\mathop{\mathbb{E}}_{i \in [\alpha]} \left[\begin{array}{r} \left| \mathsf{freq}^{(b)}(C_{i,\nu}(s); \widehat{d'_1} \dots \widehat{d'_{r-1}}; \mathsf{del}) - \mathsf{freq}^{(b)}(C_{i,\nu}(s); \widehat{d''_1} \dots \widehat{d''_{r-1}}; \mathsf{del}) \right| \\ + \mathsf{err}(C_{i,\nu}(s); \widehat{d'_1} \dots \widehat{d'_{r-1}}; \mathsf{del}) + \mathsf{err}(C_{i,\nu}(s); \widehat{d''_1} \dots \widehat{d''_{r-1}}; \mathsf{del}) \end{array} \right] < \delta_2.$$

Therefore, by the triangle inequality and the fact that the err function is always nonnegative,

$$\mathop{\mathbb{E}}_{i \in [\alpha]} \left[\begin{array}{r} \left| \mathsf{freq}^{(b)}(C_{i,\nu}(s); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}) - \mathsf{freq}^{(b)}(C_{i,\nu}(s); \widehat{d''_1} \dots \widehat{d''_{r-1}}; \mathsf{del}) \right| \\ + \mathsf{err}(C_{i,\nu}(s); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}) + \mathsf{err}(C_{i,\nu}(s); \widehat{d''_1} \dots \widehat{d''_{r-1}}; \mathsf{del}) \end{array} \right] < \delta_1 + \delta_2.$$

□

Lemma 4.9. *Consider a layered deletion pattern del on the code C whose total corruption is at most $2^{-\kappa}$. For a list of query differences $\widehat{d_1} + \dots + \widehat{d_{r-1}}$ and a threshold $\rho > 2^{-\kappa}$, there are at most $\kappa^\kappa + \rho^{-3}$ significant layers in C.*

Proof. Define $a := \lfloor(\widehat{d_1} + \ldots + \widehat{d_{r-1}})\rfloor$. The first κ^κ subsequent layers after a are all significant layers.

For any $b \in \Sigma^r$, $z \in \mathrm{Im}(\mathsf{C})$, and layer ℓ, we define the weight as follows.

$$\mathcal{W}_\ell^{(b)}(z; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) := \mathop{\mathbb{E}}_{i \in [\ell]} \left[\mathsf{freq}^{(b)}(C_{i,\kappa^\ell}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{a|\ell})^2 \right].$$

Moreover, we define the cumulative weight of the code C as follows:

$$\overline{\mathcal{W}}_\ell^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) := \mathop{\mathbb{E}}_{x \in \Sigma^n} \left[\mathcal{W}_\ell^{(b)}(\mathsf{C}(x); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) \right].$$

Assume for the sake of contradiction that there are more than $c = \rho^{-5}$ remaining significant layers aside from the first κ^κ, denoted $\ell_1 \ldots \ell_c$. Recall that a significant layer satisfies:

$$\mathop{\mathbb{E}}_{z \in \mathrm{Im}(\mathsf{C}), j \in \left[\frac{M}{\kappa^{\ell_i}}\right]} \left[\left| \mathsf{freq}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}}) - \mathsf{freq}^{(b)}(C_{j,\kappa^{\ell_i}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_i}) \right| \right] > \rho$$

where $i' = \lfloor \frac{i}{\kappa^{\ell_{i+1}-\ell_i}} \rfloor$, so that $C_{j',\kappa^{\ell_i}}(z)$ is the length κ^{ℓ_i} chunk that contains $C_{j,\kappa^{\ell_{i+1}}}(z)$.

This implies

$$\mathop{\mathbb{E}}_{z \in \mathrm{Im}(\mathsf{C}), j \in \left[\frac{M}{\kappa^{\ell_i}}\right]} \left[\left(\mathsf{freq}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}}) - \mathsf{freq}^{(b)}(C_{j,\kappa^{\ell_i}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_i}) \right)^2 \right] > \rho^2$$

by the QM-AM inequality. Expanding, we get the the left side is equal to

$$\begin{aligned}
&\overline{\mathcal{W}}_{\ell_i}^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) - \overline{\mathcal{W}}_{\ell_{i+1}}^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) \\
&+ \mathop{2\mathbb{E}}_{z \in \mathrm{Im}(\mathsf{C}), j \in \left[\frac{M}{\kappa^{\ell_i}}\right]} \left[\begin{array}{l} \mathsf{freq}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}}) \cdot \\ \left(\mathsf{freq}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}}) - \mathsf{freq}^{(b)}(C_{j,\kappa^{\ell_i}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_i}) \right) \end{array} \right]
\end{aligned}$$

Finally, note that within a block of length $\kappa^{\ell_{i+1}}$, the values of $\mathsf{freq}^{(b)}(C_{j',\kappa^{\ell_i}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_i})$, which just represent the probability of getting the output b on a randomly shifted query with the specified differences, average to $\mathsf{freq}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}})$ minus an error term corresponding to the chance that the output was err in the case of size κ^{ℓ_i} blocks due to out of bounds but is b when the block size is expanded, which we'll denote $\mathsf{e}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}})$. This term is nonnegative and at most the probability of an out of bounds error, which is at most $\kappa^{-\kappa^\kappa}$ since there are $-\kappa^\kappa$ layers before ℓ_i.

Thus, the above equation is equal to

$$\begin{aligned}
&\overline{\mathcal{W}}_{\ell_i}^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) - \overline{\mathcal{W}}_{\ell_{i+1}}^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) \\
&\quad + \mathop{2\mathbb{E}}_{z \in \mathrm{Im}(\mathsf{C}), j' \in \left[\frac{M}{\kappa^{\ell_{i+1}}}\right]} \left[\begin{array}{l} \mathsf{freq}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}}) \cdot \\ \mathsf{e}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}}) \end{array} \right] \\
&< \overline{\mathcal{W}}_{\ell_i}^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) - \overline{\mathcal{W}}_{\ell_{i+1}}^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) \\
&\quad + \mathop{2\mathbb{E}}_{z \in \mathrm{Im}(\mathsf{C}), j' \in \left[\frac{M}{\kappa^{\ell_{i+1}}}\right]} \left[\mathsf{e}^{(b)}(C_{j',\kappa^{\ell_{i+1}}}(z); \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\ell_{i+1}}) \right] \\
&< \overline{\mathcal{W}}_{\ell_i}^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) - \overline{\mathcal{W}}_{\ell_{i+1}}^{(b)}(\mathsf{C}; \widehat{d_1} \ldots \widehat{d_{r-1}}; \mathsf{del}) + \kappa^{-\kappa^\kappa}
\end{aligned}$$

Therefore,

$$\overline{\mathcal{W}}^{(b)}_{\ell_i}(\mathsf{C}; \widehat{d_1 \dots d_{r-1}}; \mathsf{del}) - \overline{\mathcal{W}}^{(b)}_{\ell_{i+1}}(\mathsf{C}; \widehat{d_1 \dots d_{r-1}}; \mathsf{del}) > \rho^2 - \kappa^{-\kappa^\kappa} > \rho^3.$$

If there are more than ρ^{-3} terms in the sequence ℓ_i, it would hold that $\overline{\mathcal{W}}^{(b)}_{\ell_1}(\mathsf{C}; \widehat{d_1 \dots d_{r-1}}; \mathsf{del}) - \overline{\mathcal{W}}^{(b)}_{\ell_c}(\mathsf{C}; \widehat{d_1 \dots d_{r-1}}; \mathsf{del}) > 1$, but all values of the weight function fall between 0 and 1, which is a contradiction. □

4.1.5 Lemmas for Proof of Lemma 4.15

The next three lemmas are the core of the proof of Lemma 4.15 in Section 4.3. They describe modifications that can be made to a list of queries $\widehat{d_1 \dots d_{r-1}}$ to make a new list of queries $\widehat{\mathrm{D}_1 \dots \mathrm{D}_{r-1}}$ that approximates the old one.

The first lemma essentially says that shifting a query difference by a small amount nearby a layer a where $f_{\mathsf{del}}(a) > 0$ does not affect the distribution of outputs.

Lemma 4.10. *Consider a string $z \in \Sigma^M$, a layered deletion pattern del on strings of length κ^a for some $a < m$ with total corruption fraction at most $2^{-\kappa}$, and query differences $\widehat{d_1 \dots d_{r-1}}$ satisfying $\widehat{d_1} + \ldots + \widehat{d_{r-1}} < \kappa^{a-\kappa^2}$. Fix $1 \le j < r$ and let $a' < \log_\kappa(\widehat{d_j}) - \kappa^{100}$ be a layer such that $f_{\mathsf{del}}(a') \ge \kappa^{-\kappa^3 r^r}$. Let $0 \le \tau < f_{\mathsf{del}}(a') \cdot \kappa^{a'-\kappa^{100}}$. Then, $\widehat{d_1 \dots d_j - \tau \dots d_{r-1}}$ $(2^{-\kappa/2})$-approximates $\widehat{d_1 \dots d_{r-1}}$ on del.*

Proof. Let z_a be a chunk of length κ^a of the string z. Let $\widehat{\mathrm{D}_1 \dots \mathrm{D}_{r-1}} := \widehat{d_1 \dots d_j - \tau \dots d_{r-1}}$. Let $q(\widehat{\mathrm{D}_1 \dots \mathrm{D}_{r-1}}) \in [\kappa^a]^r$ be the distribution of $\mathrm{Q}_1 \dots \mathrm{Q}_r$ in the original string when the queries $\widehat{\mathrm{D}_1 \dots \mathrm{D}_{r-1}}$ with a random shift are performed to the string z_a with random deletions according to del. Similarly, define $q(\widehat{d_1 \dots d_{r-1}})$. It suffices to show that the TV distance of these distributions is at most $(2^{-\kappa/2})$ (where getting out of bounds errors counts against the TV distribution on both ends).

Consider a randomly generated set of deletions from del. This assigns for each $\ell \le a$, a list of $\kappa^{a-\ell}$ values $\mathcal{E}_{\ell,i}$ for how many indices to delete in each layer using which the layered deletion algorithm is performed. Moreover, condition on the randomness of the queries' shift. Denote the induced indices in the original codeword when the queries $\widehat{d_1 \dots d_{r-1}}$ with the fixed random shift are performed by $q_1 \dots q_r$. The way we found these was by considering the string z, marking all the deleted symbols, and counting $\widehat{q_1}$ non-deleted symbols from the left, then $\widehat{d_1}$ more symbols, and so on.

Let I be the index of the first chunk of length $\kappa^{a'}$ that begins after q_j. Increment the value of $\mathcal{E}_{a',I}$ by τ. The probability that this becomes invalid (note that all valid options are equally likely since del is uniform) option for del is at most

$$\frac{\tau}{f_{\mathsf{del}}(a')\kappa^{a'}} \le \frac{f_{\mathsf{del}}(a') \cdot \kappa^{a'-\kappa^{100}}}{f_{\mathsf{del}}(a')\kappa^{a'}} \le \kappa^{\kappa^{100}}.$$

The above defines a mapping from an element of del to another element of del, except for a very small fraction of the time. In this new element of del, the queries $\widehat{\mathrm{Q}_1 \dots \mathrm{Q}_r}$ yield $q_1 \dots q_r$, because the transformation just causes an extra τ indices to get marked between q_j and q_{j+1} which exactly compensates for subtracting τ. To show that the new likelihood of getting $q_1 \dots q_r$ is at least as

large as before, we need to show this mapping is an injection. In particular, no instantiation of del is mapped to twice.

We will show the mapping is injective. If two instances of del and the query shift map to the same thing, they can only differ on the layer a' because only that changes in the mapping. If the index I is the same, then the map of adding τ is injective. If the index τ is different, then something before reaching I must have been different, to induce a different choice of I (which depends only on the choices of $\mathcal{E}$ before I), but then that index of $\mathcal{E}$ will be different in the image as well.

The two failure cases are where the mapping fails, because $\mathcal{E}_{\ell,i} + \tau$ is too large, or the initial choice of del and the query shift resulted in err rather than a valid output. The latter occurs with probability at most $\frac{\widehat{d_1}+\ldots+\widehat{d_{r-1}}}{\kappa^a} < \kappa^{-\kappa^2}$. This failure probability amounts to less than $\kappa^{-\kappa}$ in total.

Therefore, the TV distribution of the queries differs by at most $2\kappa^{-\kappa}$, since the one-sided TV distance differs by at most $\kappa^{-\kappa}$. □

The second lemma essentially says that replacing a group of query differences with one that approximates it, when the surrounding query differences are much larger in comparison, does not affect the distribution of outputs.

Lemma 4.11. *Consider a string $z \in \Sigma^M$, a layered deletion pattern del on strings of length κ^a for some $a < m$ with total corruption fraction at most $2^{-\kappa}$, and query differences $\widehat{d_1} \ldots \widehat{d_{r-1}}$. Moreover, every layer has corruption at most $\kappa^{-\kappa^2}$ and $\widehat{d_1} + \ldots + \widehat{d_{r-1}} < \kappa^{a-\kappa^2}$.*

Suppose there exists $1 \le j \le j' \le r-1$ and $a' \in \mathbb{N}$ such that $f_{\mathsf{del}}(a' + \kappa^{\kappa/1000}) > \kappa^{-\kappa^{\kappa/2000}}$

$$(\widehat{d_j} + \ldots + \widehat{d_{j'}}) \cdot \kappa^{\kappa^{\kappa/100}} < \kappa^{a'} < \min(\widehat{d_{j-1}}, \widehat{d_{j'+1}}) \cdot \kappa^{-\kappa^{\kappa/100}}.$$

Select $\widehat{\mathrm{D}_j} \ldots \widehat{\mathrm{D}_{j'}}$ such that it $((2k)^{j'-j+1}2^{-\kappa/100})$-approximates $\widehat{d_j} \ldots \widehat{d_{j'}}$ on the deletion pattern $\mathsf{del}_{a'}$. Then $\widehat{d_1} \ldots \widehat{\mathrm{D}_j} \ldots \widehat{\mathrm{D}_{j'}} \ldots \widehat{d_{r-1}}$ $((2k)^{r-1}2^{-\kappa/100})$-approximates $\widehat{d_1} \ldots \widehat{d_{j'-j+1}}$ on del.

Proof. We use a similar proof strategy as Lemma 4.10. Our goal will be to find an injective mapping from the queries $\widehat{d_1} \ldots \widehat{d_{r-1}}$ to $\widehat{\mathrm{D}_1} \ldots \widehat{\mathrm{D}_{r-1}}$ resulting in the same distribution of outputs that only fails with small probability over the randomness of choosing a query shift and an instance of del.

First, we remark that there is an injective map that except with failure probability $((2k)^{j'-j+1}2^{-\kappa/100})$ over uniformly randomly chosen options does the following. It maps a size $\kappa^{a'}$ chunk (of the $\kappa^{a-a'}$ options), query shift for that chunk, and $\mathsf{del}_{a'}$ pattern to another such that the output of $\widehat{d_j} \ldots \widehat{d_{j'}}$ is the same as $\widehat{\mathrm{D}_j} \ldots \widehat{\mathrm{D}_{j'}}$.

Secondly, we consider the distribution of a random instance of del_a and query shift. We look at where q_j falls, and our main goal will be to show that the chunk of size $\kappa^{a'}$ it is in, as well as the deletion pattern within that chunk, is essentially uniform (small TV distance from uniform). We'll start by conditioning on the deletions for chunks of size larger than $\kappa^{a'}$. The corruptions on chunks larger than $\kappa^{a'+2\kappa}$, when implemented, only affect up to $\kappa^{-\kappa}$-fraction of the chunks at all, and so ruling those out is only a negligible fraction of total chunks. The ones between $\kappa^{a'}$ and $\kappa^{a'+2\kappa}$ only cause at most $\kappa^{a'+2\kappa-\kappa^2} < \kappa^{a'-\kappa}$ deletions total, and so only affect a negligible $\kappa^{-\kappa}$ fraction of the chunk they fall into. Now, generate the small chunks' deletions randomly. The query shift dominates in which chunk q_j will fall into of the ones that weren't ruled out by the large deletions or by being in the first $\widehat{d_1} + \ldots + \widehat{d_j}$ eligible bits. In particular, the size remaining post-deletion of a chunk determines the chance of falling into it given the query shift, which differs by a factor of at most $1 - 2^{-\kappa} - \kappa^{-\kappa}$ between chunks. Thus, the distribution is TV distance at most $2^{-\kappa/2}$ from uniform.

Combining these two facts, we can apply the injective map to adjust the deletion pattern in the chunk that $\widehat{d_j}$ falls into so that queries $\widehat{d_j} \dots \widehat{d_{j'}}$ become $\widehat{\mathrm{D}_j} \dots \widehat{\mathrm{D}_{j'}}$ except with probability $((2k)^{j'-j+1}2^{-\kappa/100}) + 2^{-\kappa/2}$. In order to account for the query shift in the chunk of length $\kappa^{a'}$, one has to also adjust the amount of corruption in the length $\kappa^{a'+\kappa^{\kappa/1000}}$ length chunks directly before and after the chunk where $\widehat{d_j}$ fell. This adjustment fails with probability at most

$$\frac{\kappa^{a'}}{\kappa^{a'+\kappa^{\kappa/1000}} f_{\mathsf{del}}(a' + \kappa^{\kappa/1000})} < \kappa^{-\kappa/2}.$$

Because this map is reversible, it is injective. In all, the failure probability is at most $((2k)^{j'-j+1}2^{-\kappa/100}) + 2^{-\kappa/10}$, and so the TV distance of distribution of outputs for the two sets of queries is at most two times that, which in total results in a $((2k)^{j'-j+1}2^{-\kappa/100}) + 2^{-\kappa/10} < (2k)^{r-1}$ approximation. □

Lemma 4.12. *Consider a layered deletion pattern* del *on strings of length* κ^a *for some* $a < m$ *with total corruption fraction at most* $2^{-\kappa}$*, and query differences* $\widehat{d_1} \dots \widehat{d_{r-1}}$*. Choose* a' *such that every* $\widehat{d_i}$ *is either larger than* $\kappa^{a'} \cdot \kappa^{\kappa^{\kappa/100}}$ *or smaller than* $\kappa^{a'} \cdot \kappa^{-\kappa^{\kappa/100}}$*. Let us further assume that for all lists of query differences of length* s *at most* $r-2$ *(corresponding to* $r-1$ *queries) where the total width is at most* $\kappa^{a'}\kappa^{-\kappa^2}$*, there exists a list of query differences that* $(2k)^s(2^{-\kappa/2})$*-approximates it on at least* $(1-(2k)^s 2^{\kappa/2})$ *fraction of strings* $z \in Im(\mathsf{C})$ *for the deletion pattern* $\mathsf{del}_{a'}$*. Finally, assume that for at most*

$$\mathop{\mathbb{E}}_{j \in \left[\frac{M}{\kappa^{a'-\kappa^2}}\right]} \left[\left| \begin{array}{r} \mathsf{freq}^{(b)}(C_{j',\kappa^{\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2}}(z); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2}) \\ -\mathsf{freq}^{(b)}(C_{j,\kappa^{a'-\kappa^2}}(z); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}_{\tau|a'-\kappa^2}) \end{array} \right| \right] < 2^{-\kappa/2}$$

where $j' = \left\lfloor \frac{i}{\kappa^{\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2-a'-\kappa^2}} \right\rfloor$.

has no significant layers between $a'-\kappa^2$ *and* $\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2$*. Finally,* $f_{\mathsf{del}}(a') > \kappa^{\kappa^{\kappa/1000}}$.

Then, there is a list of queries $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}$ *that* $(2k)^{r-1}(2^{-\kappa/100})$*-approximates* $\widehat{d_1} \dots \widehat{d_{r-1}}$ *on* del*, which satisfies that all* $\widehat{\mathrm{D}_i} \le \kappa^{a'+\kappa^{\kappa/1000}}$.

Proof. We can partition the queries $\widehat{d_1} \dots \widehat{d_{r-1}}$ into groups $\widehat{d_j} \dots \widehat{d_{j'}}$ that comprise of the small query differences (and exclude the large differences). Firstly, we can apply Lemma 4.11 to successively replace the query differences $\widehat{d_j} \dots \widehat{d_{j'}}$ with a list $\widehat{\mathrm{D}_j} \dots \widehat{\mathrm{D}_{j'}}$ from the list of approximating queries that has no significant layers in the desired range, such that $(2k)^{j'-j+1}2^{-\kappa/100}$-approximates it. We can do this by choosing a' in the lemma to be $a'+\kappa^{\kappa/1000}$, because the adjacent query differences are all larger by at least a factor of $\kappa^{\kappa^{\kappa/100}}$, and the layer a' has sufficient corruption. Each iteration of this is a $(2k)^{j'-j+1}2^{-\kappa/100}$-approximation, so by Lemma 4.8, after performing all the replacements, we are left with a $(2k)^{r-1}2^{-\kappa/100}$-approximation at worst.

Now, the queries in the groups $\widehat{\mathrm{D}_j} \dots \widehat{\mathrm{D}_{j'}}$ has essentially the same distribution of outputs in the layers $a'-\kappa^2$ and $\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2$, and in fact in all layers between. At a high level, we want to show that replacing the remaining differences (large differences) with $\kappa^{a'+\kappa}$ does not change the distribution of outputs by much because there are no significant layers for any of the lists of short queries between layers $a'-\kappa^2$ and $\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2$, and so all query differences at approximately that scale function similarly.

Let the groups of short queries be $\widehat{d_{j_1}} \dots \widehat{\mathrm{D}_{j_1'}}, \widehat{\mathrm{D}_{j_2}} \dots \widehat{\mathrm{D}_{j_2'}}, \dots$, such that between each pair, there is a single $\widehat{d_{p_1}}$ that is large. (If there are multiple large differences in a row, the short query group between them is the empty list.)

We will show that we can replace any $\widehat{d_{p_i}}$ with $\kappa^{a'+\kappa^{\kappa/1000}}$ and adjust $\widehat{d_{p_{i+1}}}$ by $\widehat{d_{p_i}} - \kappa^{a'+\kappa^{\kappa/1000}}$ without changing the output distribution. In the case where $\widehat{d_{p_i}}$ is the rightmost difference, no further value $\widehat{d_{p_{i+1}}}$ need be adjusted We remark that this does not result in the same distribution of output queries induced in the original codeword, just of outputs in Σ^r. Repeating this process for each $\widehat{d_{p_i}}$ from left to right recovers the lemma statement.

By the same argument as in the proof of Lemma 4.11, the distribution of which $\kappa^{a'}$ block that $\widehat{\mathrm{D}_{j_i}} \dots \widehat{\mathrm{D}_{j_i'}}$ falls into and the corruption pattern within the block and the shift is at most $2^{-\kappa/2}$ TV distance from uniform. Then, when we reduce the value of $\widehat{d_{p_i}}$, there is a change in which block this query falls in. However, the distribution of where the list of queries falls now is still approximately uniform, since the argument would've also held using $\kappa^{a'+\kappa^{\kappa/1000}}$ for $\widehat{d_{p_i}}$. Then, the distributions of outputs for the queries induced by the differences $\widehat{\mathrm{D}_{j_i}} \dots \widehat{\mathrm{D}_{j_i'}}$ is the same in both cases. Moreover, there is a bijection between the shift and instantiation of $\mathsf{del}_{a'}$ in the block that the list of queries falls into in each case, because there are no significant layers. Since there are no significant layers, the output distribution in each length $\kappa^{a'}$ block is the same, except with small probability at most $2^{\kappa/2}$. By increasing the value of $\widehat{d_{p_{i+1}}}$ by exactly how much $\widehat{d_{p_i}}$ was decreased, the distribution of the other outputs does not change. However, the conditional distribution may change, based on the shift induced by $\widehat{\mathrm{D}_{j_i'}}$, but because the outputs could be approximately bijected without adjusting the shift of $\widehat{\mathrm{D}_{j_i'}}$ by more than $\kappa^{a'}$, this can be compensated for by adjusting the deletion in the preceding and following $\kappa^{a'+\kappa^{\kappa/2000}}$-sized block. Formally, like in the previous arguments, the map where these adjustments are made forms an injection that succeeds with high probability.

In all, this final list of queries, formed by replacing all the large differences by the value $\kappa^{a'+\kappa^{\kappa/1000}}$ results in a $(2k)^{r-1}(2^{-\kappa/100})$-approximation. □

4.2 Deletion Pattern and a Representative Set of Queries

In this section, we'll define the layered deletion pattern del used by the adversary in terms of the choice of code C.

The key idea of the proof of the main theorem (that a constant number of queries don't suffice for recovering every index) is that, given the deletion pattern del, there is a constant-sized representative set of lists of queries $\widehat{\mathcal{F}}$ such that any arbitrary query list can be simulated by an element of $\widehat{\mathcal{F}}$. That is, the output of the arbitrary list of queries and the chosen list from $\widehat{\mathcal{F}}$ have approximately the same distribution for most codewords in the code upto a random shift of the queries. More granularly, we'll let $\widehat{\mathcal{F}}_r$ be the subset of $\widehat{\mathcal{F}}$ consisting of successive differences for size r query lists.

In the following definition, we'll build up del and $\widehat{\mathcal{F}}$ simultaneously layer by layer.

The Deletion Pattern del and Representative Family of Query Differences $\widehat{\mathcal{F}}$

We define del by assigning for all $a \in [m]$ a value to $f_{\mathsf{del}}(a)$. Simultaneously, we'll build $\widehat{\mathcal{F}}$.

Preliminary values for layers $a \le 2\kappa^{\kappa^2}$: For $a \le \kappa^3$ let $f_{\mathsf{del}}(a) = 0$ and for $\kappa^3 < a < 2\kappa^{\kappa}$, let $f_{\mathsf{del}}(a) := \kappa^{-\kappa^2}$. Correspondingly, add the following elements to $\widehat{\mathcal{F}}_r$. Define the set S of important

differences as follows. Add every integer from $0 \dots \kappa^3$. Next, for $\kappa^3 < a < 2\kappa^\kappa$ and $0 < i \le \kappa^{2\kappa^2}$, add $\kappa^a \cdot \left(1 + \frac{i}{\kappa^{2\kappa^2}}\right)$. To each $\widehat{\mathcal{F}}_r$, add every $(r-1)$-sized list where each $\widehat{d_i}$ is an element of S. Also, include the empty list in $\widehat{\mathcal{F}}_1$.

Assigning layers $a \in [2\kappa^{\kappa^2}, m - \kappa^\kappa]$ (roughly) in sequence: We'll proceed one layer at a time starting from $a = \kappa^2$ to add more corruption to del. Which layers are corrupted next will depend on the current $\widehat{\mathcal{F}}$ and del. For each value of a ranging from κ^2 to m, do the following:

1. For each list of queries in $\widehat{\mathrm{Q}_1} \dots \widehat{\mathrm{Q}_r} \in \widehat{\mathcal{F}}$ with $\widehat{\mathrm{D}_1} + \dots + \widehat{\mathrm{D}_{r-1}} < \kappa^{a-\kappa^\kappa/2}$ and each $b \in \Sigma^r$, decide whether a is a significant layer for any $b \in \Sigma^r$ for the deletion process del (only looking at what it is so far) for the threshold $2^{-\kappa}$.
2. If not, then move on to the next a. Otherwise, let r be the smallest for which there is a query list higher than the threshold. Assign $f_{\mathsf{del}}(a') := \kappa^{-\kappa^3 r^r}$ for all $a - \kappa^\kappa \le a' < a + \kappa^\kappa$ (unless the already-assigned value was higher, in which case leave it).
3. For all $r' > r$ (note the strict inequality), add the following query difference lists to $\widehat{\mathcal{F}}_r$. Define the set S_a of important differences as follows: for $a - \kappa^\kappa \le a' < a + \kappa^\kappa$ and $0 < i \le \kappa^{2\kappa^2 r^r}$, add in $\kappa^a \cdot \left(1 + \frac{i}{\kappa^{2\kappa^2 r^r}}\right)$. Then, choose an arbitrary difference list $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_t} \in \widehat{\mathcal{F}}$ with $t < r - 1$ (for example, the singleton query list $\widehat{q_1} = 1$) and set $\widehat{d_1} \dots \widehat{d_t}$ to those values. Then, set $\widehat{d_{t+1}}$ to an element in S_a, and set $\widehat{q_{t+2}} \dots \widehat{q_{t'}}$ as an arbitrary element of $\widehat{\mathcal{F}}$ as before shifted such that the first query is $\widehat{q_{t+1}}$). Then set the next value $\widehat{d_{t'+1}}$ to an element in S_a and repeat.

Assigning layers $a > m - 2\kappa^\kappa$: Assign $f_{\mathsf{del}}(a) := \kappa^{-\kappa^3}$. (This part will be completed after the previous one, even though there is a slight overlap in layer.) Assign $f_{\mathsf{del}}(m) := \kappa^{-5}$.

Now, we'll prove a couple facts about del and $\widehat{\mathcal{F}}$.

Lemma 4.13. *For each r, the size of $\widehat{\mathcal{F}}_r$ is at most $\kappa^{4\kappa^2 r^r}$.*

Proof. We'll show the claim by induction on r. For $r = 1$, the only query we have added is $q_1 = 1$ so there is only one element.

Now we'll show the claim for r if it's true for all $r' < r$.

The first elements added to $\widehat{\mathcal{F}}_r$ were determined by a set S defined as every integer from $0 \dots \kappa^3$ and for $\kappa^3 < a < 2\kappa^\kappa$ and $0 < i \le \kappa^{2\kappa^2}$, adding an element. The size of this set is less than $\kappa^{3\kappa^2}$. Then, each of the $r-1$ differences between adjacent queries may be chosen from this set, for a total of $\left(\kappa^{3\kappa^2}\right)^r$ possible lists of queries. Note for clarity that this part did not involve the inductive hypothesis.

In the second way of adding elements to $\widehat{\mathcal{F}}_r$, we first find a value of a such that the layer is significant for one of the queries in $\widehat{\mathcal{F}}_{r'}$ with $r' < r$. By Lemma 4.9, for each query list, there are at most $\kappa^{2\kappa}$ values of a. In each, when we add new queries of length r, we first partition r into the sum of $r_1 \dots r_i$ where the sum is r and each $1 \le r_\iota < r$. Letting $N_{r'}$ denote the size of $\widehat{\mathcal{F}}_{r'}$, this quantity is at most

$$\prod_{\iota=1}^{i} |S_a| \cdot N_{r_\iota} \le \left(\kappa^{4\kappa^2 (r-1)^{r-1}}\right)^r \cdot \left(\kappa^{3\kappa^2}\right)^r \le \kappa^{\kappa^2 r^r}.$$

The number of options for $r_1 \dots r_i$ and the number of significant layers $\kappa^{2\kappa}$ increases this by at most a factor of κ^{κ^2}. □

Lemma 4.14. *The total maximum amount of corruption in* del*, aside from layer* m*, is less than* $2^{-\kappa}$*.*

Proof. Between layers 0 to $2\kappa^\kappa$, the amount of corruption in each layer is at most $\kappa^{-\kappa^2}$, so the total corruption is bounded by $2\kappa^\kappa \cdot \kappa^{-\kappa^2} < 2^{-\kappa-2}$.

Afterwards, each query list in $\widehat{\mathcal{F}}_r$ contributes at most $\kappa^{-\kappa^3 r^r}$ corruption for each of the $\kappa^{4\kappa^2 r^r}$ significant layers, and there are in total $\kappa^{\kappa^2 r^r}$ queries. Multiplying these quantities, the total corruption is at most $\kappa^{-\kappa^3 r^r/2}$. Adding over $r = 1$ to k, the total is less than $2^{-\kappa-1}$. The third type of deletion contributes at most $2^{-\kappa-2}$ corruption, and thus the total corruption is at most $2^{-\kappa}$. □

4.3 Main Lemma

Let $\widehat{\mathcal{F}}$ be the family of representative queries and del the layered deletion pattern.

Lemma 4.15. *For any query difference list* $\widehat{d_1} \dots \widehat{d_{r-1}}$ *for* $r \le k+1$*, define* $\ell := \lfloor \log_\kappa(\widehat{d_1} + \dots + \widehat{d_{r-1}}) \cdot \kappa^2 \rfloor$*. If* $\ell < m - 1.5\kappa^\kappa$*, there exists a member of* $\widehat{\mathcal{F}}_r$ *such that* $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}$ $2^{-\kappa/1000+r}$*-approximates* $\widehat{d_1} \dots \widehat{d_{r-1}}$ *on the deletion pattern* del_ℓ *for at least* $1 - \kappa^{-100}$ *fraction of* $z \in Im(\mathsf{C})$*.*

Proof. To show this claim, we'll induct on r.

For the base case of $r = 1$, the only possible 0 element list is the empty list, which is in $\widehat{\mathcal{F}}_1$. Any single query 0-approximates itself on all strings of length M.

We will now prove the statement for r assuming it holds for all $r' < r$. Consider any query difference list $\widehat{d_1} \dots \widehat{d_{r-1}}$. Throughout the proof, let's denote $c := \lfloor \log_\kappa(\widehat{d_1} + \dots + \widehat{d_{r-1}}) \rfloor$. We separate the proof into three cases.

Case 1: $c < 2\kappa^{\kappa^2}$**.** Consider the list of differences $\widehat{d_1} \dots \widehat{d_{r-1}}$ each of which is less than 2^c. In the first list of lists we added to $\widehat{\mathcal{F}}$, we constructed a set S of candidate differences. By the method by which the set was constructed, for every $\widehat{d_i} > \kappa^3$, there is $s \in S$ such that $\tau := \widehat{d_i} - s < \kappa^{-2\kappa^2}$. Moreover, the layer $a := \log(\widehat{d_i}) - \kappa^{100}$ is larger than 0, and has $\kappa^{-\kappa^2} > \kappa^{\kappa^3 r^r}$ corruption, so we can apply Lemma 4.10 to replace $\widehat{d_i}$ with s in the query. Doing this for each $\widehat{d_i} > \kappa^3$, iteratively creates queries that are $2^{-\kappa/2}$ approximations. By Lemma 4.8, the final query upon all the replacements is a $2^{-\kappa/2}k$ approximation, and since all the differences are now in S, the list is in $\widehat{\mathcal{F}}$ by construction.

Case 2: There exists $\widehat{\mathrm{D}'_1} \dots \widehat{\mathrm{D}'_{r'}} \in \widehat{\mathcal{F}}_{r'}$ **for some** $r' < r$ **such that** a **is a significant layer for the threshold** $2^{-\kappa}$ **for some** $c - \frac{\kappa^\kappa}{10} < a < c + \frac{\kappa^\kappa}{10}$**.**

The layer a was corrupted with corruption at least $\kappa^{-\kappa^3 r^r}$. To show this, it suffices that a was significant layer for $\widehat{\mathrm{D}'_1} \dots \widehat{\mathrm{D}'_{r'}}$ when we reached layer a in the definition process, not only for the final value of del. a is a significant layer for $\widehat{\mathrm{D}'_1} \dots \widehat{\mathrm{D}'_{r'}}$ if a is within $10\kappa^\kappa$ queries above $a' := \lfloor \mathrm{D}'_1 + \dots + \mathrm{D}'_{r'} \rfloor$. If not, then the first a' layers have had their full corruptions finalized by the time layer a had been reached, and whether something is a significant layer for queries $\widehat{\mathrm{D}'_1} \dots \widehat{\mathrm{D}'_{r'}}$ depends only on the first a' layers of corruption.

Now, we'll find the list $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_r} \in \widehat{\mathcal{F}}_r$ that approximates $\widehat{d_1} \dots \widehat{d_{r-1}}$.

Select a threshold layer $a - \frac{\kappa^\kappa}{10} < a' < a$ such that no value of $\log_\kappa \widehat{d_i}$ falls between $a' - \kappa^{\kappa/20}$ and $a' + \kappa^{\kappa/20}$. This exists because there are only $r \le k$ differences and $\frac{\kappa^\kappa}{10}$ layers.

When describing the corruption process for a as a significant layer, we constructed the set S_a of differences. The set was constructed so that for every integer between $\kappa^{a-\kappa^\kappa}$ and $\kappa^{a+\kappa^\kappa}$, there is $s \in S_a$ that is at most a factor of $\kappa^{-2\kappa^2 r^r}$ smaller. This requires that $\kappa^{a-\kappa^\kappa} > \kappa^{\kappa^2 r^r}$ due to approximation issues. This holds as long as $a > \kappa^\kappa + \kappa^2 r^r$, which is true since $c > 2\kappa^{\kappa^2}$.

Partition the list $\widehat{d_1} \dots \widehat{d_{r-1}}$ into groups of consecutive elements as follows: starting from the first $\widehat{d_i}$, let a group be all the following consecutive elements satisfying $\widehat{d_i} < \kappa^{a'}$. If there are none, just let it be the next element that is larger. As such, groups of elements are either a single large element or a consecutive list of small elements.

From left to right, if the next group of query differences is a single element larger than $\kappa^{a'}$, then by Lemma 4.10, for every $z \in \mathrm{Im}(\mathsf{C})$, there exists a way to replace it with an element of S_a such that the new query approximates the old one on all strings of length κ^m.

If the next group of query differences (length r') is a group of elements all smaller than $\kappa^{a'-\kappa^{\kappa/20}}$, then the conditions of Lemma 4.11 are satisfied, so we can replace the group of query differences with one that approximates it in $\widehat{\mathcal{F}}_{r'}$ by the inductive hypothesis. On all $z \in \mathrm{Im}(\mathsf{C})$ where the approximation held, which was $(1-(2k)^{r-1}2^{\kappa/2})$ fraction at least, the approximation holds for the new list of queries as well by Lemma 4.11.

In all, this new list of query differences has been constructed using elements of $\widehat{\mathcal{F}}_{r'}$ and with differences in S_a. As such, this list of differences is in $\widehat{\mathcal{F}}_r$ by construction.

Since each successive change to the query provided a $2^{-\kappa/100}k$-approximation of the previous step, this means that $\widehat{\mathrm{D}_1} \dots \widehat{\mathrm{D}_{r-1}}$ $2^{-\kappa/100}k$-approximates the query $\widehat{d_1} \dots \widehat{d_{r-1}}$ on $1-(2k)^r/2 \cdot 2^{\kappa/2}$ of the values of $z \in \mathrm{Im}(\mathsf{C})$ by Lemma 4.8 and a union bound on the deletion pattern .

Case 3: For all $\widehat{\mathrm{Q}'_1} \dots \widehat{\mathrm{Q}'_{r'}} \in \widehat{\mathcal{F}}_{r'}$ for some $r' < r$, it holds that a is not a significant layer for the threshold $2^{-\kappa}$ for any $c - \frac{\kappa^\kappa}{10} < a < c + \frac{\kappa^\kappa}{10}$.

Define the layer a to be the largest significant layer for some $\widehat{\mathrm{Q}'_1} \dots \widehat{\mathrm{Q}'_{r'}} \in \widehat{\mathcal{F}}_{r'}$ for some $r' < r$ that is smaller than c.

Similar to the previous case, select a threshold layer $a + \frac{\kappa^\kappa}{50} < a' < a + \frac{\kappa^\kappa}{20}$ such that no value of $\log_\kappa \widehat{d_i}$ falls between $a' - \kappa^{\kappa/80}$ and a'. Then, for at least $2^{-\kappa/2}$ fraction of values of $z \in \mathrm{Im}(C)$, the following holds by Markov's inequality and the definition of a significant layer.

$$\mathop{\mathbb{E}}_{j \in \left[\frac{M}{\kappa^{a'-\kappa^2}}\right]} \left[\left| \begin{array}{r} \mathsf{freq}^{(b)}(C_{j', \kappa^{\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2}}(z); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}_{\tau|\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2}) \\ -\mathsf{freq}^{(b)}(C_{j, \kappa^{a'-\kappa^2}}(z); \widehat{d_1} \dots \widehat{d_{r-1}}; \mathsf{del}_{\tau|a'-\kappa^2}) \end{array} \right| \right] < 2^{-\kappa/2}$$

where $j' = \left\lfloor \frac{i}{\kappa^{\widehat{d_1}+\dots+\widehat{d_{r-1}}+\kappa^2-a'-\kappa^2}} \right\rfloor$.

Now, we can apply Lemma 4.12, and we can replace all the $\widehat{d_i}$ that are large with precisely the value $\kappa^{a'}$. This new query $1-(2k)^r/2 \cdot 2^{\kappa/2}$ approximates the old one on $1-(2k)^r/2 \cdot 2^{\kappa/2}$ values of $z \in \mathrm{Im}(\mathsf{C})$, and additionally now falls into Case 2. Therefore, there is a query that approximates this query, and ultimately $1-(2k)^r \cdot 2^{\kappa/2}$ approximates the original query on $1-(2k)^r \cdot 2^{\kappa/2}$ values of $z \in \mathrm{Im}(\mathsf{C})$ on the deletion pattern del_a. □

Definition 4.16 ($\widehat{\mathcal{G}}$ and $\widehat{\mathcal{Q}}$)**.** Let $\widehat{\mathcal{G}} \subset [M]$ be the differences that appear in some list in $\widehat{\mathcal{F}}$. This set $\widehat{\mathcal{G}}$ is constant-sized. Moreover, add every multiple of $\kappa^{m-1.75\kappa^\kappa}$ that is at most M to the set $\widehat{\mathcal{G}}$.

Let $\widehat{\mathcal{Q}}$ be constructed as follows. For every tuple $g_1, \dots g_k \in \widehat{\mathcal{G}}$, let $\widehat{\mathrm{Q}_1} := g_1, \widehat{\mathrm{Q}_2} := g_1 + g_2 \dots$, and add this list of queries to $\widehat{\mathcal{Q}}$. This set $\widehat{\mathcal{Q}}$ is constant-sized.

The following lemma says that querying $\widehat{q_1} \dots \widehat{q_k}$ will produce essentially the same output as some $\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k} \in \widehat{\mathcal{Q}}$.

Lemma 4.17. *For any (ordered) query list $\widehat{q_1} \dots \widehat{q_k} \in [M]^k$ and for at least $1 - \kappa^{-4}$ fraction of $z \in Im(\mathsf{C})$, there exists a list of queries $\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k} \in \widehat{\mathcal{Q}}$ such that $\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k}$ approximates $\widehat{q_1} \dots \widehat{q_k}$ for the deletion pattern del, meaning the following. For any $b \in \Sigma^k$,*

$$\left| \mathbb{P}\left[z(\mathcal{D}[\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k}]) = b \right] - \mathbb{P}\left[z(\mathcal{D}[\widehat{q_1} \dots \widehat{q_k}]) = b \right] \right| < \kappa^{-4}.$$

Proof. Given a list of queries $\widehat{q_1} \dots \widehat{q_k} \in [M]^k$, separate the differences $\widehat{d_1} \dots \widehat{d_{k-1}}$ as follows. Find a layer $a \in [m - 1.7\kappa^\kappa, m - 1.6\kappa^\kappa]$ such that no value of $\log_\kappa \widehat{d_i}$ falls between $a - \kappa^{\kappa/20}$ and $a + \kappa^{\kappa/20}$.

Using this layer, we can designate the small and large differences. At a high level, we'll show that replacing sequences of small differences with ones that approximate them on some layer $c < a'$ does not change the output distribution by much, and adjusting large differences by $\kappa^{m-1.75\kappa^\kappa}$. The main difference between this situation and Case 2 of Lemma 4.15 is that we won't desire to shift the entire query by uniformly random amount in $[\kappa^m]$; only by an amount in $[0, \kappa^{m-5}$, since that is the corruption of layer m.

By Lemma 4.15, for at least $1 - \kappa^{-100}$ fraction of the $z \in \mathrm{Im}(\mathsf{C})$, a given short sequence of differences can be $2^{-\kappa/2000}$-approximated by one in $\widehat{\mathcal{G}}$. In particular, this means that for at least $1 - \kappa^{-90}$ of the $z \in \mathrm{Im}(\mathsf{C})$, a $2^{\kappa/5000}$-approximation holds on all of the κ^{m-10}-sized chunks.

The shift of the queries dictated by the deletion at layer m is κ^{m-5}. Therefore, when a random shift (deletion at layer m is chosen), except for probability of failure at most κ^{-5}, this deletion amount can be shifted by κ^{m-10}. In other words, except with probability at most κ^{-5}, the shift will be uniform within this chunk.

This allows us to apply Lemma 4.11 on the layer a to replace sequences of short queries with a sequence in $\widehat{\mathcal{G}}$, and since the output given a uniform shift is a $2^{-\kappa/1000}$-approximation, it is in this situation as well, since except with probability $1-\kappa^{-5}$ the query could be simulated with a uniform shift. Next, we can apply Lemma 4.10 to adjust the large queries slightly such that differences of successive queries are multiples of $\kappa^{m-1.75\kappa^\kappa}$. In total, the probability of this new query output not appropriately simulating the original $\widehat{q_1} \dots \widehat{q_k}$ is at most κ^{-4}, proving the lemma. □

4.4 Concluding Theorem 4.1 and Corollary 4.2.

We begin with a lemma that shows that $z \in \mathrm{Im}(\mathsf{C})$ can be compressed down to a constant $\overline{K}(\varepsilon, |\Sigma|, k)$ length string such that outputs of any list of k queries can be predicted by this compressed string.

Lemma 4.18. *There exists a constant $\overline{K} := \overline{K}(\varepsilon, |\Sigma|, k)$ and a compression function $f : Im(\mathsf{C}) \to \Sigma^{\overline{K}}$ such that the following holds. For any k queries $\widehat{q_1} \dots \widehat{q_k}$, there is a randomized function $\mathsf{query} : \Sigma^{\overline{K}} \to \Sigma^k$ such that for at least $1 - \kappa^{-1}$ fraction of $z \in Im(\mathsf{C})$, the distribution of $\mathsf{query}(f(z))$ has TV distance at most $1 - \kappa^{-1}$ from the outputs of the queries $\widehat{q_1} \dots \widehat{q_k}$ after applying the deletion pattern del.*

Proof. This is essentially a restatement of Lemma 4.17. In particular, define f as follows. Apply the deletion pattern del, and for every query $\widehat{\mathsf{Q}}$ in any list in $\widehat{\mathcal{Q}}$, let the output of $\widehat{\mathsf{Q}}$ be included in the compression. Then, query can be performed by taking the queries $\widehat{\mathsf{Q}_1} \dots \widehat{\mathsf{Q}_k} \in \widehat{\mathcal{Q}}$ that approximate $\widehat{q_1} \dots \widehat{q_k}$, and looking at $f(z)$ to figure those out. The output of this has TV distance at most $1-\kappa^{-1}$ for at least $1 - \kappa^{-1}$ fraction of values of z. □

Lemma 4.19. *If the output of all lists of queries can be deduced up to TV distance $1-\kappa^{-1}$ on κ^{-1}-fraction of strings from some randomized compression function $f : Im(\mathsf{C}) \to \Sigma^{\overline{K}}$, there exists a constant K so that for $n > K$, there exists $i \in [n]$ such that for uniformly random x, the index $x[i]$ cannot be guessed correctly from any query list with probability more than $\frac{1}{2}+\varepsilon^2$.*

Proof. Assume for the sake of contradiction that the queries are capable of recovering every index i with probability $\frac{1}{2}+\varepsilon^2$. Then, on $1-\kappa^{-1}$-fraction of inputs, the compressed version of it $f(\mathsf{C}(x))$ is capable of recovering i with probability $\frac{1}{2}+\varepsilon+\kappa^{-1}$.

Perform the following process, which we'll call Y: take a random string $x \in \Sigma^n$, take the compressed string $f(\mathsf{C}(x))$ and use it to recover each of $x[1]\dots x[n]$. We know that $H(Y) \le \overline{K}$. Each index i has a $\frac{1}{2}+\varepsilon^2-2\kappa^{-1} > \frac{1}{2}+\varepsilon^3$ chance (but not independent) of yielding the correct value of $x[i]$. In total, this means the entropy of each bit of the string x conditional on the guesses provided by the string is at most $1-\varepsilon^6$. By the subadditivity of entropy, the entropy of this is at most $(1-\varepsilon^6)n$, so $H(Y) \le (1-\varepsilon^6)n$. Also, the entropy of the string x, which we'll call $H(X)$, is n.

We know that

$$\begin{aligned} &H(X) \le H(X|Y) + H(Y) \\ \implies\ &n \le \overline{K} + (1-\varepsilon^6)n \end{aligned}$$

which is a contradiction for sufficiently large n. □

Combining Lemma 4.18 and Lemma 4.19 concludes the proof of Theorem 4.1.

Theorem 4.2 follows by Theorem 4 in [BCG+22], which states that the existence of LCC's in the adversarial insertion/deletion setting would imply LDC's in the adversarial insertion/deletion setting, which don't exist by Theorem 4.1.

Our result requires that deletion resilient LCC's would imply deletion resilient LDC's. Their proof carries over identically to this setting, and we briefly sketch why.

Proof Sketch. The main property of the corruption pattern they used was that no two codewords z, z' in the codespace of an LCC could have Hamming distance less than $\varepsilon' n$, where ε' is the allotted corruption for the adversary. This also holds in the deletion-only setting, because if two codewords z and z' were within Hamming distance $\varepsilon' n$, the adversary could delete all the indices on which they differ. The resulting messages would be the same, and any bit that differed between the two encodings cannot be recovered.

Then, the codespace has VC dimension $d = O_{\varepsilon,|\Sigma|}(n)$, by Lemma 6 in [BCG+22]. Restrict the code to a subset C' of size Σ^d with the same shattering set, and define the corresponding code $\Sigma^d \to C'$, as the extension of the values on the shattering set into C'. This new code is an LCC that follows a systematic encoding, which would also make it an LDC on messages of length $d = O_{\varepsilon,|\Sigma|}(n)$. This cannot exist by Theorem 4.1. □

5 Acknowledgements

Meghal Gupta is supported by an NSF Graduate Research Fellowship. The author would like to thank her advisor Dr. Venkatesan Guruswami for invaluable guidance, helpful discussions, and feedback on the manuscript. The author would also like to thank Omar Alrabiah and Yang Liu for helpful discussions and comments on the paper.

References

[AGKM23] Omar Alrabiah, Venkatesan Guruswami, Pravesh K Kothari, and Peter Manohar. A near-cubic lower bound for 3-query locally decodable codes from semirandom csp refutation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1438–1448, 2023. 3

[BB21] Alexander R. Block and Jeremiah Blocki. Private and resource-bounded locally decodable codes for insertions and deletions. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1841–1846, 2021. 2

[BBC+22] Alex Block, Jeremiah Blocki, Kuan Cheng, Elena Grigorescu, Xin Li, Yu Zheng, and Minshen Zhu. On relaxed locally decodable codes for hamming and insertion-deletion errors. *arXiv preprint arXiv:2209.08688*, 2022. 2

[BBG+20] Alexander R Block, Jeremiah Blocki, Elena Grigorescu, Shubhang Kulkarni, and Minshen Zhu. Locally decodable/correctable codes for insertions and deletions. *arXiv preprint arXiv:2010.11989*, 2020. 1, 2

[BCG+22] Jeremiah Blocki, Kuan Cheng, Elena Grigorescu, Xin Li, Yu Zheng, and Minshen Zhu. Exponential lower bounds for locally decodable and correctable codes for insertions and deletions. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 739–750. IEEE, 2022. 1, 2, 4, 5, 23

[BFLS91] László Babai, Lance Fortnow, Leonid A Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32, 1991. 3

[BGMO17] Mark Braverman, Ran Gelles, Jieming Mao, and Rafail Ostrovsky. Coding for interactive communication correcting insertions and deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, 2017. 1

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998. 3

[CLZ20] Kuan Cheng, Xin Li, and Yu Zheng. Locally decodable codes with randomized encoding. *arXiv preprint arXiv:2001.03692*, 2020. 2

[CS08] David J Coumou and Gaurav Sharma. Insertion, deletion codes with feature-based embedding: a new paradigm for watermark synchronization with applications to speech watermarking. *IEEE Transactions on Information Forensics and Security*, 3(2):153–165, 2008. 1

[Efr09] Klim Efremenko. 3-query locally decodable codes of subexponential length. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 39–44, 2009. 3

[GHL22] Venkatesan Guruswami, Xiaoyu He, and Ray Li. The zero-rate threshold for adversarial bit-deletions is less than 1/2. *IEEE Transactions on Information Theory*, 69(4):2218–2239, 2022. 2

[GKST02] Oded Goldreich, Howard Karloff, Leonard J Schulman, and Luca Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. In *Proceedings 17th IEEE Annual Conference on Computational Complexity*, pages 175–183. IEEE, 2002. 3

[GYM16] Ryan Gabrys, Eitan Yaakobi, and Olgica Milenkovic. Codes in the damerau distance for dna storage. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2644–2648. IEEE, 2016. 1

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. 1

[HS17] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: codes for insertions and deletions approaching the singleton bound. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 33–46, 2017. 2

[HS21] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings and codes for insertions and deletions—a survey. *IEEE Transactions on Information Theory*, 67(6):3190–3206, 2021. 3

[KDW03] Iordanis Kerenidis and Ronald De Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 106–115, 2003. 3

[KM23] Pravesh K Kothari and Peter Manohar. An exponential lower bound for linear 3-query locally correctable codes. *arXiv preprint arXiv:2311.00558*, 2023. 3

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 80–86, 2000. 3

[Lev66] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966. 1, 2

[LSWZY19] Andreas Lenz, Paul H Siegel, Antonia Wachter-Zeh, and Eitan Yaakobi. Coding over sets for dna storage. *IEEE Transactions on Information Theory*, 66(4):2331–2351, 2019. 1

[Mit09] Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. 2009. 3

[Mul54] David E Muller. Application of boolean algebra to switching circuit design and to error detection. *Transactions of the IRE professional group on electronic computers*, (3):6–12, 1954. 3

[OPC15] Rafail Ostrovsky and Anat Paskin-Cherniavsky. Locally decodable codes for edit distance. In *Information Theoretic Security: 8th International Conference, ICITS 2015, Lugano, Switzerland, May 2-5, 2015. Proceedings 8*, pages 236–249. Springer, 2015. 1, 2

[Ree54] Irving S Reed. A class of multiple-error-correcting codes and the decoding scheme. *IEEE Transactions on Information Theory*, 4(4):38–49, 1954. 3

[Sha48] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. 1

[SZ99] Leonard J Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE transactions on information theory*, 45(7):2552–2557, 1999. 2

[Tre04] Luca Trevisan. Some applications of coding theory in computational complexity. *arXiv preprint cs/0409044*, 2004. 3

[WDW05] Stephanie Wehner and Ronald De Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *International Colloquium on Automata, Languages, and Programming*, pages 1424–1436. Springer, 2005. 3

[Woo07] David Woodruff. New lower bounds for general locally decodable codes. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 14, 2007. 3

[Yek12] Sergey Yekhanin. Locally decodable codes. *Foundations and Trends® in Theoretical Computer Science*, 6(3):139–255, 2012. 1, 3

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il