Linear Insertion Deletion Codes in the High-Noise and High-Rate Regimes

Kuan Cheng
Department of Computer Science, Peking University, Beijing, China

Zhengzhong Jin
Massachusetts Institute of Technology, Cambridge, MA, USA

Xin Li
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

Zhide Wei
Department of Computer Science, Peking University, Beijing, China

Yu Zheng
Meta Platforms Inc

Abstract

This work continues the study of linear error correcting codes against adversarial insertion deletion errors (insdel errors). Previously, the work of Cheng, Guruswami, Haeupler, and Li [6] showed the existence of asymptotically good linear insdel codes that can correct arbitrarily close to 1 fraction of errors over some constant size alphabet, or achieve rate arbitrarily close to 1/2 even over the binary alphabet. As shown in [6], these bounds are also the best possible. However, known explicit constructions in [6], and subsequent improved constructions by Con, Shpilka, and Tamo [9] all fall short of meeting these bounds. Over any constant size alphabet, they can only achieve rate $< 1/8$ or correct $< 1/4$ fraction of errors; over the binary alphabet, they can only achieve rate $< 1/1216$ or correct $< 1/54$ fraction of errors. Apparently, previous techniques face inherent barriers to achieve rate better than 1/4 or correct more than 1/2 fraction of errors.

In this work we give new constructions of such codes that meet these bounds, namely, asymptotically good linear insdel codes that can correct arbitrarily close to 1 fraction of errors over some constant size alphabet, and binary asymptotically good linear insdel codes that can achieve rate arbitrarily close to 1/2. All our constructions are efficiently encodable and decodable. Our constructions are based on a novel approach of code concatenation, which embeds the index information implicitly into codewords. This significantly differs from previous techniques and may be of independent interest. Finally, we also prove the existence of linear concatenated insdel codes with parameters that match random linear codes, and propose a conjecture about linear insdel codes.

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases Error correcting code, Edit distance, Pseudorandomness, Derandomization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.41

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2303.17370

Funding *Kuan Cheng*: Supported by a start-up fund of Peking University.
Zhengzhong Jin: Supported in part by DARPA under Agreement No. HR00112020023 and by an NSF grant CNS-2154149. Most of the work was done while the author was a PhD student at Johns Hopkins University, and supported by NSF CAREER Award CCF-1845349.
Xin Li: Supported by NSF CAREER Award CCF-1845349 and NSF Award CCF-2127575.
Zhide Wei: Supported by a start-up fund of Peking University.
Yu Zheng: Most of the work was done while the author was a PhD student at Johns Hopkins University, and partially supported by NSF Award CCF-2127575.

1 Introduction

Error correcting codes are fundamental objects in computer science and information theory. Starting from the seminal work of Shannon and Hamming, the study of error correcting codes has led to a deep understanding of how to ensure reliable communications in various noisy channels. Furthermore, error correcting codes have found rich applications in other seemingly unrelated areas such as complexity theory, learning theory, pseudorandomness and many more. Traditionally, the errors studied are either erasures (where a transmitted symbol is replaced by a '?') or symbol modifications (where a transmitted symbol is replaced by a different symbol), and they can be either random or adversarial. Through decades of effort, we now have an almost complete understanding of codes for such errors, and constructions with efficient encoding and decoding algorithms that match or are close to various known bounds.

An important and more general type of errors, known as *synchronization errors*, however, is much less understood. These errors include insertion and deletions (so we also call them insdel errors for short), which can cause the positions of received symbols to shift. On the other hand, they occur frequently in real world applications, including disk access, integrated circuits, communication networks and so on. They are also closely related to applications in computational biology and DNA-based storage systems [3, 25]. Although the study of codes for such errors started around the same time as Shannon's works, progress has historically been slow due to the apparent difficulty of handling the loss of index information with such errors. For example, many basic questions, such as the capacity of the binary deletion channel with deletion probability p is still wide open, and the first explicit construction that has a constant rate and can correct a constant fraction of adversarial errors is not known until 1999 [21].

From now on, we will focus exclusively on adversarial insdel errors. Over the past several years, with the development of new techniques such as *synchronization strings* [17], there has been a wave of new constructions of codes for these errors [17, 16, 22, 8, 4, 15, 19, 14, 5, 7, 18, 12, 20, 24, 14]. Some of them achieve excellent parameters, e.g., codes that approach the singleton bound over a large constant alphabet [17], codes with almost optimal redundancy over the binary alphabet [8, 15], list-decodable codes over large alphabets that can correct more errors than the length of the codeword [18], and list-decodable codes over any alphabet of positive rate for the information-theoretically largest possible combination of insertions and deletions [12, 20, 24, 14]. However, none of the above constructions gives a *linear* code, and the existence of asymptotically good linear codes for insdel errors over a constant size alphabet is not known until the work of Cheng, Guruswami, Haeupler, and Li [6].

The motivation of studying linear codes comes from several aspects. First, they have compact representations using either generator matrices or parity check matrices, which directly give efficient encoding and testing algorithms with running time $O(n^2)$. Second, such codes have simple structures, so they are often easier to analyze and allow one to use powerful techniques from linear algebra. Finally, linear codes have had great success in codes for erasures and symbol modifications, achieving some of the most well known constructions with (near) optimal parameters. Thus, one could ask if the same is true for insdel codes.

As is standard in the literature of error correcting codes, the two most important parameters of a linear insdel code are δ, the fraction of insdel errors the code can correct; and R, the rate of the code, defined as the message length divided by the codeword length. In [6], the authors established several bounds regarding the tradeoff between these two parameters for linear insdel codes. First, they showed that any linear code correcting δ fraction of insdel errors must have rate at most $\frac{1}{2}(1-\delta)$, regardless of the alphabet size. This is known as the *half-singleton bound* and generalizes a previous result in [1], which shows that any linear code

that can correct even a single deletion must have a rate of at most 1/2. This bound shows a severe limitation of linear codes for insdel errors, as general codes can correct δ fraction of errors with R approaching $1-\delta$. Taking into consideration the alphabet size q, this bound can be improved to $\frac{1}{2}(1-\frac{q}{q-1}\delta)+o(1)$, which is known as the *half-Plotkin bound*. On the other hand, the authors also showed that over the field F_q, for any $\delta>0$ there exists a linear code family that can correct δ fraction of insdel errors, with rate $(1-\delta)/2-\mathsf{H}(\delta)/\log_2 q$, where H is the binary entropy function. In particular, this implies the existence of binary linear codes with rate $1/2-\varepsilon$ capable of correcting $\Omega(\varepsilon\log^{-1}\frac{1}{\varepsilon})$ fraction of insdel errors for any $\varepsilon>0$; and linear insdel codes over F_q of rate $\frac{1}{2}(1-\delta)-\varepsilon$ capable of correcting any δ-fraction of insdel errors, for a large enough $q=2^{\Theta(\varepsilon^{-1})}$, which approaches the half-singleton bound. Hence, the rate can approach 1/2 even over the binary alphabet, and the fraction of errors corrected can approach 1 over a constant size alphabet, both of which are the best possible.

Going further, [6] also constructed explicit asymptotically good linear insdel codes. However, the fraction of errors the code can correct and the rate of the code are both quite small. [6] did not specify these constants, but a rough estimate shows that the code has $\delta<1/400$ and $R<2^{-80}$. Thus a natural question left in their work is to improve these parameters.

Recently, a subsequent work by Con, Shpilka, and Tamo [9] made progress in this direction. For a field F_q with $q=\mathsf{poly}(1/\varepsilon)$, they constructed explicit linear insdel codes that can correct δ fraction of errors with rate $R=(1-4\delta)/8-\varepsilon$. For the field F_2 their explicit linear code can correct δ fraction of errors with rate $R=(1-54\delta)/1216$. Hence, for a constant size alphabet their construction can achieve $\delta<1/4$ with a positive R, or $R<1/8$ with a positive δ. For the binary alphabet, their construction can achieve $\delta<1/54$ with a positive R, or $R<1/1216$ with a positive δ. One caveat is that their codes over the binary alphabet can only decode efficiently from deletions (although they can also decode from insertions inefficiently), while their codes over the large alphabet can decode efficiently from both deletions and insertions. In another work by the same authors [10], they also showed the existence of Reed-Solomon codes over a field of size $n^{O(k)}$ that have message length k, codeword length n, and can correct $n-2k+1$ insdel errors. This achieves the half-singleton bound. They complemented the existential result by providing a deterministic construction over a field of size $n^{k^{O(k)}}$, which runs in polynomial time for $k=O(\log n/\log\log n)$. Nevertheless, in this paper we only focus on the case of a constant alphabet size.

In summary, all known explicit constructions over constant size alphabets fall short of getting rate close to 1/2, or getting the fraction of errors correctable close to 1. In fact, previous techniques seem to face inherent barriers to achieve rate better than 1/4 or correct more than 1/2 fraction of errors, which we will talk about in more details when we give an overview of our techniques.

1.1 Our Results

In this paper we further improve the fraction of errors δ and the rate R that can be achieved by linear insdel codes with efficient encoding and decoding algorithms. In the case of high noise, we give explicit constructions of insdel codes with positive rate that can correct δ fraction of errors with δ arbitrarily close to 1, over a constant size alphabet. In the case of high rate, we give explicit constructions of insdel codes that can achieve rate arbitrarily close to 1/2 and correct a positive constant fraction of errors, over the binary alphabet.[1]

[1] It's also easy to generalize our constructions to larger alphabet size, but for clarity we omit the details in this version.

Specifically, we have the following theorems.

▶ **Theorem 1** (High noise). *For any constant $\varepsilon > 0$ there exists an efficient construction of linear insdel codes over an alphabet of size* $\mathsf{poly}(1/\varepsilon)$*, with rate $\Omega(\varepsilon^2)$ that can correct $1-\varepsilon$ fraction of insdel errors (possibly inefficiently).*

With efficient decoding, the rate becomes slightly worse.

▶ **Theorem 2** (High noise). *For any constant $\varepsilon > 0$, there is a family of linear codes with rate $\Omega(\varepsilon^4)$ and alphabet size* $\mathsf{poly}(1/\varepsilon)$*, that can be encoded in polynomial time and decoded from up to $1-\varepsilon$ fraction of insdel errors in polynomial time.*

▶ **Theorem 3** (High rate). *For any constant $\varepsilon > 0$, there is a family of binary linear codes with rate $1/2-\varepsilon$, that can be encoded in polynomial time and decoded from $\Omega(\varepsilon^3 \log^{-1} \frac{1}{\varepsilon})$ fraction of insdel errors in polynomial time.*

Our constructions are based on code concatenation. We complement our explicit constructions by showing that there exist linear concatenated codes that match the parameters of random linear codes. These constructions can be considered in a sense "semi-explicit" since the outer code is explicit, and we only need to find explicit inner codes.

▶ **Theorem 4.** *For any field F_{q_0} and any constant $\delta > 0$, there exists a family of linear concatenated code over F_{q_0} where the outer code is a Reed-Solomon code, such that the code has rate $\frac{1}{2}(1-\delta) - \mathsf{H}(\delta)/\log q_0 - o(1)$ and can correct δ fraction of insdel errors, where $\mathsf{H}()$ is the binary entropy function.*

We emphasize that the inner codes here may be different for different positions. So if one wants to use brute force to search for a sequence of proper inner codes, then this may take time at least $2^{n \log^2 n}$ where n is the length of the outer codewords.

This theorem implies the following corollaries.

▶ **Corollary 5.** *For any constant $\delta > 0$, there exists a family of binary linear concatenated code where the outer code is a Reed-Solomon code, such that the code has rate $\frac{1}{2}(1-\delta)$ and can correct $\Omega(\delta \log^{-1} \frac{1}{\delta})$ fraction of insdel errors.*

▶ **Corollary 6.** *For any constants $\delta, \varepsilon > 0$ there exists a family of linear concatenated code over an alphabet of size $q = 2^{\Theta(\varepsilon^{-1})}$ where the outer code is a Reed-Solomon code, such that the code has rate $\frac{1}{2}(1-\delta) - \epsilon$ and can correct δ fraction of insdel errors.*

Finally, we study the question of whether binary linear insdel codes can achieve δ arbitrarily close to $1/2$ with a positive rate R. Notice that even for general binary codes, it is well known that the maximum fraction of deletions that any non-trivial binary code of size $\geqslant 3$ can correct is below $1/2$ since any 3 different n-bit binary strings must contain two strings with the same majority bit, and thus their longest common subsequence is at least $n/2$. For binary linear codes this can also be seen from the half-Plokin bound. A recent work by Guruswami, He, and Li [13] in fact already provided a negative answer to this question even for general binary codes. In particular, they showed that there exists an absolute constant $\alpha > 0$ such that any binary code $\mathbb{C} \subseteq \{0,1\}^n$ with $|\mathbb{C}| \geqslant 2^{\mathsf{polylog} n}$ must have two strings whose longest common subsequence has length at least $(1/2+\alpha)n$. Thus $\mathbb{C}$ cannot correct more than $1/2-\alpha$ fraction of insdel errors. Since linear codes are more restricted, one may expect that a stronger result can be proved for binary linear codes. Specifically, we have the following conjecture:

▶ **Conjecture 7.** *There exists an absolute constant $\alpha > 0$ such that any linear subspace $\mathbb{C} \subseteq \mathsf{F}_2^n$ with dimension $\geqslant 3$ must have two strings (vectors) whose longest common subsequence has length at least $(1/2 + \alpha)n$.*

However, we are not able to prove this conjecture. Instead, we can prove a weaker result.

▶ **Theorem 8.** *There exists an absolute constant $\alpha > 0$ such that any linear subspace $\mathbb{C} \subseteq \mathsf{F}_2^n$ with dimension $\geqslant 3$ must have two strings (vectors) whose longest common subsequence has length at least $(\frac{1}{2} + \frac{\alpha}{\log n})n$.*

1.2 Overview of the Techniques

There have been only two previous works on explicit constructions of asymptotically good linear insdel codes over fields of constant size, i.e., [6] and [9]. The apparent difficulty of constructing such codes comes from the following aspects: First, many of the previous constructions of (non-linear) insdel codes are based on adding index information to the codewords, either in the form of direct encoding of indices, or more sophisticated objects such as synchronization strings. Since all of these result in fixed strings, adding such information in any naive way will lead to non-linear codes. Indeed, both [6] and [9] have to find alternative ways to "embed" synchronization strings into a linear code. Specifically, [6] uses what is called a *synchronization sequence*, which is a sequence of 0's added in between each pair of adjacent symbols in a codeword. This preserves the linearity if the original code is linear. [9], on the other hand, embeds the synchronization string by combining a codeword symbol x and a synchronization string symbol a into a pair $(x, a \cdot x)$, where $\cdot$ is the multiplication over the corresponding field F_q. This also preserves the linearity over F_q, but now the symbols from the synchronization strings are mixed with symbols from the codeword, and it is not easy to tell them apart. Note that for decoding, one needs to first use the synchronization string to recover the positions of the codeword symbols. To solve this problem, [9] also needs to add buffers of 0's between adjacent pairs, where the length of a buffer is at least as long as the pair $(x, a \cdot x)$.

It can be seen that the added 0's in the above two approaches form an inherent barrier to achieving high rate or high fraction of correctable errors. In [6], a constant number of 0's are added in between each pair of adjacent symbols in a codeword, which already decreases the rate and the possible decoding radius to a small constant. In [9], the operation of converting a codeword symbol x and a synchronization string symbol a into a pair $(x, a \cdot x)$ already decreases the rate of the code to below $1/2$, while adding 0's as buffers decreases the rate even more to below $1/4$. Similarly, add 0's as buffers also decreases the possible decoding radius to below $1/2$. For binary codes, [9] needs to use another layer of code concatenation, which further decreases the rate and decoding radius.

The key idea in all our constructions is to eliminate the use of 0's as buffers or synchronization sequences. Instead, we embed synchronization information directly into the codewords. To achieve this, we also use code concatenation, where for the outer code we choose a suitable Reed-Solomon code. On the other hand, the key difference between our constructions and standard concatenated codes is that we choose a *different* inner code for *every position* of the outer code. This way, we can make sure that the inner codewords corresponding to outer codeword symbols at different positions are far enough from each other, and thus we can roughly tell them apart by just looking at the received codeword. By using linear inner codes for all positions, this preserves the linearity of the code, and at the same time eliminates the use of 0's. On a high level, this is why our constructions can achieve either high rate (arbitrarily close to $1/2$) or high fraction of correctable errors (arbitrarily close to 1). We now discuss our techniques in more details for the two cases.

Constructions for high error. Note that to correct $1-\varepsilon$ fraction of insdel errors, a linear code must have alphbet size at least $1/\varepsilon$ by the half-Plotkin bound. Here we use an alphabet of size $\mathsf{poly}(1/\varepsilon)$. With an appropriately chosen parameter $\gamma = \Omega(\varepsilon)$, after picking an outer Reed-Solomon code with codeword length n, rate γ and relative distance $1-\gamma$, our strategy is to design n different inner codes $\mathbb{C}_{\mathsf{in}}^1, \cdots, \mathbb{C}_{\mathsf{in}}^n$. The goal is to ensure that codewords in different inner codes have large edit distance, or equivalently, the length of their longest common subsequence (LCS for short) is at most $\gamma n'$ where $n' = O(\log n)$ is the block length of the inner code. However, since all these codes are linear, 0 is a codeword of each inner code, and two 0's (even from different inner codes) are guaranteed to have 0 edit distance. We design the inner codes to ensure this is the only bad case.

More specifically, we ensure that for any two inner codewords x, y, unless they are both 0 or they correspond to the same message in one inner code $\mathbb{C}_{\mathsf{in}}^i$, their edit distance is large. We show that if we pick n random linear codes for $\mathbb{C}_{\mathsf{in}}^1, \cdots, \mathbb{C}_{\mathsf{in}}^n$, then this property holds with high probability. Furthermore, we can derandomize this by using a small biased sample space to generate the n generator matrices of $\mathbb{C}_{\mathsf{in}}^1, \cdots, \mathbb{C}_{\mathsf{in}}^n$. Roughly, this is because the property we want is *local* – it only looks at any two inner codewords x, y. By using a small biased sample space, we can show that (roughly) under the above conditions, any non-trivial parity of the bits (we treat a symbol in the alphabet of size $\mathsf{poly}(1/\varepsilon)$ as a binary string of length $O(\log(1/\varepsilon))$) of (x, y) has a small bias. Hence a standard XOR lemma implies the joint distribution of (x, y) is close to uniform. Since $n' = O(\log n)$, we only need to look at $\mathsf{poly}(n)$ such pairs of (x, y). Thus it suffices to choose the error in the small biased sample space to be $1/\mathsf{poly}(n)$. This gives us a sample space of size $\mathsf{poly}(n)$ and we can exhaustively search for a good construction. This gives us n different inner codes with rate $\Omega(\gamma)$.

Using these inner codes, it is now relatively straightforward to argue about the parameters of the concatenated code. The rate is $\Omega(\gamma^2) = \Omega(\varepsilon^2)$. To argue about the distance, we consider the LCS between any two different codewords C_1, C_2, and divide it sequentially into blocks according to the inner codewords of C_1. Each block now covers a substring of C_2. Intuitively, by the property of our inner codes, each block contains only a small number of matches compared to the total size of this block in C_1 and the substring covered in C_2, unless it is a 0 inner codeword in C_1 and is matched to another 0 inner codeword in C_2, or it is a match between the same inner codeword in a single inner code $\mathbb{C}_{\mathsf{in}}^i$. However our outer code guarantees that the latter cannot happen too many times (i.e., at most $O(\gamma n)$ times). Therefore the LCS has length at most $O(\gamma n n')$. By choosing γ appropriately, the code can correct $1-\varepsilon$ fraction of insdel errors.

We present a simple polynomial time decoding algorithm. Given any received string y, we consider the partition of y into n substrings $y_1, \cdots, y_n$ such that $y = y_1 \circ y_2 \circ \cdots \circ y_n$, where each y_i can be the empty string. For each y_i, we find the closest codeword $x_i \in \mathbb{C}_{\mathsf{in}}^i$ in edit distance and record their edit distance Δ_i. We then minimize $\Delta = \sum_{i\in[n]} \Delta_i$, by using a simple dynamic programming. We show that as long as there are not too many errors, by using the optimal partition returned from the dynamic programming, one can correctly recover a small fraction of the outer codewords. Intuitively, this is because if the partition results in too many errors in the recovered outer codewords, then again by the property of our inner codes, the quantity Δ will be very large, unless there are a lot of errors. We then use a list decoding algorithm for the Reed-Solomon code to get a list of candidate codewords, and search the list to find the correct codeword, which is the one closest to y in edit distance. For technical reasons, this decreases the rate of the code to $\Omega(\varepsilon^4)$.

Constructions with high rate. Now we explain our construction with high rate and polynomial time encoding and decoding. We first exhibit a warm-up construction achieving rate $1/3 - \gamma$. Then we improve the rate to $1/2 - \gamma$, while the construction will be significantly more involved due to additional issues arising in the analysis.

Inheriting the structure of the general construction, our first construction is as the following. The outer code is a Reed-Solomon code with block length n, alphabet size n, relative distance δ and rate $(1 - \delta)$, where $\delta = \gamma/2$. To achieve a high rate, we will design the inner codes to have a large rate, ideally close to $1/2$. At the same time, we also need to ensure that the code can correct a positive constant fraction of errors, thus we want to make sure that the LCS between any two different codewords is not too large.

As before, we will design the inner codes such that ideally, codewords from different inner codes are far away from each other (or equivalently, have small LCS). However, there are additional issues in the analysis of the LCS. First, the 0 codewords from different inner codes are always the same. This is inevitable since we are dealing with linear codes. Second, in a matching between two different codewords C_1, C_2, some inner codeword of C_1 may be matched to a substring of the concatenation of two adjacent inner codewords of C_2. Thus it is not enough to just ensure that codewords from different inner codes are far away from each other. We note that this issue also occurs in our constructions for high noise. However, there we designed the inner codes to have small rate but large distance, so the LCS between different inner codewords is quite small. When some inner codeword of C_1 is matched to a substring of the concatenation of two adjacent codewords of C_2, the size of the matching in this part at most doubles the size of the LCS between two different inner codewords, and is affordable in that case. Here however, since we are trying to achieve a high rate, the distance between two different inner codewords becomes quite small, and the LCS becomes relatively large (e.g., larger than $1/2$ fraction). Hence, we cannot afford to double this size.

On a high level, we resolve the second issue by strengthening our local property of inner codes, while our analysis will show that the first issue can also be resolved as a consequence. We begin by discussing the local property we need to achieve rate $1/3 - \gamma$. The distinct binary inner codes $\mathbb{C}_{\mathsf{in}}^1, \mathbb{C}_{\mathsf{in}}^2, \ldots, \mathbb{C}_{\mathsf{in}}^n$ are constructed to have block length n', message length $k' = (1/3 - \gamma/2)n'$, with the following property: for every $i, j \in [n]$, for every codeword w in $\mathbb{C}_{\mathsf{in}}^i$, for every two codewords u, v from two adjacent inner codes $\mathbb{C}_{\mathsf{in}}^j, \mathbb{C}_{\mathsf{in}}^{j+1}$, unless $w = u$ or $w = v$, the distance between w and any substring of $u \circ v$ is at least $d' = \Omega(n')$. We first explain why this property implies a good decoding radius and then explain how to construct these inner codes.

We show the decoding radius by directly providing the following decoding algorithm. On an input y which is a corrupted version of a codeword z, the algorithm first finds a string $\tilde{z} \in \{0,1\}^{nn'}$ which has a maximum block matching with y. A block matching is defined to be a set of matches where each match, denoted as $(i, [\alpha, \beta])$, consists of a non-zero inner codeword $u \in \mathbb{C}_{\mathsf{in}}^i$ and a substring $y_{[\alpha,\beta]}$, such that their edit distance is at most $d'/2$. Furthermore, the matching is monotone in the sense that the substrings of y involved in the matching do not overlap and the matches cannot cross. We call u a candidate string for the i-th block. We give a simple dynamic programming algorithm to find a maximum block matching together with a corresponding sequence of candidates. To construct $\tilde{z}$, we first fill these candidates to their corresponding blocks and then set all the other blocks to be 0.

Now we show that as long as there are at most $\rho nn'$ errors for some small constant $\rho > 0$, $\tilde{z}$ agrees with z in most of the blocks (inner codewords). To show this, divide z into blocks $z^1 \circ z^2 \circ \cdots \circ z^n$ such that each z^i corresponds to an inner codeword. Similarly, divide y into blocks $y^1 \circ y^2 \circ \cdots \circ y^n$ such that each y^i is the corrupted version of z^i. Notice that there can

be at most $\frac{\rho n n'}{d'/2} = (c\gamma)n$ blocks with at least $d'/2$ errors, for some constant $c = c(\rho)$. So the maximum block matching has size at least $\hat{n} - c\gamma n$ where $\hat{n}$ is the number of non-zero blocks in z. Now consider a maximum block matching and the sequence of candidates returned by the algorithm. We show that there are at most $c\gamma n$ candidates that are not equal to the corresponding blocks of z, by using the local property. As we fill all the other blocks to be 0, this also implies there are at most $c\gamma n$ zero-blocks being incorrectly recovered. Hence the algorithm correctly recovers $1 - O(c\gamma)$ fraction of blocks in z. By taking c (and thus also ρ) to be a small enough constant, one can use the list-decoding algorithm of Reed-Solomon codes to recover z.

Next, we explain how to construct the inner codes. We start by considering a random construction, that is, all the inner codes are independent random linear codes. We show the local property holds with high probability. Consider arbitrary codewords $w \in \mathbb{C}_{\mathsf{in}}^i \backslash \{0\}, u \in \mathbb{C}_{\mathsf{in}}^j, v \in \mathbb{C}_{\mathsf{in}}^{j+1}$ for some $i, j \in [n]$, where $w \neq u$ and $w \neq v$. Here the inequality means the two codewords are either from different inner codes or they correspond to different messages in one inner code. Suppose there is a substring w' of $u \circ v$, which has distance $< d'$ to w. So the LCS between w and w' should be $\ell \geqslant \frac{|w|+|w'|-d'}{2}$. Notice that $\ell \leqslant |w| \leqslant n'$. Consider any monotone alignment between w and w'. Because $w \neq u, w \neq v$ and the inner codes are all independent and generated randomly, by a similar argument as in [6], the event that the alignment is indeed a matching of bits happens with probability at most $2^{-\ell}$. We then apply a union bound over all possible alignments of size ℓ and all possible codewords w, u, v. A key observation is that the number of all possible codewords w, u, v is $2^{3k'}$ since we have three different codewords here. However, we have $\ell \leqslant n'$. Therefore for the union bound to work, we have to set $k' < n'/3$. This is the reason that we can only achieve rate close to $1/3$ with this construction.

Next, we derandomize the construction by replacing the uniform randomness used with an ε-biased distribution. Here, as before, we crucially use the fact that our property for the inner codes is local: the only place where we use randomness is when we bound the probability that an alignment is a valid matching, and it only involves three codewords. Since $n' = O(\log n)$, by using a standard XOR Lemma and taking $\varepsilon = 1/\mathsf{poly}(n)$, we can argue that when restricted to any three codewords, the ε-biased distribution is $1/\mathsf{poly}(n)$ close to the uniform distribution in statistical distance. This is enough for the union bound since there are at most $\mathsf{poly}(n)$ such triples w, u, v.

Since we only need $O(\log n)$ random bits to generate the above ε-biased distribution, one can exhaustively search for a good construction that satisfies our local property. This also takes polynomial time since one only needs to check every triple of inner codewords.

In our improved construction, we add new ideas to bypass the rate $1/3$ barrier in the above construction, by giving a new local property of the inner codes. Recall that the reason we need to choose $k' < n'/3$ in the above construction is that the alignment we consider in the local property consists of matches that involve three different codewords, which results in a $2^{3k'}$ term in the union bound, but the alignment has size at most n'. In the new local property, we generalize this by considering alignments that involve $2s + 1$ different codewords for some integer s. In a simplified version, consider any two different codewords C_1, C_2 of the concatenated codes and an LCS between them, we analyze any s consecutive inner codewords in C_1, and how they can be matched to a substring in C_2. Note that the s consecutive inner codewords cannot be matched to a substring with length much larger than sn', or there are already many unmatched bits in C_2. So we can imagine a new local property like the following: let w be the concatenation of any s adjacent inner codewords, and u be the concatenation of any $s + 1$ adjacent inner codewords. As long as the codewords in w and

u are sufficiently different, the distance between w and any substring of u is at least $\Omega(n')$. The idea is that an alignment between w and u can have size up to sn', while the union bound gives a $2^{(2s+1)k'}$ term. Thus we can potentially achieve $k' < \frac{s}{2s+1}n'$, and if s is large enough, the rate is close to $1/2$. Note that the warm-up construction i corresponds to the case of $s = 1$.

However, it is not straightforward to make this idea work. The main issue is that unlike the simple case of $s = 1$, when we consider s consecutive inner codewords for $s > 1$, there can be multiple 0 codewords in them, which can potentially be matched to the 0 codewords in u. Furthermore, there can be inner codewords in w and u that correspond to the same message in a single inner code. These issues will increase the probability that the alignment is a valid matching and can cause the union bound to fail. To fix this, we require the "unique" blocks in w to be dense. Specifically, we define a unique block of w (or u) to be a non-zero inner codeword such that either no block of u (or w) is in the same inner code with it, or any block of u (or w) in the same inner code with it corresponds to a different message. Now we define the following new local property:

For every w which is a sequence of $t = O(\frac{\log \frac{1}{\gamma}}{\gamma^2})$ consecutive inner codewords, every u which is a sequence of $t + 1$ consecutive inner codewords, and every w' which is a substring of u, the distance between w and w' is at least $d' = \Omega(\gamma n')$, as long as the number of unique blocks in w or u is at least $s = \Omega(\gamma t)$. By the distance property of the outer code, for any two different concatenated codewords, in at least one of them, the fraction of such t consecutive inner codewords with at least s unique blocks is a constant.

Using this new property, we can design a similar decoding algorithm as that of the first construction, and with a similar analysis, achieve decoding radius $\Omega(\gamma^3 n/\log \frac{1}{\gamma})$.

We defer these details to the technical part and mainly explain here how to construct the inner codes with the new property and why this indeed gives a rate of $1/2 - \gamma$.

Similar to before, we start with a construction where all inner codes are independent random linear codes, and later derandomize it with an ε-biased space. As long as the parameters s, t are constants, it is easy to see that the derandomization step still works. Therefore, now we only focus on the random construction and argue that the new local property holds with high probability. For this, we use a delicate combinatorial and probabilistic argument.

Suppose the property is not satisfied with some concatenated codewords C_1, C_2. Then there exists a w' such that the edit distance between w, w' is less than d', which implies the LCS between w and w' is $\ell > (|w| + |w'| - d')/2$. Consider an arbitrary monotone alignment M between w and w' of size ℓ. We have two cases. The first case is that there is a pair of indices (i, j) in M such that $|i - j| \geqslant d'$. This implies that there cannot be any pair of indices (i', j') in M such that $i' = j'$, for otherwise there are already at least d' bits in C_1 or C_2 that are not matched. Let $\hat{t}$ be the larger number of non-zero blocks in w and w'. Note that $\hat{t} \geqslant s$. Since all inner codes are independent and random, and every pair of indices (i, j) in M has $i \neq j$, the probability that M is a matching is at most $2^{-((\hat{t}-1)n' - O(d'))}$ (w' can have length as small as $(t + 1)n - n - d'$). Now if we apply the union bound, the main term is actually the total number of possible tuples of the non-zero inner codewords. Since there are at most $2\hat{t}$ non zero blocks in w and u, this number is at most $2^{2\hat{t}k'}$. Thus as long as s is a large enough integer, the rate of the code can approach $1/2$.

The second case is that every pair of indices (i, j) in M has $|i - j| < d'$. In this case, we focus on the unique blocks. Let s' be the larger number of unique blocks in w and u, and for simplicity assume u has more unique blocks. We delete all matches where the endpoint in u is not in a unique block, or the endpoint in w falls out of the block at the same position as the block in u which contains the endpoint in u. Thus we attain a trimmed alignment

M'. Under the assumed condition in this case, we don't lose too many matches. Indeed the number of matches left is at least $\ell' = s'(n' - d') - n' - d'$. We now upper bound the probability that there exists such an M' which is a valid matching. Since this event is implied by the original event, this also provides an upper bound of the original event.

The probability that any M' is a valid matching is $2^{\ell'}$, by our definition of unique blocks. Now in the union bound, the main term turns out to be the total number of possible tuples of the inner codewords corresponding to the s' unique blocks in u and the other s' blocks at the same positions in w, which is roughly $2^{(2s')k'}$. Notice that $s' \geqslant s$. Thus in this case, as long as s is large enough, the rate of the code can also approach $1/2$.

The existence of linear concatenated codes matching random linear codes. This part is similar in spirit to Thommesen's work [23], which shows the existence of binary linear concatenated codes with Reed-Solomon outer codes that asymptotically meet the Gilbert-Varshamov bound. In particular, we also take a Reed-Solomon code as the outer code, and use an independent random linear inner code for every symbol of the outer codeword. Interestingly, here we take the outer code to be a $[n, k = (1-\gamma)n/2, d = (1+\gamma)n/2]_q$ Reed-Solomon code with $q = \Theta(n)$, i.e., the rate of the outer code is less than $1/2$. On the other hand, we take all inner codes to have rate 1. Using a careful probabilistic counting argument together with an estimate of the number of Reed-Solomon codewords with a specific weight (as done in [23]), we can prove the existence of linear concatenated insdel codes with parameters as in Theorem 4.

The choice of the parameters of the outer code is different from our explicit constructions, suggesting that maybe different constructions based on these parameters can lead to better explicit linear insdel codes.

Organization of the paper. Our general construction is exhibited in Section 3. The high error construction and its analysis are given in Section 4. We put the technical details of the rest of our results in the full version.

2 Preliminaries

Notation. Let Σ be an alphabet. For a string $x \in \Sigma^*$,

1. $|x|$ denotes the length of the string.
2. $x[i, j]$ denotes the substring of x from position i to position j (both endpoints included).
3. $x[i]$ denotes the i-th symbol of x.
4. $x \circ x'$ denotes the concatenation of x and some other string $x' \in \Sigma^*$.
5. For a string s which is a concatenation of shorter strings $s_1, s_2, \ldots, s_t$, the i-th block of s refers to s_i.

▶ **Definition 9** (Edit distance and Longest Common Subsequence). *For any two strings $x, y \in \Sigma^n$, the edit distance $\Delta_E(x, y)$ is the minimum number of edit operations (insertions and deletions) required to transform x into y.*[2] *A longest common subsequence of x and y is a longest pair of subsequences of x and y that are equal as strings. We use $\mathsf{LCS}(x, y)$ to denote the length of a longest common subsequence between x and y.*

[2] The standard definition of edit distance also allows substitution, but for simplicity we only consider insertions and deletions here, as a substitution can be replaced by a deletion followed by an insertion.

Note that $\Delta_E(x,y) = |x| + |y| - 2 \cdot \mathsf{LCS}(x,y)$. We use $\Delta_H(x,y)$ to denote the Hamming distance between two strings x and y.

▶ **Definition 10.** *An (n,m,d)-code C is an error-correcting code (for Hamming errors) with codeword length n, message length m, such that the Hamming distance between every pair of codewords in C is at least d.*

▶ **Definition 11.** *Fix an alphabet Σ, an error-correcting code $C \subseteq \Sigma^n$ for edit errors with message length m and codeword length n consists of an encoding function $Enc : \Sigma^m \to \Sigma^n$ and a decoding function $Dec : \Sigma^* \to \Sigma^m$. The code can correct k edit errors if for every y, s. t. $\Delta_E(y, Enc(x)) \leqslant k$, we have $Dec(y) = x$. The rate of the code is defined as $\frac{m}{n}$.*

We say C is a linear code if the alphabet Σ is a finite field F_q and the encoding function $\mathrm{Enc} : \mathsf{F}_q^m \to \mathsf{F}_q^n$ is a F_q-linear map.

We use the following list decoding algorithm for Reed-Solomon codes due to Guruswami and Sudan [11].

▶ **Theorem 12.** *Given a family of Reed-Solomon codes of message rate γ, an error rate of $\varepsilon = 1 - \sqrt{\gamma}$ can be list-decoded in polynomial time.*

We use U_n to denote the uniform distribution on $\{0,1\}^n$.

▶ **Definition 13.** *An ε-biased distribution X over $\{0,1\}^n$ is such that for any $S \subseteq [n]$, $|\Pr[\bigoplus_{i \in S} X_i = 1] - 1/2| \leqslant \varepsilon$. A function $g : \{0,1\}^s \to \{0,1\}^n$ is an ε-biased generator if $g(U_s)$ is an ε-biased distribution.*

The following ε-biased generator is used.

▶ **Theorem 14** ([2])**.** *For every $n \in \mathbb{N}$, every $\varepsilon \in (0,1)$, there exists an explicit ε-biased generator $\{0,1\}^s \to \{0,1\}^n$ with $s = O(\log n + \log(1/\varepsilon))$.*

We also need the following XOR lemma.

▶ **Lemma 15** (XOR Lemma)**.** *The statistical distance between an ε-biased distribution and a uniform distribution, both over $\{0,1\}^n$, is at most $\varepsilon\sqrt{2^n}$.*

3 General Construction of Our Codes

All our codes follow the general strategy of code concatenation, which we describe below.

The outer code $\mathbb{C}_{\mathsf{out}}$ with encoding function : $\mathsf{Enc}_{out} : \Sigma_{\mathrm{out}}^k \to \Sigma_{\mathrm{out}}^n$ is an $[n,k,d]$ Reed Solomon Code for Hamming errors. We then use n different inner codes $\mathbb{C}_{\mathsf{in}}^1, \ldots, \mathbb{C}_{\mathsf{in}}^n$, such that for any $i \in [n]$, $\mathbb{C}_{\mathsf{in}}^i$ is a linear code $\mathsf{Enc}_{in}^i : \Sigma_{\mathrm{out}} \to \Sigma_{\mathrm{in}}^{n'}$, where n' is the block length of the inner code. In this paper Σ_{in} always has constant size and we let $n' = \Theta(\log n)$. For different applications, we will need the inner codes to have slightly different properties.

Our final code $\mathbb{C}$ works naturally by first encoding the message using the outer code, then encoding each symbol of the outer code using the inner codes. This gives a codeword over Σ_{in} with length $N = n \cdot n'$. If the outer code and all the inner codes are linear, the concatenated code is also linear.

4 Constructions For High Noise

In this section we give our linear codes that can correct $1 - \varepsilon$ fraction of insdel errors, for any constant $\varepsilon > 0$. Our codes can still achieve a constant rate.

The construction. Following our general construction, we take $\mathbb{C}_{\mathsf{out}}$ to be an $[n, k, d]_n$ Reed-Solomon code with $|\Sigma_{\text{out}}| = n$, $k = \gamma n$ and $d = (1-\gamma)n$ for some constant $\gamma > 0$ to be chosen later. We construct n different inner codes $\mathbb{C}^1_{\mathsf{in}}, \ldots, \mathbb{C}^n_{\mathsf{in}}$ with alphabet size $|\Sigma_{\text{in}}| = \mathsf{poly}(1/\gamma)$, message length $k' = \Theta(\log n)$, and codeword length $n' = \Theta(\log n)$, with the following property.

▶ **Property 1.** *For any two codewords $x \in \mathbb{C}^i_{in}, y \in \mathbb{C}^j_{in}$, if either of the following two conditions holds:*
1. *$i \neq j$, and $x \neq 0^{n'}$ or $y \neq 0^{n'}$.*
2. *$i = j$ and $x \neq y$.*

Then we have $\mathsf{LCS}(x, y) \leqslant \gamma n'$.

▶ **Lemma 16.** *There exists an efficient construction of n inner codes $\mathbb{C}^1_{in}, \ldots, \mathbb{C}^n_{in}$, where each $\mathbb{C}^i_{in}$ has alphabet size $|\Sigma_{in}| = \mathsf{poly}(1/\gamma)$ and rate $\Omega(\gamma)$.*

Proof. We first show that if we pick n independent random linear inner codes $\mathbb{C}^1_{\mathsf{in}}, \ldots, \mathbb{C}^n_{\mathsf{in}}$ over an alphabet size $|\Sigma_{\text{in}}| = \mathsf{poly}(1/\gamma)$, then they satisfy Property 1 with high probability. We then show how to derandomize the construction using a small biased sample space.

Fix a field F_q. For each $\mathbb{C}^i_{\mathsf{in}}$ we independently pick $\log n$ uniformly random vectors in $\mathsf{F}_q^{n'}$ with $n' = \Theta(\log n/\gamma)$ as the basis for $\mathbb{C}^i_{\mathsf{in}}$, or equivalently, the rows in the generating matrix of $\mathbb{C}^i_{\mathsf{in}}$. We bound the probability that there exist two codewords $x \in \mathbb{C}^i_{\mathsf{in}}, y \in \mathbb{C}^j_{\mathsf{in}}$ that satisfy the conditions of Lemma 16 but $\mathsf{LCS}(x, y) > \gamma n'$.

▷ Claim 17. Consider any fixed common subsequence between x and y of length t, where the corresponding indices in x are $\{s_1, \cdots, s_t\}$ and the corresponding indices in y are $\{r_1, \cdots, r_t\}$. Then

$$\Pr[\forall k \in [t], x_{s_k} = y_{r_k}] \leqslant q^{-t}.$$

To prove the claim we have two cases.

Case 1: $i \neq j$, and $x \neq 0^{n'}$ or $y \neq 0^{n'}$. This is the easy case. Since $i \neq j$, and all the entries in the generating matrices of $\mathbb{C}^i_{\mathsf{in}}$ and $\mathbb{C}^j_{\mathsf{in}}$ are chosen independently uniformly from F_q, we know that the events $x_{s_k} = y_{r_k}$ are all independent, even if $x = 0^{n'}$ or $y = 0^{n'}$. Furthermore, the probability of each such event is $1/q$. Hence the claim follows.

Case 2: $i = j$. In this case, the events $x_{s_k} = y_{r_k}$ are not necessarily all independent. However, the claim still follows from the following claim in [6], which deals exactly with this situation.

▷ Claim 18. [Claim 4.2 of [6]] Let G be a random generating matrix for a linear code over F_q. For any two different messages x^i, x^j and codewords $C^i = x^i G, C^j = x^j G$, consider any fixed common subsequence between C^i and C^j of length t, where the corresponding indices in C^i are $\{s_1, \cdots, s_t\}$ and the corresponding indices in C^j are $\{r_1, \cdots, r_t\}$. Then

$$\Pr[\forall k \in [t], C^i_{s_k} = C^j_{r_k}] \leqslant q^{-t}.$$

Now by a union bound, and noticing that the total number of possible cases where two strings of length n' have a common subsequence of length $\gamma n'$ is at most $\binom{n'}{\gamma n'}^2$, we have

$$\Pr[\text{Property 1 does not hold}] \leqslant n^2 q^{2\log n} \binom{n'}{\gamma n'}^2 q^{-\gamma n'} \leqslant \left(\frac{e}{\gamma}\right)^{2\gamma n'} n^2 q^{2\log n - \gamma n'}.$$

Therefore, one can set $q = (\frac{e}{\gamma})^3$ and $n' = \Theta(\log n/\gamma)$ so that the above probability is $q^{-\Omega(\log n)} = 1/\mathsf{poly}(n)$.

Next we show how to derandomize the above construction using a small biased space. Without loss of generality we assume the field we use is F_q with $q = 2^\ell$. Thus, by choosing an arbitrary basis $b_1, \cdots, b_\ell$ in F_q we can identify the field with the vector space F_2^ℓ, such that any $a \in \mathsf{F}_q$ can be expressed as $a = \sum_{i\in[\ell]} a_i b_i$, where $\forall i, a_i \in \mathsf{F}_2$. In this way, the generating matrix of each $\mathbb{C}_{\mathsf{in}}^i$ can be viewed as consisting of $\ell n' \log n = \Theta(\ell \log^2 n)$ bits.

We pick a τ-biased sample space with $n\ell n' \log n$ bits for some $\tau = 1/\mathsf{poly}(n)$ to be chosen later. Note that by Theorem 14 this can be generated by $O(\log n)$ uniform random bits.

Given ℓ bits $a_1, \cdots, a_\ell$ which defines the field element $a = \sum_{i\in[\ell]} a_i b_i$, and any $p \in \mathsf{F}_q$, consider the operation $p \cdot a$ and the corresponding coefficient in the basis b_1. It's not hard to see that this is a F_2-linear function (i.e., a parity) of $a_1, \cdots, a_\ell$. Call this parity $L_p(a_1, \cdots, a_\ell)$. We have the following claim.

▷ Claim 19. $L_p(a_1, \cdots, a_\ell) \equiv 0$ if and only if $p = 0$.

Proof of the claim. The "if" part is trivially true. For the other part, note that if $p \neq 0$ then $pb_1, \cdots, pb_\ell$ must also be linearly independent and thus form a basis of F_q. Therefore, some pb_i must have a non-zero coefficient in b_1 and thus $L_p(a_1, \cdots, a_\ell)$ has a term a_i in the parity, therefore it cannot be the 0 function. ◁

Note that there are altogether 2^ℓ different parity functions involving $a_1, \cdots, a_\ell$, and $q = 2^\ell$ elements in F_q. Thus the previous claim immediately implies the following claim.

▷ Claim 20. Any parity function involving $a_1, \cdots, a_\ell$ is equivalent to $L_p(a_1, \cdots, a_\ell)$ for some $p \in \mathsf{F}_q$.

Now consider the two codewords $x \in \mathbb{C}_{\mathsf{in}}^i, y \in \mathbb{C}_{\mathsf{in}}^j$. Let x_0 and y_0 be the corresponding messages for x and y respectively. We now have the following claim.

▷ Claim 21. Unless $i = j$ and $y_0 = p \cdot x_0$ or $x_0 = p \cdot y_0$ for some $p \in \mathsf{F}_q$, under the τ-biased sample space, the joint distribution of (x, y) is $q^{n'}\tau$-close to the uniform distribution over $\mathsf{F}_q^{2n'}$.

Proof of the claim. Let $x = (x_1, \cdots, x_{n'}) \in \mathsf{F}_q^{n'} = \mathsf{F}_2^{\ell n'}$ and $y = (y_1, \cdots, y_{n'}) \in \mathsf{F}_q^{n'} = \mathsf{F}_2^{\ell n'}$. Consider any non-trivial parity of the $2\ell n'$ bits, which by Claim 20 corresponds to the coefficient of b_1 under some function $\sum_{k\in[n']}(p_k^x x_k + p_k^y y_k)$, where $\forall k, p_k^x, p_k^y \in \mathsf{F}_q$, and they are not all 0.

If $i \neq j$, then $\sum_{k\in[n']} p_k^x x_k$ and $\sum_{k\in[n']} p_k^y y_k$ use different bits in the τ-biased sample space. Since x, y are not both $0^{n'}$, the resulted parity is a non-trivial parity of the bits in the sample space, which by definition has bias at most τ.

Otherwise we have $i = j$. Let G be the generating matrix for $\mathbb{C}_{\mathsf{in}}^i$, thus $x = x_0 G$ and $y = y_0 G$. For any $k \in [n']$, let G_k be the k'th column of G. We have

$$\sum_{k\in[n']}(p_k^x x_k + p_k^y y_k) = \sum_{k\in[n']}(p_k^x x_0 G_k + p_k^y y_0 G_k) = \sum_{k\in[n']}(p_k^x x_0 + p_k^y y_0) G_k.$$

Notice that each entry in each G_k is independently uniformly chosen from $\mathsf{F}_q = \mathsf{F}_2^\ell$. Thus by Claim 19 if the coefficient of the above sum in b_1 is the trivial parity 0, then we must have $\forall k \in [n'], p_k^x x_0 + p_k^y y_0 = 0$. This implies that either $y_0 = p \cdot x_0$ or $x_0 = p \cdot y_0$ for some $p \in \mathsf{F}_q$.

Otherwise, the parity is a non-trivial parity of the bits in the sample space, which by definition has bias at most τ. Now, by Lemma 15, the joint distribution of (x, y) is $q^{n'}\tau$-close to the uniform distribution over $\mathsf{F}_q^{2n'}$. ◁

Back to the proof of our lemma. If the conditions of the above claim hold, then the joint distribution of (x, y) is $q^{n'}\tau$-close to the uniform distribution. Hence, the probability that there exists any common subsequence of length $\gamma n'$ between x and y is at most $\binom{n'}{\gamma n'}^2 q^{-\gamma n'} + q^{n'}\tau$.

On the other hand, if the conditions of the above claim do not hold, then without loss of generality assume that $y_0 = p \cdot x_0$ for some $p \in \mathsf{F}_q$. Hence $p \neq 1$. In this case, notice that we also have $y = p \cdot x$, and thus the probability that there exists any common subsequence of length $\gamma n'$ between x and y is completely determined by the random variables in x. Note that any non-trivial parity of the bits in x is also a non-trivial parity of the bits of the τ-biased sample space, which has bias at most τ. By Lemma 15, the distribution of x is $q^{n'/2}\tau$-close to being uniform on $\mathsf{F}_q^{n'}$.

We have the following claim.

▷ Claim 22. Let x be a uniformly random vector in $\mathsf{F}_q^{n'}$, and $y = p \cdot x$. Then

$$\Pr[\exists \text{a common subsequence of length } t \text{ between } x \text{ and } y] \leqslant \binom{n'}{t}^2 q^{-t}.$$

Proof of the claim. Consider any fixed common subsequence of length t between x and y. Assume where the corresponding indices in x are $\{s_1, \cdots, s_t\}$ and the corresponding indices in y are $\{r_1, \cdots, r_t\}$, such that $s_1 < s_2 < \cdots < s_t$ and $r_1 < r_2 < \cdots < r_t$. For any $k \in [t]$, let $m_k = \max(s_k, r_k)$. Notice that $m_1 < m_2 < \cdots < m_t$. Define E_k to be the event $x_{s_k} = y_{r_k}$.

For each $k \in [t]$, if $s_k = r_k$, then

$$\Pr[E_k] = \Pr[x_{s_k} = p \cdot x_{s_k}] = \Pr[x_{s_k} = 0] = \frac{1}{q}.$$

Furthermore, since $s_k = r_k = m_k$ is larger than all $\{s_{k'}, r_{k'}, k' < k\}$, the event E_k is independent of all $\{E_{k'}, k' < k\}$. Thus

$$\Pr[E_k | \{E_{k'}, k' < k\}] = \frac{1}{q}.$$

Otherwise, $s_k \neq r_k$ and without loss of generality assume $s_k > r_k$. This means $s_k = m_k$ and is larger than all $\{s_{k'}, r_{k'}, k' < k\}$. We can now first fix all $\{x_{s_{k'}}, y_{r_{k'}}, k' < k\}$ and y_{r_k}, and conditioned on this fixing x_{s_k} is still uniform over F_q. Thus

$$\Pr[E_k] = \Pr[x_{s_k} = p \cdot x_{r_k}] = \frac{1}{q}.$$

Note that any such fixing also fixes the outcomes of all $\{E_{k'}, k' < k\}$. Hence we also have

$$\Pr[E_k | \{E_{k'}, k' < k\}] = \frac{1}{q}.$$

Therefore, the above equation holds in all cases, and for all k. This gives

$$\Pr[\bigcap_{k \in [t]} E_k] \leqslant q^{-t},$$

and the claim follows from a union bound. ◁

Since x is $q^{n'/2}\tau$-close to being uniform on $\mathsf{F}_q^{n'}$, the probability that there exists any common subsequence of length $\gamma n'$ between x and y is at most $\binom{n'}{\gamma n'}^2 q^{-\gamma n'} + q^{n'/2}\tau$ in this case.

To summarize, using the τ-biased sample space we always have that the probability that there exists any common subsequence of length $\gamma n'$ between x and y is at most $\binom{n'}{\gamma n'}^2 q^{-\gamma n'} + q^{n'}\tau$. By another union bound, we have

$$\begin{aligned}\Pr[\text{Property 1 does not hold}] &\leqslant n^2 q^{2\log n}\left(\binom{n'}{\gamma n'}^2 q^{-\gamma n'} + q^{n'}\tau\right)\\ &\leqslant \left(\frac{e}{\gamma}\right)^{2\gamma n'} n^2 q^{2\log n - \gamma n'} + n^2 q^{n'+2\log n}\tau.\end{aligned}$$

Therefore, one can still set $q = (\frac{e}{\gamma})^3$, $n' = \Theta(\log n/\gamma)$, and $\tau = q^{-\Omega(\log n/\gamma)} = 1/\mathsf{poly}(n)$ so that the above probability is $q^{-\Omega(\log n)} = 1/\mathsf{poly}(n)$.

Once we know this, we can exhaustively search the τ-biased sample space and find a sample point which gives us a construction that satisfies Property 1. Since we only have $\mathsf{poly}(n)$ sample points and checking each sample point takes polynomial time, altogether this takes polynomial time. ◀

Note that our concatenated code $\mathbb{C}$ now has rate $\Omega(\gamma^2)$. Further, Property 1 implies the following property:

▶ **Property 2.**
1. *$\forall i \neq j$, we have $\mathbb{C}_{in}^i \bigcap \mathbb{C}_{in}^j = \{0^{n'}\}$.*
2. *For any $i, j \in [n]$ and any two codewords $x \in \mathbb{C}_{in}^i, y \in \mathbb{C}_{in}^j$, if $x \neq y$ then $\mathsf{LCS}(x, y) \leqslant \gamma n'$.*

Let z be any substring of a codeword from the concatenated code $\mathbb{C}$, and assume z is a substring of $z_j \circ z_{j+1} \circ \cdots \circ z_{j+\ell}$, where $\forall t$, z_{j+t} is a codeword in $\mathbb{C}_{\mathsf{in}}^{j+t}$. We say the codewords $\{z_{j+t}, t = 0, \cdots, \ell\}$ contribute to the string z.

We now show that Property 1 and Property 2 give us the following lemma.

▶ **Lemma 23.** *Let x be a codeword from the code $\mathbb{C}_{in}^i$. Let z be any substring of a codeword from the concatenated code $\mathbb{C}$, and $\{z_{j+t}, t = 0, \cdots, \ell\}$ are the inner codewords contributing to z. If $\forall t$, $z_{j+t} \neq x$, then we have $\mathsf{LCS}(x, z) < 2\gamma(|x| + |z|)$.*

Proof. By Property 2, the longest common subsequence between x and any z_{j+t} has length at most $\gamma n'$. If $\ell = 1$, then we have

$$\mathsf{LCS}(x, z) \leqslant \gamma n' < 2\gamma(|x| + |z|).$$

Otherwise we have $\ell \geqslant 2$. Notice that $|z| > (\ell - 2)n'$. Thus we have

$$\mathsf{LCS}(x, z) \leqslant \ell\gamma n' \leqslant 2\gamma(\ell - 1)n' < 2\gamma(|x| + |z|).$$ ◀

We can now prove the following lemma.

▶ **Lemma 24.** *For any two different codewords $C_1, C_2 \in \mathbb{C}$, we have $\Delta_E(C_1, C_2) > 2(1-6\gamma)N$.*

Proof. We upper bound $\mathsf{LCS}(C_1, C_2)$ as follows. Consider a particular longest common subsequence and divide it sequentially according to the n inner codewords in C_1. Let the codewords in C_1 be $x_1, \cdots, x_n$ and the corresponding substrings in C_2 under the LCS be $z_1, \cdots, z_n$.

By Lemma 23, for any $i \in [n]$, we must have $\mathsf{LCS}(x_i, z_i) \leqslant 2\gamma(|x_i| + |z_i|)$, unless some inner codeword in z_i is equal to x_i. This could happen either because $x_i = 0^{n'}$ or because z_i contains part of x_i from exactly the i'th inner code. In the latter two cases, we call such an index i *bad*. Note that for a bad i we have $\mathsf{LCS}(x_i, z_i) \leqslant n'$, and there are at most γn such bad indices for either case, by our choice of the outer code. Let t be the number of bad indices, thus $t \leqslant 2\gamma n$. Therefore,

$$\begin{aligned}\mathsf{LCS}(x, z) &= \sum_{i \text{ is not bad}} \mathsf{LCS}(x_i, z_i) + \sum_{i \text{ is bad}} \mathsf{LCS}(x_i, z_i) \\ &\leqslant 2\gamma \sum_{i \text{ is not bad}} (|x_i| + |z_i|) + tn' \\ &< 2\gamma(2n'n) + 2\gamma nn' = 6\gamma N,\end{aligned}$$

where the last inequality follows from the fact that if the number of bad indices is larger than 0, then $\sum_{i \text{ is not bad}}(|x_i| + |z_i|) < 2n'n$. Therefore $\Delta_E(C_1, C_2) > 2N - 12\gamma N = 2(1 - 6\gamma)N$. ◀

Setting $\gamma = \varepsilon/6$, this gives the following theorem.

▶ **Theorem 25.** *For any constant $\varepsilon > 0$ there exists an efficient construction of linear insdel codes over an alphabet of size $\mathsf{poly}(1/\varepsilon)$, with rate $\Omega(\varepsilon^2)$ that can correct $1 - \varepsilon$ fraction of insdel errors (possibly inefficiently).*

References

1. Khaled A.S. Abdel-Ghaffar, Hendrik C. Ferreira, and Ling Cheng. On linear and cyclic codes for correcting deletions. In *2007 IEEE International Symposium on Information Theory (ISIT)*, pages 851–855, 2007.
2. Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.
3. J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss. A dna-based archival storage system. *ACM SIGARCH Comput. Archit. News*, 44:637–649, 2016.
4. Joshua Brakensiek, Venkatesan Guruswami, and Samuel Zbarsky. Efficient low-redundancy codes for correcting multiple deletions. *IEEE Transactions on Information Theory*, 64(5):3403–3410, 2018. Preliminary version in SODA 2016.
5. Boris Bukh, Venkatesan Guruswami, and Johan Håstad. An improved bound on the fraction of correctable deletions. *IEEE Trans. Information Theory*, 63(1):93–103, 2017. Preliminary version in SODA 2016. doi:10.1109/TIT.2016.2621044.
6. Kuan Cheng, Venkatesan Guruswami, Bernhard Haeupler, and Xin Li. Efficient linear and affine codes for correcting insertions/deletions. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–20, 2021. doi:10.1137/1.9781611976465.1.
7. Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Block Edit Errors with Transpositions: Deterministic Document Exchange Protocols and Almost Optimal Binary Codes. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.37.
8. Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Deterministic document exchange protocols, and almost optimal binary codes for edit errors. *Journal of the ACM (JACM)*, 69(6):1–39, 2022. Preliminary version in 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS).
9. Roni Con, Amir Shpilka, and Itzhak Tamo. Explicit and efficient constructions of linear codes against adversarial insertions and deletions. *IEEE Transactions on Information Theory*, 68(10):6516–6526, 2022. doi:10.1109/TIT.2022.3173185.

10 Roni Con, Amir Shpilka, and Itzhak Tamo. Reed solomon codes against adversarial insertions and deletions. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 2940–2945, 2022. doi:10.1109/ISIT50566.2022.9834672.

11 V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999. doi:10.1109/18.782097.

12 Venkatesan Guruswami, Bernhard Haeupler, and Amirbehshad Shahrasbi. Optimally resilient codes for list-decoding from insertions and deletions. In *Proccedings of the 52nd Annual ACM Symposium on Theory of Computing*, pages 524–537, 2020. doi:10.1145/3357713.3384262.

13 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*, pages 1–1, 2022. doi:10.1109/TIT.2022.3223023.

14 Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *IEEE Trans. Information Theory*, 63(4):1961–1970, 2017.

15 Bernhard Haeupler. Optimal document exchange and new codes for insertions and deletions. In *60th IEEE Annual Symposium on Foundations of Computer Science*, pages 334–347, 2019.

16 Bernhard Haeupler, Aviad Rubinstein, and Amirbehshad Shahrasbi. Near-Linear Time Insertion-Deletion Codes and (1+eps)-Approximating Edit Distance via Indexing. *Proceeding of the ACM Symposium on Theory of Computing (STOC)*, pages 697–708, 2019.

17 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: codes for insertions and deletions approaching the singleton bound. *Journal of the ACM (JACM)*, 68(5):1–39, 2021. Preliminary version in 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC).

18 Bernhard Haeupler, Amirbehshad Shahrasbi, and Madhu Sudan. Synchronization strings: List decoding for insertions and deletions. *Proceeding of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 76:1–76:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.76.

19 Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization strings: Channel simulations and interactive coding for insertions and deletions. *Proceeding of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 75:1–75:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.75.

20 Tomohiro Hayashi and Kenji Yasunaga. On the list decodability of insertions and deletions. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 86–90. IEEE, 2018.

21 Leonard J. Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Trans. Inf. Theory*, 45(7):2552–2557, 1999. Preliminary version in SODA 1997. doi:10.1109/18.796406.

22 Jin Sima and Jehoshua Bruck. Optimal k-deletion correcting codes. In *IEEE International Symposium on Information Theory*, pages 847–851, 2019. doi:10.1109/ISIT.2019.8849750.

23 C. Thommesen. The existence of binary linear concatenated codes with reed - solomon outer codes which asymptotically meet the gilbert- varshamov bound. *IEEE Transactions on Information Theory*, 29(6):850–853, 1983. doi:10.1109/TIT.1983.1056765.

24 Antonia Wachter-Zeh. List decoding of insertions and deletions. *IEEE Transactions on Information Theory*, 64(9):6297–6304, 2017.

25 S. M. Hossein Tabatabaei Yazdi, Ryan Gabrys, and Olgica Milenkovic. Portable and error-free dna-based data storage. *Scientific Reports*, 7:2045–2322, 2017. doi:10.1038/s41598-017-05188-1.