Coded trace reconstruction in a constant number of traces

Joshua Brakensiek
Computer Science Department
Stanford University
Stanford, California
jbrakens@cs.stanford.edu

Ray Li
Computer Science Department
Stanford University
Stanford, California
rayyli@cs.stanford.edu

Bruce Spang
Computer Science Department
Stanford University
Stanford, California
bspang@cs.stanford.edu

***Abstract*—The coded trace reconstruction problem asks to construct a code $C \subset \{0,1\}^n$ such that any $x \in C$ is recoverable from independent outputs ("traces") of x from a binary deletion channel (BDC). We present binary codes of rate $1-\varepsilon$ that are efficiently recoverable from $\exp(O_q(\log^{1/3}(\frac{1}{\varepsilon})))$ (a constant independent of n) traces of a BDC_q for any constant deletion probability $q \in (0,1)$. We also show that, for rate $1-\varepsilon$ binary codes, $\tilde{\Omega}(\log^{5/2}(1/\varepsilon))$ traces are required. The results follow from a pair of black-box reductions that show that average-case trace reconstruction is essentially equivalent to coded trace reconstruction. We also show that there exist codes of rate $1-\varepsilon$ over an $O_\varepsilon(1)$-sized alphabet that are recoverable from $O(\log(1/\varepsilon))$ traces, and that this is tight.**

***Keywords*-trace reconstruction; coded trace reconstruction; information theory; deletion channel; synchronization strings**

I. Introduction

The *trace reconstruction* problem was first proposed in [Lev01a], [Lev01b] and further developed in [BKKM04]. In trace reconstruction, we wish to recover an unknown binary string $x \in \{0,1\}^n$ given a few random subsequences of x. Each subsequence, or *trace*, is generated by sending x through the *binary deletion channel with deletion probability* q (BDC_q), which independently deletes each symbol of x with probability $q \in (0,1)$. In particular, the positions of the deleted bits are not known. For example, deleting either the first or second bit of "110" gives the trace "10".

Trace reconstruction has been primarily studied in two settings: *worst-case*, in which the input string x is chosen adversarially, and *average-case*, when the input string x is chosen uniformly at random over all possible n-bit strings. The fundamental question in both settings is to determine the minimum number of traces $T = T(n)$ needed in order to recover a length n string x correctly with high probability. In both settings, there is currently an exponential gap (as a function of n) for bounding $T(n)$ – see Section I-A for the best known bounds.

In this work, we consider an emerging [HM14], [CGMR20], [AVDiF19] variant of the trace reconstruction known as *coded trace reconstruction*. In this model, we want the smallest T such that there exists a high rate code $C \subset \{0,1\}^n$ such that, for an adversarially chosen $x \in C$, we can recover x with high probability from T traces. This model is directly motivated by DNA storage [YGM17], [CGMR20], in which data is stored as multiple encoded strands of DNA. Besides directly generalizing the trace reconstruction problem, coded trace reconstruction also generalizes the well-studied problem of determining the capacity of the binary deletion channel.

In this coded setting, we wish to design codes for trace reconstruction with high *rate*, which is defined[1] to be $\log|C|/n$. We consider the regime in which the rate is $1-\varepsilon$ (i.e., $|C| \approx 2^{(1-\varepsilon)n}$), where $\varepsilon \in (0,1)$ is a small constant or shrinking as a function of n. In particular, the key question we study is as follows.

Question I.1. For a given $\varepsilon \in (0,1)$ and positive integer n, what is the smallest T such that we can construct a binary code of rate $1-\varepsilon$ and length n recoverable from T traces?

Contributions.: We summarize the main contributions of our work below. See Section I-B for formal theorem statements. In all these results, we consider any constant $q \in (0,1)$.

1) **Binary codes with constant number of traces.** For $\varepsilon \in (0,1)$, we construct an infinite family of binary codes of rate $1-\varepsilon$ efficiently recoverable from a constant number of traces over the BDC_q (independent of n). This follows as an immediate corollary (Corollary I.5) of the following more general result we prove.
2) **Black-box upper bounds from average-case trace reconstruction.** We show that, if average-case trace reconstruction on length n strings succeeds with sufficiently high probability in $T(n)$ traces, then there exist rate $1-\varepsilon$ codes that are decodable from $T(\tilde{O}_q(1/\varepsilon))$ traces over the BDC_q (Theorem I.4). In particular, by a result in [HPP18], $\exp(O_q(\log^{1/3}(\frac{1}{\varepsilon}))) < \frac{1}{\varepsilon^{o(1)}}$ traces suffice (Corollary I.5).
3) **Black-box lower bounds from average-case trace reconstruction.** Conversely, we show that if average-case reconstruction on length n strings requires $T(n)$ traces, then reconstruction of any binary code of rate $1-\varepsilon$ requires $T(\tilde{\Omega}_q(1/\sqrt{\varepsilon}))$ traces over the

[1]All logs and exps are base 2 unless otherwise specified.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00052

BDC_q (Theorem I.8). In particular, by a recent result [Cha19], $\tilde{\Omega}_q(\log^{5/2}(1/\varepsilon))$ traces are required (Corollary I.9).

4) **Near-equivalence of average-case and coded trace reconstruction.** The two black-box reductions together imply that estimating the optimal number of traces for a code of rate $1-\varepsilon$ is equivalent to closing the lower and upper bounds within a polynomial for average-case trace reconstruction on strings of length $\mathrm{poly}(1/\varepsilon)$ (Remark I.11).
5) **Optimal number of traces for constant-sized alphabet.** We also consider the coded trace reconstruction problem over larger alphabets than binary. In particular, we give rate[2] $1-\varepsilon$ codes over an alphabet of size $O_\varepsilon(1)$ that are efficiently encodable and decodable from $O(\log_{1/q}(1/\varepsilon))$ traces (Theorem I.12). We show this is optimal up to a constant factor (Theorem I.13). This shows that coded trace reconstruction is strictly easier for larger alphabets than for binary alphabets. To the best of our knowledge, this is the first non-trivial tight result in *any* model of trace reconstruction for the deletion channel.

A. Related work

We now discuss how our results are situated at the intersection of the trace reconstruction and coding theory literature.

Classical trace reconstruction.: One of the main motivations for trace reconstruction is the application to DNA sequencing in computational biology [BKKM04]. When DNA is sequenced, the results may have insertion, deletion, and substitution errors. The original goal of trace reconstruction was to understand a simplified model of how an unknown piece of DNA can be recovered from its sequences. Recently, sequencing has been used for DNA storage [YGM17], [CGMR20], in which data is encoded so that it can be stored in DNA. This code needs to be decodable using a trace reconstruction-like process, while being high rate and using as few traces as possible.

The theoretical worst-case setting of trace reconstruction, recovering an arbitrary binary string, was originally studied in [Lev01a], [Lev01b], [BKKM04], [HMPW08]. The current state of the art was derived independently in [DOS17] and [NP17], who show that $\exp(O(n^{1/3}))$ traces suffice for any constant deletion probability $q \in (0,1)$. A very recent result [Cha20] shows that $\exp(O(n^{1/5}))$ traces suffice for any $q \in (0, 1/2]$. Several works have also considered lower bounds for worst-case trace reconstruction [BKKM04], [HMPW08], [MPV14a], [HL$^+$20], [Cha19]. The best known lower bound is $\Omega\left(\frac{n^{3/2}}{\log^{16} n}\right)$ traces [Cha19], which has an exponential gap compared to the best known upper bound. Our work does not use or address worst-case trace reconstruction.

In the average-case setting studied by [HMPW08], [MPV14a], [PZ17], [HPP18], the best upper bound is given by [HPP18], who showed that, for all deletion probabilities $q \in (0,1)$, a subpolynomial $\exp(O(\log^{1/3} n))$ traces suffice to recover a random string with high probability. Several works have also considered lower bounds for average-case trace reconstruction [MPV14a], [HL$^+$20], [Cha19]. The current best bound of $\Omega\left(\frac{\log n^{5/2}}{(\log\log n)^{16}}\right)$ traces [Cha19] again has an exponential gap. Our work shows that resolving the optimal number of traces up to a constant factor for coded trace reconstruction is essentially equivalent to average-case reconstruction.

Trace reconstruction over a larger alphabet is less well studied. [MPV14b], [DOS17] show that it is possible to turn any trace reconstruction algorithm over a non-binary alphabet into a trace over a binary alphabet and use binary trace reconstruction to solve the problem, at a small cost to the failure probability. For coded trace reconstruction, we show that there is a substantial benefit to using a non-binary alphabet. For constant-sized alphabets, we show a matching upper and lower bound, determining the optimal number of traces up to a constant factor.

Coded trace reconstruction.: Coded trace reconstruction generalizes the classical questions above about trace reconstruction. The worst-case trace reconstruction question over a binary alphabet asks how many traces $T(n)$ are needed to achieve error probability $o(1)$ for the code $C = \{0,1\}^n$. As we show in Section II-B, average-case trace reconstruction is equivalent to asking how many traces $T(n)$ are needed to achieve error probability $o(1)$ for a code C of size $2^n(1-o(1))$. We use this connection to average-case trace reconstruction to construct much longer codes which are recoverable from few traces.

Cheraghchi, Gabrys, Milenkovic, and Ribeiro [CGMR20] formulated the coded trace reconstruction problem considered here. Among other constructions, they give explicit constructions of binary codes of rate $1-O(\frac{1}{\log\log n})$ recoverable in $\exp(O(\log\log n)^{2/3})$ traces, and rate $1-O(\frac{1}{\log n})$ code recoverable in $\mathrm{poly}\log n$ traces. Our work improves the number of traces and allows a wider range of rates. For any $\varepsilon \geq n^{-o(1)}$, we show that there exist binary codes of rate $1-\varepsilon$ recoverable in $\exp(O_q(\log^{1/3}(\frac{1}{\varepsilon})))$ traces. Taking $\varepsilon = \Theta(\frac{1}{\log\log n})$ and $\varepsilon = \Theta(\frac{1}{\log n})$ gives the respective improvements to [CGMR20] in the number of traces. We emphasize that all the constructions of [CGMR20] have polynomial time encoding and decoding, whereas our constructions have polynomial time decoding in all considered parameter settings, but only polynomial time encoding when $\varepsilon \geq \Omega(\frac{\log\log n}{\log n})$.

Although our work deals with a constant fraction of deletions, several prior works considered coding for trace

[2]The rate of a code $|C|$ of length n over an alphabet Σ is $\frac{\log_{|\Sigma|}|C|}{n}$

reconstruction for small numbers of deletions. Haeupler and Mitzenmacher [HM14] showed that, for any fixed integer T, as the deletion probability q approaches 0, there exists a binary code recoverable from T traces across the BDC_q with rate $1-O(H(q^T))$, where H is the binary entropy function. By contrast, our codes handle deletion probabilities arbitrarily close to 1. We show, for example, that there exist binary codes of rate 0.99 recoverable from $T = O(1)$ traces of the $\text{BDC}_{0.99}$. Abroshan, Venkataramanan, Dolecek, and Guillén [AVDiF19] consider coding for channels applying a constant number of deletions. They concatenate ℓ Varshamov-Tenengolts [VT65] codes of length m to construct a code of length $m\ell$ and rate $1-O(\frac{\log m}{m})$ for any $m, \ell \geq 1$. They bound the error probability for recovering for a channel that applies exactly ℓ' deletions, when $\ell' < \ell$.

Other trace reconstruction variants.: There has recently been a variety of work on other problems related to trace reconstruction, which our work does not address. [GM19] considers the problem of recovering a string from the multiset of all its length L substrings. [BCF+19] studies population recovery under the deletion channel, an extension to trace reconstruction where we recover an unknown distribution over input strings, rather than a single input string. In [KMMP19], the authors consider the problems of reconstructing matrices and sparse strings from traces.

Codes for the deletion channel.: The optimal rate for coded trace reconstruction with one trace is also known as the *capacity* of the binary deletion channel, a well-studied and difficult problem. The capacity of the binary deletion channel with deletion probability q is clearly at most $1-q$, the capacity of the simpler binary erasure channel. When $q \to 0$, the capacity is known to approach $1-H(q)$, where $H(q)$ is the binary entropy function (see [DG01] for the lower bound and [KM13], [KMS10] for the upper bound). When $q \to 1$, the capacity is known to be $\Theta(1-q)$, but the exact capacity is known only to be roughly between $0.11(1-q)$ [DM06], [DM07], and $0.41(1-q)$ [RD15]. A polynomial time encodable/decodable code meeting this up to a constant factor was given in [GL19], [CS20]. The current best capacity upper bounds for intermediate q (e.g., $q = 0.5$) are given by [FD10], [RD15], [Che18]. We incorporate techniques used in constructing codes for the binary deletion channel in our construction of Theorem I.4. Our work shows that, at $q = 1-\delta$, if one is allowed to reconstruct from $O_\delta(1)$ traces of the BDC_q rather than only one trace, the capacity of the resulting channel improves from $\Theta(\delta)$ to 0.99.

B. Main results

We now define the coded trace reconstruction problem formally and state our main theorems. For $q \in (0,1)$ and $x \in \{0,1\}^n$, we let $\text{BDC}_q(x)$ denote the probability distribution of output of x across the BDC_q. We let $\{0,1\}^*$ denote the set of binary strings of any length.

Definition I.2. For $q, \delta \in (0,1)$ and positive integers n and T, we say a code $C \subset \{0,1\}^n$ is (T, q, δ) *trace reconstructible* if there exists a *decoding function* $\text{Dec} : (\{0,1\}^*)^T \to C$ such that, for all $c \in C$,

$$\mathop{\mathbf{Pr}}_{z_1,\ldots,z_T \sim \text{BDC}_q(c)}[\text{Dec}(z_1,\ldots,z_T) \neq c] < \delta.$$

Typically, we desire $\delta \to 0$ as $n \to \infty$. We say C is *decodable* in time t if Dec can be computed in time t. We say C is *encodable* in time t if there exists a bijection $\text{Enc} : \{1,\ldots,|C|\} \to C$ that can be evaluated in time t. The following notation, denoting the optimal number of traces for average-case trace reconstruction, is used throughout the paper.

Definition I.3. For $m \geq 1, q \in (0,1)$, and $\beta \geq 0$, let $T^{(\text{avg})}_{q,\beta}(m)$ denote the smallest integer T such that there exists a trace reconstruction algorithm for the BDC_q using T traces that, on a uniformly random string x of length m, succeeds with probability (over the randomness of the string and channel) at least $1 - \frac{1}{3m^\beta}$. When β is omitted, we take $\beta = 0$.

By repetition of the reconstruction algorithm and subsequently taking a majority vote, we have $T^{(\text{avg})}_q(m) \leq T^{(\text{avg})}_{q,\beta}(m) \leq O(\beta \log m) \cdot T^{(\text{avg})}_q(m)$, so $T^{(\text{avg})}_{q,\beta}(m)$ and $T^{(\text{avg})}_q(m)$ are roughly the same size for constant β.

Binary upper bound.: We prove the following upper bound for coded trace reconstruction, which allows bounds for average-case trace reconstruction to be turned into bounds for coded trace reconstruction.

Theorem I.4. *For all $q, \varepsilon \in (0,1)$, there exists constants $n_0 = 1/\varepsilon^{O_q(1)}$, $\beta = \Theta_q(1), n_R = \Theta_q(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$, and $\delta = 2^{-\varepsilon^{O_q(1)}n}$ such that, for all $n \geq n_0$, there exists a code $C \subset \{0,1\}^n$ of rate $1-\varepsilon$ that is $(T^{(\text{avg})}_{q,\beta}(n_R), q, \delta)$ trace reconstructible. Furthermore, the encoding can be done in time $\text{poly}_{\varepsilon,q}(n)$ and trace reconstruction can be done in time $\text{poly}(n)$.*

We can instantiate Theorem I.4 using the state-of-the-art construction for average-case trace reconstruction of Holden, Pemantle, and Peres [HPP18], which states that $T^{(\text{avg})}_q(\frac{1}{\varepsilon}) \leq \exp(O_q(\log^{1/3}\frac{1}{\varepsilon}))$. Doing so gives the following.

Corollary I.5. *For all $q, \varepsilon \in (0,1)$, there exists constants $n_0 = 1/\varepsilon^{O_q(1)}, T = \exp(O_q(\log^{1/3}(\frac{1}{\varepsilon})))$, and $\delta = 2^{-\varepsilon^{O_q(1)}n}$ such that, for all $n \geq n_0$, there exist codes of length n and rate at least $1-\varepsilon$ that are (T, q, δ) trace reconstructible.*

Remark I.6. In coding theory, we are sometimes interested in codes with rate quickly approaching 1, and our bounds on the number of traces hold in this setting as well. For every $q \in (0,1)$, Theorem I.4 and Corollary I.5 holds for all integers $n \geq \frac{1}{\varepsilon^{\Omega_q(1)}}$. Thus, we obtain obtain similar results

for ε going to 0 with n so long as $\varepsilon \geq \frac{1}{n^{O_q(1)}}$. Setting $\varepsilon = O(\frac{1}{\log n})$, we have codes of rate $1-O(\frac{1}{\log n})$ recoverable from $\exp(O_q(\log\log n)^{1/3})$ traces with failure probability $2^{-\tilde{O}_q(n)}$, improving upon the $\operatorname{poly}\log n$ number of traces in [CGMR20] needed for the same ε. Our construction also gives a better bound on the number of traces when $\varepsilon = O(\frac{1}{\log\log n})$, improving from $\exp(O_q(\log\log n)^{2/3})$ traces to $\exp(O_q(\log\log\log n)^{1/3})$ traces.

Remark I.7. While we improve on the number of traces in [CGMR20] and also give polynomial time decoding like in [CGMR20], their codes are all polynomial time encodable, whereas ours are only so when $\varepsilon \geq \Omega(\frac{\log\log n}{\log n})$: a careful look at our runtimes shows our code is encodable in time $t_{enc}(\Theta(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}))\cdot \operatorname{poly} n$, where $t_{enc}(n')$ is the amount of time needed to encode a string of length n' used for average-case trace reconstruction (see full version [BLS19] for more details). Naively we upper bound $t_{enc}(n') \leq 2^{O(n')}$. Thus, when $\varepsilon = O(\frac{1}{\log n})$, while we improve on the number of traces from [CGMR20] and also give polynomial time decoding, only [CGMR20] has codes with both encoding and reconstruction in polynomial time. Furthermore, the constants in our code are quite large, making them currently impractical. Still, we hope the ideas in our construction could be used for future efficient constructions.

Binary lower bound.: We also prove the following converse, showing that the number of traces needed for rate $1-\varepsilon$ trace reconstruction is at least the number of traces needed for average-case trace reconstruction on length $\frac{1}{\varepsilon^{1/2-o(1)}}$ strings with failure probability 1/3.

Theorem I.8. *For all $q, \delta \in (0,1)$, for sufficiently small $\varepsilon > 0$, there exists $m = \tilde{\Omega}_q(\frac{1}{\varepsilon^{1/2}})$ such that, if $T = T_q^{(\text{avg})}(m)$, all rate $1-\varepsilon$ codes of sufficiently large length are* not *$(T-1, q, \delta)$-trace reconstructible.*

Using Theorem I.8, we can adapt the state-of-the-art lower bound for average case trace reconstruction into a lower bound for coded trace reconstruction. Recently Chase [Cha19], building off work of Holden and Lyons [HL+20], showed that $T_q^{(\text{avg})}(m) \geq \tilde{\Omega}_q((\log m)^{5/2})$.[3] Applying Theorem I.8 to this result gives us the following lower bound.

Corollary I.9. *For all $q, \delta \in (0,1)$ and $\varepsilon > 0$ sufficiently small, there exists $T = \tilde{\Omega}_q((\log\frac{1}{\varepsilon})^{5/2})$ such that all rate $1-\varepsilon$ codes of sufficiently large length are* not *(T, q, δ)-trace reconstructible.*

Remark I.10. Theorem I.8 holds when $n \geq \tilde{\Omega}_q(\frac{1}{\varepsilon^2})$. Hence, similar to Remark I.6, the lower bound of Theorem I.8 holds for ε approaching 0 with n, so long as $\varepsilon \geq \Omega_q(\frac{1}{n^{1/2}})$.

[3]Here, $\tilde{\Omega}(\cdot)$ suppresses $\log\log$ factors. In fact, they show something stronger: even achieving success probability $\exp(m^{-0.15})$ requires that many traces.

Remark I.11. Theorem I.4 and Theorem I.8 together show that the optimal number of traces for a code of rate $1-\varepsilon$ is bounded above and below by the number of traces for average-case trace reconstruction of a string of length $\operatorname{poly}(1/\varepsilon)$. More precisely, there exist $m_1 = \tilde{\Omega}_q(\frac{1}{\sqrt{\varepsilon}})$ and $m_2 = \tilde{O}_q(\frac{1}{\varepsilon})$ such that the optimal number of traces for rate $1-\varepsilon$ coded trace reconstruction with failure probability $\frac{1}{3}$ is between $T_q^{(\text{avg})}(m_1)$ and $O_q(\log\frac{1}{\varepsilon})\cdot T_q^{(\text{avg})}(m_2)$. Hence any qualitative improvement to the upper or lower bounds for coded trace reconstruction implies an analogous improvement for average-case trace reconstruction and vice versa.

Large alphabet upper and lower bounds.: So far, we have focused on codes for binary alphabets. By defining the deletion channel for strings over larger alphabets in the same way as the binary deletion channel, one can ask questions for coded trace reconstruction over larger alphabets. In this setting, our results are stronger in two ways. Firstly, we are able to show matching upper and lower bounds for large alphabet trace reconstruction. Secondly, these constructions are simpler and do not rely on average-case trace reconstruction results.

Theorem I.12. *For all $q, \varepsilon \in (0,1)$ and infinitely many n, there exists a rate $1-\varepsilon$ code over an alphabet of size $2^{O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})}$ that is (T, q, δ) trace reconstructible for $T = O(\log_{1/q}\frac{1}{\varepsilon})$ and $\delta = 2^{-\Omega(n)}$ and which is encodable in time $O(n)$ and decodable in time $O(nT)$.*

And as the following lower bound shows, this is tight in terms of the number of traces.

Theorem I.13. *Any code (over any alphabet) of rate $1-\varepsilon$ is not $(\lfloor\log_{1/q}\frac{1}{\varepsilon}\rfloor, q, o(1))$ trace reconstructible.*

We do not know if the dependence on ε for the alphabet size in Theorem I.12 is optimal. We leave understanding the trade-off between alphabet size and number of traces as an open question for future work.

C. Techniques

In this section we describe our constructions. We first combine synchronization strings [HS17] and erasure codes [GI05] to give our large alphabet construction (Theorem I.12), and match this construction with a simple lower bound (Theorem I.13).

Extending these ideas to our binary code construction (Theorem I.4) requires more work, and we introduce a novel technique for binary code concatenation, turning our large alphabet code from Theorem I.12 into a binary code. This concatenation also leverages codes for the binary deletion channel (e.g. [GL19]), and bounds for average-case trace reconstruction [HPP18].

We finish this section by describing our lower bound for coded trace reconstruction for the binary alphabet (Theorem I.8). Trace reconstruction lower bounds usually find a

hard pair of strings and prove that it takes many traces to distinguish these strings. Coded trace reconstruction can simply avoid these hard pairs of strings, which makes applying prior results difficult. Using techniques from information theory, we are able to transfer average-case trace reconstruction lower bounds to the coded setting.

Large alphabet construction and lower bound.: As a warm-up, first observe that any binary code $C \subset \{0,1\}^n$ can be turned into a code C' over an alphabet of size $2n$ by mapping each codeword $(r_1, \ldots, r_n)$ to a codeword $((r_1,1),(r_2,2),\ldots,(r_n,n)) \in (\{0,1\}\times[n])^n$. This code has very low rate, but has the useful property that the deletion channel is essentially turned into an erasure channel: from a received string, we can always recover the indices of the received symbols, and thus the corresponding r_i. If C is a code of rate $1-\varepsilon$ tolerating a $\delta = \text{poly}(\varepsilon)$ fraction of erasures, C' is recoverable from $O(\log_{1/q} \frac{1}{\varepsilon})$ traces: with high probability at most $q^T < \delta$ fraction of symbols are never received, producing less than δn erasures, which can be corrected.

Our construction for large alphabets (Theorem I.12) uses the above intuition, but relies on synchronization strings to avoid ruining the rate of the resulting code. Instead of specifying the exact position of each symbol, we include a symbol of a synchronization string [HS17] from a much smaller alphabet of size $\text{poly}\left(\frac{1}{\varepsilon}\right)$. We take our starting code C to be over a large alphabet of size $2^{O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})}$ and tolerate a $\delta = \text{poly}(\varepsilon)$ fraction of erasures [GI05]. Increasing the size of the alphabet beyond that of [GI05] helps ensure the correct rate when combining with the synchronization string. At the cost of a few more erasures, we can convert the outputs on the deletion channel into outputs with erasures and correct the erasures.

For the lower bound (Theorem I.13), any code of rate $1-\varepsilon$ recovering from T traces must also be able to recover from the erasure channel with erasure probability q^T, which has capacity at most $1-q^T$. Therefore, $1-\varepsilon < 1-q^T$ so $\log_{1/q}\frac{1}{\varepsilon}$ traces are necessary for the erasure channel, and thus the deletion channel.

Binary alphabet construction.: Our construction for binary alphabets (Theorem I.4) uses additional ideas beyond those in the large alphabet construction. Again, we use a high rate error correcting code with codewords $(r_1,\ldots,r_{n_{out}}) \in C$ and a synchronization string $(s_1,\ldots,s_{n_{out}})$. Naively, one might "concatenate" the large alphabet construction with a high rate code of length $n_{in} = O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ recoverable from a $O_\varepsilon(1)$ number of traces (which exists by [HPP18]), so that each pair (r_i, s_i) is encoded in a binary string a_i of length n_{in}, and the final codeword is the concatenation $a_1||\cdots||a_{n_{out}}$. Then, to recover the message, we first use the T traces of the codeword $a_1||\cdots||a_{n_{out}}$ to recover T traces of each a_i. As in [CGMR20], we can make sure we know where the traces of the a_i start and finish by adding buffers of long runs on the ends of each a_i. From the traces of each a_i, we run the inner trace reconstruction to recover each a_i, and thus recover the pair (r_i, s_i). We then run the outer error correction to fix any incorrectly decoded r_i's.

This construction does not work for a subtle reason. Because the length of each a_i is a constant, we expect a (very small) constant fraction of the a_i's buffers to be deleted, and we also expect a (very small) constant fraction of a_i's to have deletions applied so that the interior of the a_i looks like a buffer (we call this a "spurious" buffer). From the T traces of the codeword, we try to recover T traces of each of the a_i's using the buffers, but these T traces, supposedly of a_i, might contain some traces of, e.g., a_{i-5} or a_{i+3}. Therefore, we need to know the synchronization symbols s_i to determine which substrings of each of the T traces belong to which a_i. Thus, recovering the synchronization symbols must happen *before* running trace reconstruction on the a_i's. However, the synchronization symbols s_i are encoded in the a_i, so in this construction the synchronization symbols cannot be recovered until *after* the trace reconstruction.

To avoid this issue, our construction crucially encodes the content symbol r_i and the synchronization symbol s_i separately. To our knowledge, this kind of concatenation has not appeared in other constructions of deletion codes. Each content symbol r_i is encoded using a high rate code of length $n_R = \Theta(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ obtained from bounds on average-case trace reconstruction. Each synchronization symbol is encoded in a code of length $n_S = \Theta(\log\frac{1}{\varepsilon})$ decodable in, crucially, 1 trace from the binary deletion channel. We can afford a very low rate code for the synchronization symbols because they are over a much smaller alphabet than the content symbols. Furthermore, we structure the encoded content symbols and encoded synchronization symbols so that they are not easily confused with each other.

For the final decoding algorithm, we first recover the synchronization symbols within each trace. We then use the synchronization strings to determine the parts of each trace that corresponding to traces of a particular a_i. We then use these traces of a_i in trace reconstruction to recover each content symbol r_i. Finally, we use the error correction of the outer code C to fix any mistakes in this process.

Binary alphabet lower bound.: Our binary lower bound (Theorem I.8) reduces coded trace reconstruction to constructing a code over an appropriately chosen memoryless channel, i.e. a channel where each alphabet symbol is corrupted independently or the other symbols. In particular, we partition the input string $x \in \{0,1\}^n$ into n/m substrings of length $m \approx 1/\sqrt{\varepsilon}$. We then upper bound the rate of a code $C \subset (\{0,1\}^m)^{n/m}$ over alphabet $\{0,1\}^m$ recovering a sequence x of length m substrings from $T = T_q^{(\text{avg})}(m)$ independent traces of each of the n/m substrings. This is easier than recovering x from T independent traces of itself, so any rate upper bound for the code for n/m substrings yields a rate upper bound for the original coded trace reconstruction problem.

Now, we can view the problem as coding over a discrete memoryless channel: we view our binary code as a code of length n/m over the input alphabet $\mathcal{X} = \{0,1\}^m$ and the channel produces outputs in $\mathcal{Y} = (\{0,1\}^*)^T$, corresponding to T independent traces of the elements of $\mathcal{X}$. By Shannon's noisy channel coding theorem [Sha48], the capacity of this channel equals the maximum, over distributions λ on $\mathcal{X}$, of the mutual information $I(X_\lambda, Y_\lambda)$, where $X_\lambda \in \mathcal{X}$ is sampled from λ and $Y_\lambda \in \mathcal{Y}$ is a tuple of T strings each sampled as an independent trace of X_λ. Thus, to upper bound the rate of C, it suffices to upper bound the mutual information $I(X_\lambda, Y_\lambda)$ for all distributions λ on $\mathcal{X}$. If the distribution λ is "far" from the uniform distribution, we can upper bound the mutual information by the entropy of $X_\lambda \sim \lambda$. Otherwise, if λ is "close" to the uniform distribution, the mutual information is limited by the performance of average-case trace reconstruction. In either case, we get an upper bound on the mutual information which implies an upper bound on the rate of a code correctable from T traces.

D. Paper organization

In Section II, we define a few building blocks for our work. These include synchronization strings, codes for the binary deletion channel, and high rate error correcting codes. In Section III, we present the proofs of our coded trace reconstruction results over large alphabets in Theorems I.12 and Theorem I.13. These proofs are simpler and serve as warm-ups for our results over binary alphabets, which require additional ideas. In Section IV, we prove Theorem I.8, giving a black-block reduction from lower bounds for average-case trace reconstruction to lower bounds for coded trace reconstruction.

The proof of Theorem I.4 is deferred to the full version [BLS19].

II. Preliminaries

A. Basics

All logs and exps are base 2 unless otherwise specified. For an alphabet Σ, we let Σ^* denote the set of strings over Σ of any length. For strings w, w', we let ww' denote the concatenation of strings w and w'. We may also denote the concatenation by $w||w'$ for clarity. For a string w and integer i, let w^i denote the string $ww\cdots w$ with w repeated i times. A *substring* is a sequence of consecutive characters in a string. A *run* is a maximal substring of a string all of whose bits are the same. A *partial function* $f : A \nrightarrow B$ is a function from a subset of A to B. For $x \in (0,1)$, let $H(x) = -x\log x - (1-x)\log(1-x)$ denote the binary entropy function.

A *code* C of length n over an alphabet Σ is a subset of Σ^n. The elements of C are called *codewords*, and n is called the *length* of the code. If $|\Sigma| = 2$, we say C is a binary code. The *rate* of a code C is defined to be $\frac{\log|C|}{n\log|\Sigma|}$. A code may have an associated *message set* $\mathcal{M}$ and *encoding function* $\mathrm{Enc} : \mathcal{M} \to C$, which is an injective map from messages to codewords. By default, $\mathcal{M} = \{1, \ldots, |C|\}$. A code is *decodable under the* BDC_q *with failure probability* δ if it is $(1, q, \delta)$ trace reconstructible. To *construct* a code means to produce a description of its encoding and decoding functions. Given two codes $C_1 \subset \Sigma_1^{n_1}$ and $C_2 \subset \Sigma_2^{n_2}$ with $|\Sigma_1| \le |C_2|$, a *concatenation* of C_1 and C_2 is a code $C \subset \Sigma_2^{n_1 n_2}$ whose codewords are $\mathrm{Enc}_2(c_1)||\ldots||\mathrm{Enc}_2(c_{n_1})$ where $c_1\cdots c_{n_1} \in C_1$, and where $\mathrm{Enc}_2 : \Sigma_1 \to C_2$ is a fixed injective map.

We use the following forms of the Chernoff bound (e.g., [DP09])

Lemma II.1 (Chernoff bound – discrete)**.** *Let $X_1, \ldots, X_n$ be independent and identically distributed random variables with mean μ supported on $\{0,1\}$ Then, for $\delta \ge 0$,*

$$\mathbf{Pr}[X_1 + \cdots + X_n \le (1-\delta)\cdot n\mu] \le e^{-\frac{\delta^2}{2}\cdot n\mu}$$
$$\mathbf{Pr}[X_1 + \cdots + X_n \ge (1+\delta)\cdot n\mu] \le e^{-\frac{\delta^2}{2+\delta}\cdot n\mu}.$$

Lemma II.2 (Chernoff bound – continuous)**.** *Let $X_1, \ldots, X_n$ be independent and identically distributed random variables with mean μ supported on $[0,1]$ Then, for $\delta \ge 0$,*

$$\mathbf{Pr}[X_1 + \cdots + X_n \ge (1+\delta)\cdot n\mu] \le e^{-2\delta^2\cdot\mu^2 n}.$$

B. Short codes from average-case trace reconstruction

In this section, we show a connection between short codes for trace reconstruction and average-case trace reconstruction. We use this connection to construct short, high-rate, trace reconstructible codes, which are building blocks in our main result.

The current state of the art for the optimal number of traces for average-case trace reconstruction is due to Holden, Pemantle, and Peres [HPP18], who show the following bound on $T^{(\mathrm{avg})}_{q,\beta}(n)$.

Theorem II.3 ([HPP18])**.** *For all $q \in (0,1)$ and $\beta \ge 1$, we have $T^{(\mathrm{avg})}_{q,\beta}(n) \le \exp(O_{q,\beta}(\log^{1/3} n))$.*

Note that the paper [HPP18] only states Theorem II.3 for failure probability $1/n$, but their proof works in the same way for any polynomial failure probability $1/n^\beta$. There is also a slick way to amplify the failure probability in average-case trace reconstruction: with polynomially more traces, we can turn failure probability $1/n$ into $1/n^\beta$, by appending random bits to each trace and running trace reconstruction for $n' = n^\beta$ (see e.g., Theorem 3.2 of [BCSS19]).

We now have the following two simple observations that results for average-case trace reconstruction show the existence of codes for coded trace reconstruction and vice versa.

Claim II.4. *If there exists a code of size $2^n(1-o(1))$ that is $(T, q, o(1))$ trace reconstructible, then average case*

trace reconstruction can be done in T traces with failure probability $o(1)$.

And conversely,

Lemma II.5. *Let $\beta > 1$, $q \in (0,1)$, and $T = T^{(\text{avg})}_{q,2\beta}(n)$. For all positive integers n, there exists a code C with $|C| \geq (1-n^{-\beta})2^n$ that is $(T, q, n^{-\beta})$ trace reconstructible.*

We need to combine these short trace reconstruction codes into a longer one in Theorem I.4. The following notion helps prevent these short codes from being confused with the other components of our construction.

Definition II.6. A string w is *m-protected* if it can be written as $w = 0^m w^\circ 1^m$, where w° starts with a 1, ends with a 0, and every substring of w° of length $m' \geq m/4$ has between $\frac{m'}{4}$ and $\frac{3m'}{4}$ 1s (inclusive). In any m-protected string w, we let w° denote the string w with the leading m 0s and the trailing m 1s deleted. We refer to w° as the *interior* of w. A code is m-protected if all of its codewords are m-protected.

We use short codes which are both m-protected and trace reconstructible in our construction. The full version [BLS19] shows that such codes exist.

C. Synchronization strings

Synchronization strings [HS17] are useful tools for turning synchronization errors (insertions and deletions) into erasures (replacing symbol with the symbol '?') and substitution errors (replacing symbol with another symbol). Here, we state the construction of synchronization strings that we use and a few useful properties.

Definition II.7 (Insertion-deletion distance)**.** Given two strings $S \in \Sigma^n$ and $T \in \Sigma^m$, the *insertion-deletion distance* between S and T, denoted $\mathrm{ID}(S,T)$ is the minimum number of characters that needed to be inserted into S and deleted from S to produce T.

Insertion-deletion distance is similar to *edit distance* which allows for substitutions at a cost of 1. Observe that if S and T have disjoint character sets, then $\mathrm{ID}(S,T)$ is the sum of their lengths.

Definition II.8 (η-synchronization string)**.** String $S \in \Sigma^n$ is an η-synchronization string if for every $1 \leq i < j < k \leq n+1$, we have that $\mathrm{ID}(S[i,j), S[j,k)) > (1-\eta)(k-i)$.

Theorem II.9 (Theorems 4.5 and 4.7 of [HS18])**.** *For any $\eta \in (0,1)$ and all n, one can construct an η-synchronization string of length n in time $\mathrm{poly}(n)$ over an alphabet of size $6000\eta^{-4}$.*

We now describe some useful properties of synchronization strings. Informally, a *string matching* between two strings describes how to transform one string into the other via insertions and deletions. We use a definition of string matching equivalent to the one introduced in [HS17].

Definition II.10 (String matching)**.** For strings c and c' of length n and n', respectively, a *string matching* is a strictly increasing partial function $i^* : [n'] \nrightarrow [n]$ such that, for all j in the domain of i^*, we have $c_{i^*(j)} = c'_j$. Given a string matching, an index $j \in [n']$ is called *successfully transmitted* if it is in the domain of i^*, and is called an *insertion* otherwise. An element $i \in [n]$ is called a *deletion* if it is not in the codomain of i^*.

A *(n,δ)-indexing algorithm for a string S* takes as input a string S' of length n' with an unknown string matching $i^* : [n'] \nrightarrow [n]$ having at most $n\delta$ insertions and deletions and outputs an index in $[n] \cup \{\perp\}$ for every index in $[n']$. We say the algorithm *decodes index $j \in [n']$ correctly* under a string matching i^* if it outputs $i^*(j)$ for index j when $i^*(j)$ exists and outputs $\perp$ if it does not exist. A *misdecoding* of an algorithm is a successfully transmitted, incorrectly decoded index $j \in [n']$. An indexing algorithm is *error free* if every $j \in [n']$ is correctly decoded or is assigned $\perp$.

Haeupler and Shahrasbi proved many results showing that synchronization strings yield indexing algorithms with few misdecodings. In this work, we use the following two results.

Theorem II.11 (Theorem 5.10 of [HS17])**.** *Let S be an η-synchronization string of length n. Then there exists an (n,δ)-indexing algorithm for S guaranteeing at most $\frac{2n\delta}{1-\eta}$ misdecodings. Furthermore, this algorithm runs in time $O(n^4)$*

Theorem II.12 (Theorem 6.18 of [HS17])**.** *Let S be an η-synchronization string of length n. There exists a linear time error-free deletion-only (n,δ)-indexing algorithm for S guaranteeing at most $\frac{\eta}{1-\eta} \cdot n\delta$ misdecodings.*

D. High rate error correcting codes

Our constructions leverage high rate (rate $1-\varepsilon$) error correcting codes that are polynomial time encodable and decodable from a $\mathrm{poly}(\varepsilon)$ fraction of worst-case substitution errors. The following error correcting code of Guruswami and Indyk [GI05] allows linear time encoding/decoding of our large alphabet construction.

Proposition II.13. *For every $\varepsilon \in (0, \frac{1}{2})$ and Σ whose size is a power of 2, there exist an infinite family of codes over Σ of rate $1-\varepsilon$ encodable in linear time and decodable in linear time from up to a fraction $\frac{1}{40}\varepsilon^3$ of worst-case substitution errors.*

III. Optimal number of traces for large alphabet codes

We begin by describing the upper and lower bounds for coded trace reconstruction over a large alphabet. Many of the tools used in this section are important building blocks

for the analysis of coded trace reconstruction over a binary alphabet.

A. Upper bound

Proof of Theorem I.12: We start by defining a few parameters for our construction.

Parameters. Let $T = \lceil \log_{1/q} \frac{160}{\varepsilon^3} \rceil$. Let $q' = \frac{1+q}{2}$ and $\eta = \frac{\varepsilon^3}{160T}$. Let Σ_S be an alphabet such that there exist η-synchronization strings over Σ_S, and assume $|\Sigma_S|$ is a power of 2. We may take $|\Sigma_S| = O_q(\mathrm{poly}\, \frac{1}{\varepsilon})$ by Theorem II.9.

Code. Let C_1 be a length n erasure code over an alphabet Σ_C of size $|\Sigma_S|^{\lceil 2/\varepsilon \rceil}$, rate at least $1 - \frac{\varepsilon}{2}$, and decodable from a $\frac{\varepsilon^3}{40}$ fraction of worst-case substitution errors, given by Proposition II.13. Let $s_1, s_2, \ldots, s_n$ be an η-synchronization string over alphabet Σ_S. Let $\Sigma = \Sigma_C \times \Sigma_S$. Let C be a code with encoding $\mathcal{M} \to \Sigma^n$ whose codewords are $(c_1, s_1), \ldots, (c_n, s_n)$ for codewords $(c_1, \ldots, c_n) \in C$.

Decoding algorithm. For $t \in [T]$, let $z^{(t)} = (x_1^{(t)}, y_1^{(t)}), \ldots, (x_{n^{(t)}}^{(t)}, y_{n^{(t)}}^{(t)})$ be the tth trace, which has length $n^{(t)}$. Call a trace $z^{(t)}$ for $t \in [T]$ *useful* if $n^{(t)} \geq (1 - q') \cdot n$.

1) For every useful trace $z^{(t)}$, run the error-free deletion-only (n, q')-indexing algorithm in Theorem II.12 to obtain indices $i_1^{(t)}, \ldots, i_{n^{(t)}}^{(t)} \in [n] \cup \{\perp\}$.
2) For $i = 1, \ldots, n$, if there exists a useful $t \in [T]$ and index $j \in [n^{(t)}]$ such that $i_j^{(t)} = i$, then let $\hat{c}_i = x_j^{(t)}$. Otherwise, let $\hat{c}_i = \perp$.
3) Run the erasure decoding for C_1 on the string $(\hat{c}_1, \ldots, \hat{c}_n)$ to obtain a message in $\mathcal{M}$.

Efficiency. The code C_1 and synchronization string can each be constructed in polynomial time. Since C_1 has linear time encoding, so does our code. Decoding takes time $O(n \log \frac{1}{\varepsilon})$: the indexing algorithm for synchronization strings takes linear time by Theorem II.9 and we run it T times, and decoding the code C_1 from the resulting erasures takes linear time by Proposition II.13.

Rate. The rate of the code C_1 is at least $1 - \frac{\varepsilon}{2}$, so there are $|\Sigma_C|^{n(1-\frac{\varepsilon}{2})} = |\Sigma|^{n(1-\frac{\varepsilon}{2}) \cdot \frac{\log |\Sigma_C|}{\log |\Sigma|}} \geq |\Sigma|^{n(1-\varepsilon)}$ codewords. The inequality follows as $\frac{\log |\Sigma_C|}{\log |\Sigma|} > 1 - \frac{\varepsilon}{2}$ Hence, the rate of C is at least $1 - \varepsilon$.

Analysis. First, the probability that some trace is not useful is equal to the probability that a binomial $B(n, 1-q)$ is at most $(1-q')n = \frac{1-q}{2}n$, which, by the Chernoff bound, is at most $e^{-(1-q)n/8}$. Thus, the probability that there exists a trace that is not useful is, by the union bound, at most $T \cdot e^{-(1-q)n/8} \leq 2^{-\Omega(n)}$.

For all useful $t \in [T]$, $z^{(t)}$ is obtained from applying at most $q'n$ deletions to c. Thus, the (n, q') indexing-algorithm in Theorem II.12 succeeds with at most $\frac{\eta}{1-\eta} \cdot nq' < 2\eta n$ misdecodings. Hence, for all $j \in [n^{(t)}]$, we either have $i_j^{(t)} = \perp$ or j is correctly decoded, in which case $x_j^{(t)} = c_{i_j}$. We conclude that, for all $i = 1, \ldots, n$, we either have $\hat{c}_i = c_i$ or $\hat{c}_i = \perp$. We now simply need to lower bound the number of $\hat{c}_i$ that are not $\perp$. If every trace is useful, for each index i with $\hat{c}_i = \perp$, either (c_i, s_i) is deleted in every trace or some trace has a misdecoding at the image of (c_i, s_i). The expected number of symbols (c_i, s_i) deleted in every trace is $q^T n$, so by the Chernoff bound II.1, the probability that there are more than $2q^T n$ symbols deleted in every trace is $2^{-\Omega_q(n)}$. Across all traces, the total number of misdecodings is at most $T \cdot 2\eta n$ by above. Thus, with probability at least $1 - 2^{-\Omega_q(n)}$, there are at most $2q^T n + 2T\eta n < \frac{\varepsilon^3}{40} n$ indices i with $\hat{c}_i = \perp$. Hence, as the code C_1 tolerates $\frac{\varepsilon^3}{40} \cdot n$ errors (and thus erasures), we decode our message correctly. ∎

B. Lower bound

Proof of Theorem I.13: For brevity, let DC_q denote the deletion channel with deletion probability q. Let EC_q denote the erasure channel with erasure probability q. That is EC_q takes an input string and independently with probability q replaces each symbol with the symbol '?'.

We show that a $(T, q, o(1))$ trace reconstructible code over the DC_q is a code for EC_{q^T} with block error probability $o(1)$. To do this, we show that we can turn an output of EC_{q^T} into T independent outputs of DC_q. From a single symbol sent over EC_{q^T}, one can produce T independent copies of the symbol sent across EC_q: if the output is an erasure, return T erasures, and if the output is the original symbol, return the output of T independent copies of the symbol over EC_q, conditioned on not all outputs being erasures. Using the above, from a single output from EC_{q^T}, one symbol at a time, produce T independent outputs over EC_q, and replace the erasures with deletions to obtain T independent outputs over DC_q, as desired. Since the capacity of EC_{q^T} is $1 - q^T$ (see e.g. [Sha48]), we have that our code cannot be $(T, q, o(1))$ trace reconstructible when $1 - \varepsilon > 1 - q^T$, i.e. $T < \log_{1/q} \frac{1}{\varepsilon}$. ∎

IV. Lower bound on traces for binary codes

In this section, we prove the following theorem, which implies Theorem I.8.

Theorem IV.1. *Let $q \in (0, 1)$ and $\varepsilon < \frac{1}{4}$. Let $m = \lfloor \sqrt{\frac{1/\varepsilon}{128 \log(1/\varepsilon)}} \rfloor$ and $T = T_{q,0}^{(\mathrm{avg})}(m) - 1$. Then, for all $\delta \in (0, 1)$, there exists $n_0 = O_\delta(1/\varepsilon^2)$ such that all rate $1 - \varepsilon$ codes of length at least n_0 are not (T, q, δ)-trace reconstructible.*

A. Mutual information and Shannon's theorem

Recall that the entropy of a random variable X is $H(X) \stackrel{\mathrm{def}}{=} -\sum_x \mathbf{Pr}[X = x] \log \mathbf{Pr}[X = x]$. For two random variables X and Y their conditional entropy of Y given X is defined to be $H(X|Y) \stackrel{\mathrm{def}}{=} \sum_y \mathbf{Pr}[Y = y] \cdot H(X|Y = y)$, where $H(X|Y = y)$ is the entropy of the random variable X given that $Y = y$. From this, we can define their mutual information $I(X, Y)$ to be

$I(X,Y) \stackrel{\text{def}}{=} H(X) - H(X|Y)$. A *discrete memoryless channel* has finite input alphabet $\mathcal{X}$ and finite output alphabet $\mathcal{Y}$, and is given by a matrix $w(y|x)$, denoting, for each $x \in \mathcal{X}$, a distribution over received symbols $y \in \mathcal{Y}$. With w, any probability distribution over $\mathcal{X}$ gives a joint distribution on $\mathcal{X}, \mathcal{Y}$.

Given a discrete memoryless channel w, we say a code $C \subset \mathcal{X}^n$ is *decodable with failure probability at most* δ if there exists a map $f : \mathcal{Y}^n \to \mathcal{X}^n$ such that, for all $x_1 \cdots x_n \in C$, we have

$$\mathop{\mathbf{Pr}}_{y_i \sim w(\cdot|x_i)}[f(y_1, \ldots, y_n) \neq x_1 \cdots x_n] \leq \delta.$$

We need the following result, which provides a strong converse to Shannon's noisy channel coding theorem [Sha48].

Theorem IV.2 (e.g. Theorem 3.3.1 of [Wol78])**.** *Let $w(\cdot|\cdot)$ define a discrete memoryless channel with inputs $\mathcal{X}$ and outputs $\mathcal{Y}$. Let*

$$R_{cap} \stackrel{\text{def}}{=} \max_{p(x)} I(X,Y), \tag{1}$$

where the maximum is taken over probability distributions on $\mathcal{X}$, and let $\gamma > 0$. Then, for all $\delta \in (0,1)$, there exists $n_0 = O_\delta(\frac{1}{\gamma^2})$ such that, for all $n \geq n_0$ there do not exist codes of rate $\frac{R_{cap}+\gamma}{\log|\mathcal{X}|}$ decodable with failure probability at most δ under the channel $w(\cdot|\cdot)$.[4][5]

A classic result known as Fano's inequality can be used to lower bound the mutual information $I(X,Y)$ in (1) with a quantity involving the probability of error. The following result by Tebbe and Dwyer [TI68] helps bound the mutual information $I(X,Y)$ in the other direction, and is useful in our proof.

Lemma IV.3 ([TI68])**.** *Let $\delta \in (0,1)$. Suppose we are given a probability distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ such that, for all maps $f : \mathcal{Y} \to \mathcal{X}$, we have $\mathbf{Pr}_{X,Y}[f(Y) \neq X] \geq \delta$. Then $H(X|Y) \geq \frac{\delta}{2}$.*

B. Random to coded lower bound

Let $\mathcal{X}_m \stackrel{\text{def}}{=} \{0,1\}^m$ and $\mathcal{Y}_{m,T} \stackrel{\text{def}}{=} (\{0,1\}^{\leq m})^T$. For all q, m and T, there is a natural channel with inputs $\mathcal{X}_m$ and outputs $\mathcal{Y}_{m,T}$. We induce a joint probability distribution on $\mathcal{X}_m, \mathcal{Y}_{m,T}$ as follows. Let λ be a probability density function on $\mathcal{X}_m$. Let $X_\lambda \sim \mathcal{X}_m$ be the distribution where x is sampled with probability $\lambda(x)$. We let Y_λ be the output of T independent traces of the sampled $x \sim X_\lambda$ across the BDC_q.

Note that since $H(X_\lambda) \leq m$, for any distribution $X_\lambda \sim \mathcal{X}_m$, we have that $I(X_\lambda, Y_\lambda) \leq m$. We show in Lemma IV.4 that if $T \approx T^{(\text{avg})}_{q,0}(m)$, then this upper bound can be improved by a significant amount. This upper bound is subsequently used in Theorem IV.1 to show a limitation of the capacity of coded trace reconstruction.

Lemma IV.4. *Let $\beta \geq 1$. Suppose $T = T^{(\text{avg})}_{q,0}(m) - 1$ for $m \geq 32$. For all probability distributions X_λ on $\mathcal{X}_m$, if $Y_\lambda \in \mathcal{Y}_{m,T}$ is distributed as T independent traces of X_λ, then*

$$I(X_\lambda, Y_\lambda) \leq m - \frac{1}{32m \log m}.$$

Proof: Let $\mathcal{X}'$ be the elements of $\mathcal{X}$ with $\lambda(x) \geq \frac{1}{(m \log m)2^m}$. We consider two cases.

Case 1: $|\mathcal{X}'| \leq 2^{m-1/3}$**.** We have

$$\begin{aligned}
&I(X_\lambda, Y_\lambda) \\
&\leq H(X_\lambda) \\
&= \sum_{x \in \mathcal{X}'} \lambda(x) \log \frac{1}{\lambda(x)} + \sum_{x \notin \mathcal{X}'} \lambda(x) \log \frac{1}{\lambda(x)} \\
&\leq \log|\mathcal{X}'| + \sum_{x \notin \mathcal{X}'} \lambda(x) \log \frac{1}{\lambda(x)} && (2) \\
&\leq \log|\mathcal{X}'| + \sum_{x \notin \mathcal{X}'} \frac{1}{(m \log m)2^m} \cdot \log((m \log m)2^m) && (3) \\
&\leq \log|\mathcal{X}'| + 2^m \cdot \frac{1}{m(\log m)2^m} \log(m(\log m)2^m) \\
&= m - \frac{1}{3} + \frac{m + \log m + \log\log m}{m \log m} \\
&< m - \frac{1}{3} + \frac{1}{4} < m - \frac{1}{32m \log m}. && (4)
\end{aligned}$$

In (2) we used that $\sum_{x \in \mathcal{X}} \lambda(x) \leq 1$ and that $z \log \frac{1}{z}$ is concave. In (3) we used that $z \log \frac{1}{z}$ is increasing for $z < 1/3$. In (4) we used that m is sufficiently large.

Case 2: $|\mathcal{X}'| \geq 2^{m-1/3}$**.**

For this case, a similar argument appears in [HL$^+$20] (Proposition 4.1). Let $\sigma(x)$ be the uniform distribution on the elements of $\mathcal{X}$. Let $\mu(x)$ be the uniform distribution on the elements of $\mathcal{X}'$. Consider any trace reconstruction algorithm $f : \mathcal{Y}_{m,T} \to \mathcal{X}_m$. Note that

$$\mathbf{Pr}[f(Y_\sigma) \neq X_\sigma] \leq \frac{|\mathcal{X} \setminus \mathcal{X}'|}{|\mathcal{X}|} + \frac{|\mathcal{X}'|}{|\mathcal{X}|} \mathbf{Pr}[f(Y_\mu) \neq X_\mu].$$

By definition, $T \stackrel{\text{def}}{=} T^{(\text{avg})}_{q,0}(m) - 1$ and $|\mathcal{X}'| \geq 2^{-1/3}|\mathcal{X}|$, so

$$\mathbf{Pr}[f(Y_\mu) \neq X_\mu] \geq 2^{-1/3} \mathbf{Pr}[f(Y_\sigma) \neq X_\sigma] - (1 - 2^{-1/3})$$

$$\geq \frac{2^{-1/3}}{3} - 1 + 2^{-1/3} > \frac{1}{8}.$$

Let $\nu(x)$ be the probability distribution on $\mathcal{X}$ given by

$$\nu(x) \stackrel{\text{def}}{=} \frac{\lambda(x) - \frac{1}{2m \log m}\mu(x)}{1 - \frac{1}{2m \log m}}.$$

[4]The quantity R is often referred to as the *capacity* of the channel

[5]Typically the normalizing term $\frac{1}{\log|\mathcal{X}|}$ is not present when stating Shannon's capacity theorem. This is because the "rate" used in Shannon capacity is often defined as $\frac{\log|C|}{n}$, whereas the rate for us is defined as $\frac{\log|C|}{n \log|\mathcal{X}|}$.

We have $|\mathcal{X}'| \geq \frac{1}{2}|\mathcal{X}|$, so μ assigns probability at most $\frac{2}{2^m}$ to each element of $\mathcal{X}'$. Since λ assigns probability at least $\frac{1}{(m\log m)2^m}$ to each element of $\mathcal{X}'$, $\nu(x) \geq 0$ for all x. Furthermore, it is easy to check that $\sum_{x\in\mathcal{X}} \nu(x) = 1$, so $\nu(x)$ is a legitimate probability distribution. We can sample from λ as follows: with probability $\frac{1}{2m\log m}$ sample from μ, otherwise, sample from ν. Thus, for any recovery algorithm $f : \mathcal{Y}_{m,T} \to \mathcal{X}_m$.

$$\begin{aligned}
&\mathbf{Pr}[f(Y_\lambda) \neq X_\lambda] \\
&= \frac{1}{2m\log m}\mathbf{Pr}[f(Y_\mu) \neq X_\mu] \\
&\quad + \left(1 - \frac{1}{2m\log m}\right)\mathbf{Pr}[f(Y_\nu) \neq X_\nu] \\
&\geq \frac{1}{2m\log m} \cdot \mathbf{Pr}[f(Y_\mu) \neq X_\mu] \\
&\geq \frac{1}{2m\log m} \cdot \frac{1}{8} = \frac{1}{16m\log m}.
\end{aligned}$$

The last inequality is uses (IV-B). Thus, $H(X_\lambda|Y_\lambda) \geq \frac{1}{32m\log m}$ by Lemma IV.3. We thus may bound

$$\begin{aligned}
I(X_\lambda, Y_\lambda) &= H(X_\lambda) - H(X_\lambda|Y_\lambda) \\
&\leq \log|\mathcal{X}| - H(X_\lambda|Y_\lambda) \\
&\leq m - \frac{1}{32m\log m}.
\end{aligned}$$

This covers all cases, completing the proof. ∎

Proof of Theorem IV.1: Recall $m = \lfloor\sqrt{\frac{1/\varepsilon}{128\log(1/\varepsilon)}}\rfloor$ and $T = T^{(\text{avg})}_{q,0}(m) - 1$. Let n_0' be the constant given by Theorem IV.2 with the parameter $\gamma \stackrel{\text{def}}{=} \varepsilon m$. Let $n_0 \stackrel{\text{def}}{=} m \cdot n_0' \leq O(\frac{1}{\varepsilon^2})$.

We first prove that codes of rate $1 - 2\varepsilon$ are not (T, q, δ) trace reconstructible when n is any sufficiently large multiple of m. Let C be a code that is (T, q, δ) trace reconstructible when $n \geq n_0$ is a multiple of m. We show C must have rate less than $1 - 2\varepsilon$. Let $n_{out} \stackrel{\text{def}}{=} \frac{n}{m}$. For each $i \in [n_{out}]$, given a codeword $c = (c_1, \ldots, c_n) \in C$, let X_i denote the string

$$X_i \stackrel{\text{def}}{=} c_{(i-1)m+1}, c_{(i-1)m+2}, \ldots, c_{im}.$$

Let $Y_i \in \mathcal{Y}_{m,T}$ be a tuple of T of strings distributed as independent traces of X_i under the BDC_q. By assumption of our code, it is possible to recover c from $Y_1, \ldots, Y_{n_{out}}$ with failure probability at most δ: take the trace-wise concatenation of $Y_1, \ldots, Y_{n_{out}}$ and use the trace reconstruction algorithm that is assumed. Hence, the code C, when interpreted as a code in $\mathcal{X}^{n_{out}}$, achieves failure probability δ on the memoryless channel $w(\cdot|\cdot)$ with inputs $\mathcal{X}_m$ and outputs $\mathcal{Y}_{m,T}$ where Y is distributed as T independent traces of X. By Lemma IV.4, we have

$$\max_{\lambda \text{ on } \mathcal{X}_m} I(X_\lambda, Y_\lambda) \leq m\left(1 - \frac{1}{32m^2\log m}\right) \leq m(1 - 4\varepsilon),$$

since ε sufficiently small. By Theorem IV.2, since $\gamma \stackrel{\text{def}}{=} \varepsilon m$ and $n_{out} \geq n_0'$, our code C, when interpreted as a code in $\mathcal{X}^{n_{out}}$, must have rate less than

$$\frac{1}{\log|\mathcal{X}|}\left(\max_{\lambda \text{ on } \mathcal{X}_m} I(X_\lambda, Y_\lambda) + \gamma\right) < 1 - 2\varepsilon,$$

as desired.

Now suppose n is not a multiple of m. Then, suppose for contradiction that $C \subset \{0,1\}^n$ is a code of length n and rate $1 - \varepsilon$ that is (T, q, δ) trace reconstructible. By a simple counting argument, there exists a code $C' \subset \{0,1\}^{n'}$ of rate $1 - \varepsilon - \frac{\varepsilon n'}{n-n'} > 1 - 2\varepsilon$ and a string w such that $c'||w \in C$ for all $c' \in C'$. Furthermore, recovering all codewords of C requires recovering all codewords of the form $c'||w$ for $c' \in C'$. The failure probability of recovering c' from T traces of $c'||w$ is at least the failure probability of recovering c' from T traces of c', which, as we showed, is more than δ, a contradiction. ∎

V. Conclusion and Open Problems

In this paper, we considered the coded trace reconstruction problem. We obtain lower and upper bounds on the problem which show that the average-case trace reconstruction problem is essentially equivalent to the coded trace reconstruction problem. Even with this contribution, there are still many questions left unanswered.

1) The most fundamental open question in this space is closing the exponential gaps for the worst-case trace reconstruction and average-case trace-reconstruction. For worst-case trace reconstruction, the optimal number of traces is between $\tilde{\Omega}(n^{3/2})$ and $\exp(O(n^{1/3}))$ (or $\exp(O(n^{1/5}))$ for $q \leq 1/2$), and for average-case trace reconstruction, the optimal number of traces is between $\tilde{\Omega}(\log^{5/2} n)$ and $\exp(O(\log^{1/3} n))$.
2) One way to generalize the coded trace reconstruction model considered in this paper to consider a more general synchronization channel, such as with insertions and deletions. For example, such a model could insert k random bits between x_i and x_{i+1} with probability $(1-q)q^k$ and then apply i.i.d. deletions with probability q. See the recent survey by Cheraghchi and Ribeiro [CR20] for an overview of various models for random insertions, deletions, substitutions and replications. The authors suspect that similar primitives to those used in this paper could be useful in these more general settings.
3) Another combinatorial variant of this question is *necklace reconstruction*. This question is similar to ordinary trace reconstruction, except a random cyclic shift is also applied to each trace, and the original string needs to be recovered up to an arbitrary cyclic shift. Many protocols for the traditional trace reconstruction problem exploit that the initial prefix of the trace can be easily determined by looking at the prefixes of

the traces. For necklace reconstruction, this strategy would no longer work (due to the random shift), so new techniques need to be developed. Even beating $O((1-q)^{-n})$ traces, the probability of receiving the whole necklace as a trace, seems nontrivial. A recent paper [NR20] studies this problem.

4) A challenging question in the context of coded trace reconstruction is formulating other interesting models beyond i.i.d. deletions. Adversarial deletions is not an interesting model because the adversary could delete the same bits on each trace, reducing the problem to the deletion code problem. One possibility of such a model would be adversarial deletions subject to some global constraints–such as the distribution of deletions being approximately k-wise independent.
5) Another challenge is coming up with deletion models and codes that more accurately correspond to practical use cases and string lengths. Trace reconstruction as used in DNA computing often considers string of approximately length 100 (e.g., [OAC$^+$18]). Constructing such codes may require different techniques than those used in this paper.
6) We do not know if Theorem I.12 achieves the smallest alphabet size for $O(\log_{1/q} \frac{1}{\varepsilon})$ traces. It would be interesting to determine the trade-off between alphabet size and number of traces.

VI. Acknowledgements

We thank Mary Wootters for sponsoring the Coding Theory Reading Group at Stanford where this project was started. We thank Nina Holden for helpful discussions on the error probabilities in the paper [HPP18]. We thank Venkatesan Guruswami for help discussions on high rate error correcting codes and suggesting the Justesen code construction for use in the binary upper bound.We thank João Ribeiro for helpful discussions about the work [CGMR20] and feedback on an earlier draft of the paper. We thank Wesley Pegden for suggesting one of the open problems. We thank Venkatesan Guruswami, Mary Wootters, Aviad Rubinstein, Moses Charikar, and Sivakanth Gopi for helpful discussions, encouragement, and feedback on an earlier draft of the paper.

Joshua is supported by an NSF Graduate Research Fellowship. Ray is supported by an NSF Graduate Research Fellowship grant DGE-1656518 and NSF grant CCF-1814629.

References

[AVDiF19] Mahed Abroshan, Ramji Venkataramanan, Lara Dolecek, and Albert Guillén i Fàbregas. Coding for deletion channels with multiple traces. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1372–1376. IEEE, 2019.

[BCF$^+$19] Frank Ban, Xi Chen, Adam Freilich, Rocco A Servedio, and Sandip Sinha. Beyond trace reconstruction: Population recovery from the deletion channel. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 745–768. IEEE, 2019.

[BCSS19] Frank Ban, Xi Chen, Rocco A Servedio, and Sandip Sinha. Efficient average-case population recovery in the presence of insertions and deletions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[BKKM04] Tugkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. *SODA*, 2004.

[BLS19] Joshua Brakensiek, Ray Li, and Bruce Spang. Coded trace reconstruction in a constant number of traces. *arXiv preprint arXiv:1908.03996*, 2019.

[CGMR20] Mahdi Cheraghchi, Ryan Gabrys, Olgica Milenkovic, and Joao Ribeiro. Coded trace reconstruction. *IEEE Transactions on Information Theory*, 2020.

[Cha19] Zachary Chase. New Lower Bounds for Trace Reconstruction. *arXiv.org*, May 2019.

[Cha20] Zachary Chase. New upper bounds for trace reconstruction. *arXiv preprint arXiv:2009.03296*, 2020.

[Che18] Mahdi Cheraghchi. Capacity upper bounds for deletion-type channels. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 493–506, 2018.

[CR20] Mahdi Cheraghchi and João Ribeiro. An overview of capacity results for synchronization channels. *IEEE Transactions on Information Theory*, 2020. to appear.

[CS20] Roni Con and Amir Shpilka. Explicit and efficient constructions of coding schemes for the binary deletion channel. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 84–89. IEEE, 2020.

[DG01] Suhas Diggavi and Matthias Grossglauser. On transmission over deletion channels. In *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing*, pages 573–582, 2001.

[DM06] Eleni Drinea and Michael Mitzenmacher. On lower bounds for the capacity of deletion channels. *IEEE Trans. Information Theory*, 52(10):4648–4657, 2006.

[DM07] Eleni Drinea and Michael Mitzenmacher. Improved lower bounds for the capacity of i.i.d. deletion and duplication channels. *IEEE Trans. Information Theory*, 53(8):2693–2714, 2007.

[DOS17] Anindya De, Ryan O'Donnell, and Rocco A Servedio. Optimal mean-based algorithms for trace reconstruction. *STOC*, pages 1047–1056, 2017.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[FD10] Dario Fertonani and Tolga M. Duman. Novel bounds on the capacity of the binary deletion channel. *IEEE Trans. Information Theory*, 56(6):2753–2765, 2010.

[GI05] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Trans. Information Theory*, 51(10):3393–3400, 2005.

[GL19] Venkatesan Guruswami and Ray Li. Polynomial time decodable codes for the binary deletion channel. *IEEE Trans. Information Theory*, 65(4):2171–2178, 2019.

[GM19] Ryan Gabrys and Olgica Milenkovic. Unique reconstruction of coded strings from multiset substring spectra. *IEEE Transactions on Information Theory*, 65(12):7682–7696, 2019.

[HL$^+$20] Nina Holden, Russell Lyons, et al. Lower bounds for trace reconstruction. *Annals of Applied Probability*, 30(2):503–525, 2020.

[HM14] Bernhard Haeupler and Michael Mitzenmacher. Repeated deletion channels. *2014 IEEE Information Theory Workshop (ITW 2014)*, pages 152–156, August 2014.

[HMPW08] Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. Trace reconstruction with constant deletion probability and related results. *SODA*, 2008.

[HPP18] Nina Holden, Robin Pemantle, and Yuval Peres. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. In *Conference On Learning Theory*, pages 1799–1840, 2018.

[HS17] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization Strings: Codes for Insertions and Deletions Approaching the Singleton Bound. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing - STOC 2017*, pages 33–46, 2017.

[HS18] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: Explicit constructions, local decoding, and applications. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 841–854, 2018.

[KM13] Yashodhan Kanoria and Andrea Montanari. Optimal coding for the binary deletion channel with small deletion probability. *IEEE Trans. Information Theory*, 59(10):6192–6219, 2013.

[KMMP19] Akshay Krishnamurthy, Arya Mazumdar, Andrew McGregor, and Soumyabrata Pal. Trace reconstruction: Generalized and parameterized. In *27th Annual European Symposium on Algorithms (ESA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[KMS10] Adam Kalai, Michael Mitzenmacher, and Madhu Sudan. Tight asymptotic bounds for the deletion channel with small deletion probabilities. In *2010 IEEE International Symposium on Information Theory*, pages 997–1001. IEEE, 2010.

[Lev01a] Vladimir I Levenshtein. Efficient reconstruction of sequences. *IEEE Trans. Information Theory*, 47(1):2–22, 2001.

[Lev01b] Vladimir I Levenshtein. Efficient Reconstruction of Sequences from Their Subsequences or Supersequences. *Journal of Combinatorial Theory*, 2001.

[MPV14a] Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace Reconstruction Revisited. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014*, volume 8737, pages 689–700. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[MPV14b] Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace Reconstruction Revisited. *ESA*, 8737(2):689–700, 2014.

[NP17] Fedor Nazarov and Yuval Peres. Trace reconstruction with exp(O(n1/3)) samples. *STOC*, 2017.

[NR20] Shyam Narayanan and Michael Ren. Circular trace reconstruction. *arXiv preprint arXiv:2009.01346*, 2020.

[OAC$^+$18] Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, et al. Random access in large-scale dna data storage. *Nature biotechnology*, 36(3):242, 2018.

[PZ17] Yuval Peres and Alex Zhai. Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 228–239, 2017.

[RD15] Mojtaba Rahmati and Tolga M. Duman. Upper bounds on the capacity of deletion channels using channel fragmentation. *IEEE Trans. Information Theory*, 61(1):146–156, 2015.

[Sha48] Claude Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3):379423, 1948.

[TI68] D. Tebbe and Samuel J. Dwyer III. Uncertainty and the probability of error (corresp.). *IEEE Trans. Information Theory*, 14(3):516–518, 1968.

[VT65] RR Varshamov and GM Tenengolts. Codes which correct single asymmetric errors (in russian). *Automatika i Telemkhanika*, 161(3):288–292, 1965.

[Wol78] Jacob Wolfowitz. *Coding theorems of information theory*. Springer-Verlag, 1978.

[YGM17] S. M. Hossein Tabatabaei Yazdi, Ryan Gabrys, and Olgica Milenkovic. Portable and Error-Free DNA-Based Data Storage. *Scientific Reports*, 7:5011, 2017.