Polynomial-time trace reconstruction in the smoothed complexity model

XI CHEN†, Columbia University, USA
ANINDYA DE‡, University of Pennsylvania, USA
CHIN HO LEE§, Harvard University, USA
ROCCO A. SERVEDIO¶, Columbia University, USA
SANDIP SINHA∥, Columbia University, USA

In the trace reconstruction problem, an unknown source string $x \in \{0, 1\}^n$ is sent through a probabilistic deletion channel which independently deletes each bit with probability $\delta$ and concatenates the surviving bits, yielding a trace of $x$. The problem is to reconstruct $x$ given independent traces. This problem has received much attention in recent years both in the worst-case setting where $x$ may be an arbitrary string in $\{0, 1\}^n$ [DOS19, NP17, HHP18, HL20, Cha21a, Cha21b] and in the average-case setting where $x$ is drawn uniformly at random from $\{0, 1\}^n$ [PZ17, HPP18, HL20, Cha21a, Cha21b].

This paper studies trace reconstruction in the smoothed analysis setting, in which a “worst-case” string $x_{\text{worst}}$ is chosen arbitrarily from $\{0, 1\}^n$, and then a perturbed version $x$ of $x_{\text{worst}}$ is formed by independently replacing each coordinate by a uniform random bit with probability $\sigma$. The problem is to reconstruct $x$ given independent traces from it.

Our main result is an algorithm which, for any constant perturbation rate $0 < \sigma < 1$ and any constant deletion rate $0 < \delta < 1$, uses poly$(n)$ running time and traces and succeeds with high probability in reconstructing the string $x$. This stands in contrast with the worst-case version of the problem, for which $\exp(\tilde{O}(n^{1/5}))$ is the best known time and sample complexity [Cha21b].

Our approach is based on reconstructing $x$ from the multiset of its short subwords and is quite different from previous algorithms for either the worst-case or average-case versions of the problem. The heart of our work is a new poly$(n)$-time procedure for reconstructing the multiset of all $O(\log n)$-length subwords of any source string $x \in \{0, 1\}^n$ given access to traces of $x$.

CCS Concepts: ♦ Mathematics of computing → Probabilistic inference problems; ♦ Theory of computation → Design and analysis of algorithms.

Additional Key Words and Phrases: trace reconstruction, smoothed analysis

* A preliminary version appeared in the proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2021.
† Supported by NSF grants CCF-1703925 and IIS-1838154.
‡ Supported by NSF grants CCF-1926872 and CCF-1910534.
§ Supported by a grant from the Croucher Foundation and by the Simons Collaboration on Algorithms and Geometry. Work done while at Columbia University.
¶ Supported by NSF grants CCF-1814873, IIS-1838154, CCF-1563155, and by the Simons Collaboration on Algorithms and Geometry.
∥ Supported by NSF grants CCF-1814873, CCF-18022809, IIS-1838154, CCF-1617955, CCF-1740833, and by the Simons Collaboration on Algorithms and Geometry.

Authors’ addresses: Xi Chen, xichen@cs.columbia.edu, Columbia University, USA; Anindya De, anindyad@cis.upenn.edu, University of Pennsylvania, USA; Chin Ho Lee, chlee@seas.harvard.edu, Harvard University, USA; Rocco A. Servedio, rocco@cs.columbia.edu, Columbia University, USA; Sandip Sinha, sandip@cs.columbia.edu, Columbia University, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.
1549-6325/2022/8-ART $15.00
https://doi.org/10.1145/3560819

ACM Trans. Algor.
1 INTRODUCTION

Trace reconstruction is a simple-to-state algorithmic problem which has been intensively studied yet remains mysterious in many respects. The problem captures some of the core algorithmic challenges that arise in dealing with the deletion channel; this is a noise process which, when given an input string, independently deletes each coordinate with some fixed probability $\delta$ and outputs the concatenation of the surviving coordinates. In the trace reconstruction problem an algorithm is given access to independent traces of a fixed unknown string $x \in \{0, 1\}^n$, where a "trace" of $x$, denoted $z \sim \text{Del}_\delta(x)$, is the string $z$ that results from passing $x$ through a deletion channel. The task is to use these traces to reconstruct the unknown string $x$.

Variants of the trace reconstruction problem have a long history, going back at least to [Kal73]. The problem was studied on and off throughout the 2000s [Lev01b, Lev01a, BKKM04, KM05, VS08, HMPW08, MPV14], and has seen a renewed surge of recent interest over the past few years [DOS19, NP17, PZ17, HPP18, HHP18, Cha21a, BCF+19, BCSS19, KMMP19, Nar21, HPPZ20] with the development of new algorithms and lower bounds for both the worst-case and average-case versions of the problem as well as various generalizations. Below we describe these two versions of the problem and recall the current state of the art for each of them.

1.1 Prior work: Worst-case and average-case trace reconstruction

The original version of the trace reconstruction problem is the worst-case version, in which the unknown string $x$ is an arbitrary (i.e. adversarially chosen) string from $\{0, 1\}^n$. This version of the problem has proved to be quite challenging; the first non-trivial result is due to Batu et al. [BKKM04], who gave a poly$(n)$-time algorithm that uses poly$(n)$ traces and succeeds when the deletion rate $\delta$ is very small, at most $n^{-1/2-\epsilon}$ for any $\epsilon > 0$. In [HMPW08] Holenstein et al. gave an algorithm that runs in $\exp(\tilde{O}(n^{1/2}))$ time using $\exp(\tilde{O}(n^{1/2}))$ traces and succeeds for any $\delta$ bounded away from 1 by a constant. Simultaneous and independent works of De et al. [DOS17] (see also [DOS19]) and Nazarov and Peres [NP17] gave an algorithm that improves the running time and sample complexity of [HMPW08] to $\exp(O(n^{1/3}))$. Recently, this was improved to $\exp(\tilde{O}(n^{1/3}))$ by Chase [Cha21b]. In this same constant-$\delta$ regime, successively stronger lower bounds on the required sample complexity were given by [MPV14, HL20], culminating in a $\Omega(n^{3/2})$ lower bound due to Chase [Cha21a].

Another natural variant of the trace reconstruction problem is the average-case version; in this variant the unknown string $x$ is assumed to be drawn uniformly at random from $\{0, 1\}^n$, and the goal is for the algorithm to succeed with high probability over the random choice of $x$. This problem variant is motivated both by the apparent difficulty of the worst-case problem and by the fact that in various application domains it may be overly pessimistic to assume that the input string $x$ is adversarially generated. Much more efficient algorithms are known for the average-case problem: several early works [BKKM04, KM05, VS08] gave efficient algorithms that succeed for trace reconstruction of almost all $x \in \{0, 1\}^n$ for various $o(1)$ deletion rates $\delta$, and [HMPW08] gave an algorithm that runs in poly$(n)$ time using poly$(n)$ traces when $\delta$ is at most some sufficiently small constant. More recent results of Peres and Zhai [PZ17] and Holden et al. [HPP18, HPPZ20], which build on worst-case trace reconstruction results of [DOS19, NP17], substantially improve on this, with [HPP18, HPPZ20] giving an algorithm which uses $\exp(O(\log^{1/3} n))$ traces to reconstruct a random $x \in \{0, 1\}^n$ in $n^{1+o(1)}$ time when the deletion rate is any constant bounded away from 1.

Summarizing the results described above, the current $\exp(\tilde{O}(n^{1/5}))$ state-of-the-art for worst-case trace reconstruction is exponentially higher than the current $\exp(O(\log^{1/3} n))$ state-of-the-art for average-case trace reconstruction. Given this substantial gap, it is natural to investigate intermediate formulations of the problem between the worst-case and average-case models.
1.2 This work: Smoothed analysis of trace reconstruction

The well-studied smoothed analysis model, introduced by Spielman and Teng [ST04], provides a natural framework for interpolating between worst-case and average-case complexity. In smoothed analysis the input to an algorithm is obtained by applying a random \( \sigma \)-perturbation to a worst-case input instance; here \( \sigma \) is a “perturbation rate,” which it is natural to scale so that \( \sigma = 1 \) corresponds to a truly random instance and \( \sigma = 0 \) corresponds to a worst-case instance. By choosing intermediate settings of \( \sigma \) it is possible to interpolate between worst-case and average-case problem variants.

We now give a detailed statement of the smoothed trace reconstruction problem that we consider. First, a “worst-case” string \( x^{\text{worst}} \) is chosen arbitrarily from \( \{0, 1\}^n \), and then a randomly perturbed version \( x \) of the string \( x^{\text{worst}} \) is formed by independently replacing each coordinate of \( x^{\text{worst}} \) by a uniform random bit with probability \( \sigma \). The goal is to reconstruct \( x \) given access to independent traces drawn from \( \text{Del}_\delta(x) \). Note that when \( \sigma = 0 \) this reduces to the worst-case trace reconstruction problem, and when \( \sigma = 1 \) this reduces to the average-case problem.

Note also that for simplicity we suppose that the exact values of \( \sigma \), \( \delta \), and \( \delta \) are provided to the reconstruction algorithm (but the only information that the algorithm has about \( \sigma \) is what can be gleaned from the traces).

As our main result, we give an algorithm for the smoothed trace reconstruction problem. For any initial string \( x^{\text{worst}} \), our algorithm can recover a \( 1 - 1/\text{poly}(n) \) fraction of perturbed strings \( x \) obtained from \( x^{\text{worst}} \) (for any \( \text{poly}(n) \)) in polynomial time for any constant perturbation rate \( 0 < \sigma \leq 1 \) and any constant deletion rate \( 0 < \delta < 1 \). More precisely, the main theorem we prove is the following:

**Theorem 1 (Polynomial time smoothed trace reconstruction).** Let \( 0 < \delta, \eta, \tau < 1 \) and \( 0 < \sigma \leq 1 \). Let \( x^{\text{worst}} \) be an arbitrary and unknown string in \( \{0, 1\}^n \) and let \( x \) be formed from \( x^{\text{worst}} \) by independently replacing each bit of \( x^{\text{worst}} \) with a uniform random bit from \( \{0, 1\} \) with probability \( \sigma \).

There is an algorithm with the following guarantee: with probability at least \( 1 - \eta \) (over the random generation of \( x \) from \( x^{\text{worst}} \)), it is the case that the algorithm, given access to independent traces drawn from \( \text{Del}_\delta(x) \), outputs the string \( x \) with probability at least \( 1 - \tau \) (over the random traces drawn from \( \text{Del}_\delta(x) \)). Its running time, as well as the number of traces it uses, is

\[
\left( \frac{n}{\eta} \right)^{O\left( \frac{1}{\sigma} \log \frac{1}{\tau} \right)} \log \frac{1}{\tau}.
\]

It is interesting that while the best currently known algorithms for the worst-case problem, corresponding to \( \sigma = 0 \), require \( \exp(O(n^{1/3})) \) time, for any constant perturbation rate we can solve the problem in a dramatically more efficient way. Intuitively, this shows that worst-case instances for trace reconstruction are “few and far between,” in the sense that even a small perturbation of such an instance typically makes it much easier to solve.

1.3 Techniques

Before describing our approach we briefly recall some of the methods used in prior work for the worst-case and average-case problems and discuss why these approaches do not seem applicable to the smoothed problem that we consider.

**Worst-case algorithms.** All of the known worst-case algorithms [HMPW08, DOS19, NP17] (prior to the initial publication of this work) for deletion rates bounded away from 1 are “mean-based,” meaning that they only use estimates of the \( n \) expected values \( E_{y - \text{Del}_\delta(x)}[y_i], \) \( i = 1, \ldots, n \). The two papers [DOS19, NP17] both show that mean-based algorithms can only succeed if they are given estimates of these expectations that are additively \( \pm \exp(-\Omega(n^{1/3})) \)-accurate, and hence mean-based algorithms must inherently use \( \exp(\Omega(n^{1/3})) \) traces for the worst-case problem. Inspection of [DOS19, NP17] shows that these worst-case lower bounds for mean-based algorithms in fact hold for a \( 1 - o_n(1) \) fraction of strings in \( \{0, 1\}^n \). Thus, the mean-based algorithmic approach of
[HMPW08, DOS19, NP17] will not work for our smoothed variant of the problem (and indeed our algorithm is not a mean-based algorithm).

**Average-case algorithms.** The average-case algorithms of [PZ17, HPP18, HPPZ20] work by aligning individual traces (and are not mean-based). The analysis builds off of some of the structural results established in [DOS19, NP17], but also employs sophisticated probabilistic arguments which heavily depend on the randomness of the source string \( x \) being reconstructed; in particular, in the smoothed setting it is not at all clear whether there is a suitable test that makes alignment possible.

As noted in [HPP18, HPPZ20], their average-case algorithm extends to the setting in which the target string \( x \) is drawn from the \( p \)-biased distribution over \( \{0, 1\}^n \) (under which each bit \( x_i \) is independently taken to be 1 with probability \( p \)). Taking \( p = \sigma/2 \), this corresponds to our smoothed analysis model in the special case in which the original string \( x^{\text{worst}} \) is promised to be the string \( 0^n \). Equivalently, we can view our smoothed analysis problem as a more challenging variant of \( p \)-biased average-case trace reconstruction — more challenging because the initial string \( (x^{\text{worst}}) \) is no longer promised to be \( 0^n \), but rather is both arbitrary and moreover unknown to the reconstruction algorithm. It is not clear how to extend the \( p \)-biased average-case results of [HPP18, HPPZ20] even to the setting in which the starting string \( x^{\text{worst}} \) is a known arbitrary string, let alone to our setting in which \( x^{\text{worst}} \) is both arbitrary and unknown. (On the other hand, it should be noted that the algorithms of [HPP18, HPPZ20] use a sub-polynomial number of traces for the average-case problem, while our algorithm uses polynomially many traces for the smoothed problem.)

### 1.4 Our approach: Reconstruction from subwords and the subword deck reconstruction problem

In contrast with prior algorithms for the worst-case and average-case problem, our approach is based on first reconstructing **subwords** of the target string and then reconstructing the target string from those subwords. Recall that a subword of a string \( x = (x_0, \ldots, x_{n-1}) \) is a sequence of contiguous characters of \( x \), i.e. a \( (b-a+1) \)-character string \( x[a:b] := (x_a, x_{a+1}, \ldots, x_b) \) for some \( 0 \leq a \leq b \leq n-1 \).

**Reconstruction from subwords.** Given a length \( 1 \leq k \leq n \), let us write subword\((x,k)\) to denote the **multiset** of all \( n-k+1 \) length-\( k \) subwords of \( x \); we refer to this multiset as the **k-subword deck** of \( x \). For example, if \( n = 7 \) and \( k = 3 \), then the \( k \)-subword deck of \( x = 1101011 \) would be the 5-element multiset \([010, 011, 101, 101, 110] \).

In general the \( k \)-subword deck of \( x \) may not uniquely identify the string \( x \) within \([0, 1]^n\) unless \( k \) is very large; for example, the two multisets

\[
\text{subword}\left(0^n/4, 1^n/4, 0^n/4, 1^n/4, 1^n/4\right), k) \quad \text{and} \quad \text{subword}\left(0^n/1, 1^n/4, 1^n/4, 1^n/4, 1^n/4\right), k)
\]

are identical for every \( k \leq n/4 - 1 \). This simple example shows that for worst-case strings \( x \), the \( k \)-subword deck of \( x \) may not suffice to information-theoretically specify \( x \) unless \( k \) is linear in \( n \) (and achieving \( k \) linear in \( n \) seems to necessitate exponential time and sample complexity).

The starting point of our approach is the observation that the situation is markedly better for **random perturbations** of worst-case strings: for any worst-case string \( x^{\text{worst}} \in \{0, 1\}^n \), with high probability a random \( \sigma \)-perturbation \( x \) of \( x^{\text{worst}} \) is such that subword\((x,k)\) does uniquely identify \( x \) within \([0, 1]^n\) even if \( k \) is relatively small. Moreover, there is an efficient algorithm to reconstruct \( x \) from subword\((x,k)\). This is captured by the following result, which we prove in Section 3:

**Lemma 2 (Reconstructing perturbed strings from their subword decks).** Let \( 0 < \sigma, \eta < 1 \). There is a deterministic algorithm **Reconstruct-from-subword-deck** which takes as input the \( k \)-subword deck subword\((x,k)\) of a string \( x \in \{0, 1\}^n \), where \( k = \Theta(\log(n/\eta)/\sigma) \), and outputs either a string in \([0, 1]^n\) or "fail." **Reconstruct-from-subword-deck** runs in \( \text{poly}(n) \) time and has the following property: for any \( x^{\text{worst}} \in \{0, 1\}^n \), if \( x \) is a random \( \sigma \)-perturbation of
\(x^{\text{worst}}\) (i.e. \(x\) is obtained by independently replacing each bit of \(x^{\text{worst}}\) with a uniform random bit with probability \(\sigma\)), then with probability at least \(1 - \eta\) the output of \text{Reconstruct-from-subword-deck} on input \(\text{subword}(x, k)\) is the string \(x\).

\textbf{The subword deck reconstruction problem.} Lemma 2 naturally motivates the algorithmic problem of \textit{subword deck reconstruction}: given access to independent traces drawn from \(\text{Del}_0(x)\) and a length \(k\), can we reconstruct the \(k\)-subword deck of \(x\)? Our main algorithmic contribution is an efficient algorithm for this problem:

**Theorem 3 (Reconstructing the \(k\)-subword deck of \(x\)).** Let \(0 < \delta, \tau < 1\). There is an algorithm, which we call \text{Reconstruct-subword-deck}, which takes as input a parameter \(1 \leq k \leq n\) and access to independent traces of an unknown source string \(x \in \{0, 1\}^n\). The running time of \text{Reconstruct-subword-deck}, as well as the number of traces it uses, is

\[
\left( n \frac{2}{1 - \delta} \right)^{O(1/(1-\delta))} \frac{2}{\log \tau}.
\]

\text{Reconstruct-subword-deck} has the following property: for any string \(x \in \{0, 1\}^n\), with probability at least \(1 - \tau\) the output of \text{Reconstruct-subword-deck} is the \(k\)-subword deck subword\((x, k)\).

Theorem 1 follows immediately from Lemma 2 and Theorem 3, taking the "\(k\)" of Theorem 3 to be \(\Theta((\log n)/\eta)/\sigma\) as in Lemma 2. We note that Theorem 3 dominates the overall running time of Theorem 1, and that Theorem 3 works for arbitrary strings.

The algorithm in Lemma 2 and its analysis are relatively straightforward. To explain the main idea, we define the notion of the right (and left) extension of a string. (Starting from this point, it will be convenient for us to index a string \(x \in \{0, 1\}^n\) using \(0, \ldots, n-1\) as \(x = (x_0, \ldots, x_{n-1})\).

**Definition 4.** Given a \(k\)-bit string \(w = (w_0, \ldots, w_{k-1}) \in \{0, 1\}^k\), a \(k\)-bit string \((w_1, \ldots, w_{k-1}, b)\) for some \(b \in \{0, 1\}\) is said to be a right-extension of \(w\). We define left-extensions of a string similarly.

At a high level, the algorithm relies on the fact that if \(x\) is obtained by a random \(\sigma\)-perturbation of \(x^{\text{worst}}\), then \(x\) has useful local uniqueness properties. More precisely, for \(k = O((\log n)/\tau)/\sigma\), a simple probabilistic argument shows that with high probability \(x[n - k : n - 1]\) is the unique element of subword\((x, k)\) with no right-extension in subword\((x, k)\). Consequently, we can identify \(x[n - k : n - 1]\) from the \(k\)-subword deck subword\((x, k)\) of \(x\). This argument can be extended inductively without much difficulty to in fact identify the whole of \(x\).

In contrast, Theorem 3 is substantially more challenging. The structural results that underlie Theorem 3 are based on two different sets of analytic arguments. The first argument only works when \(\delta \leq 1/2\) and employs (real) Taylor series; the second argument works for the entire range of \(\delta < 1\) and employs tools from complex analysis. While the first argument is more limited in scope of applicability, it is somewhat more elementary (which we see as a positive feature) and introduces a new ingredient (the so-called "generalized deletion polynomial," see Section 5.2) which might be useful in future work, and thus we include both arguments in the paper. In this proof overview below we only describe the second argument.

We begin by observing that subword\((x, k)\) can be obtained by computing the multiplicity of occurrences of each \(w \in \{0, 1\}^k\) in the set subword\((x, k)\); we denote this multiplicity by \#\((w, x)\). The first key step is to define a univariate polynomial (in the variable \(\zeta\)) \(\text{SW}_{x, w}(\zeta)\) which has the following two key properties: (i) \(\text{SW}_{x, w}(0) = \#(w, x)\), and (ii) using traces from \(\text{Del}_0(x)\), we have an unbiased estimator for \(\text{SW}_{x, w}(\zeta)\) for \(\zeta = \delta\). Next, observe that given traces from \(\text{Del}_0(x)\), we can trivially simulate traces from \(\text{Del}_\tau(x)\) for any \(\delta' \geq \delta\), and hence we can get an unbiased estimator for \(\text{SW}_{x, w}(\zeta)\) for \(\zeta \in [\delta, 1]\). Recall, though, that our goal is to estimate \(\text{SW}_{x, w}(\zeta)\) at \(\zeta = 0\) and thus items (i) and (ii) above do not give us an unbiased estimator for \(\text{SW}_{x, w}(0)\).

The most obvious idea at this point would be to do polynomial interpolation and use estimates for \(\text{SW}_{x, w}(\zeta)\) for \(\zeta \in [\delta, 1]\) to infer \(\text{SW}_{x, w}(0)\). Unfortunately, directly applying Lagrange interpolation is too naive an approach: to
We view this paper as a first exploration, establishing that the algorithmic framework of smoothed analysis can closely relate to Theorem 5.1 of [BEK99] (see also [BE97]), implies that SW$_{\alpha}$ to be (to be the $\ell$-subword of $\ell$-subword deck). Given such a string $x \in \{0,1\}^n$, we write $x[a:b]$ to denote the subword $(x_a, x_{a+1}, \ldots, x_b)$. We write $\lceil \log \rceil$ to denote natural logarithm and $\log$ to denote logarithm to the base 2.

We denote the set of non-negative integers by $\mathbb{Z}_{\geq 0}$. For a vector $\alpha = (\alpha_1, \ldots, \alpha_\ell) \in \mathbb{Z}_{\geq 0}^\ell$, we write $|\alpha|$ to denote $\alpha_1 + \alpha_2 + \cdots + \alpha_\ell$, and write $\alpha!$ to denote $\alpha_1! \cdot \alpha_2! \cdot \cdots \cdot \alpha_\ell!$. We write $\omega$ to denote

**Subword deck.** Fix a string $x \in \{0,1\}^n$ and an integer $k \in [n]$. A $k$-subword of $x$ is a (contiguous) subword of $x$ of length $k$, given by $(x_a, x_{a+1}, \ldots, x_{a+k-1})$ for some $a \in \{0 : n - k\}$. For a string $w \in \{0,1\}^k$, let $\#(w,x)$ denote the number of occurrences of $w$ as a subword of $x$. We define the $k$-subword deck of $x$, denoted subword$(x,k)$, to be the $(n - k + 1)$-size (unordered) multiset of all $k$-subwords of $x$. We also extend the notation of $\#(w,x)$ to

2 PRELIMINARIES

**Notation.** Given a nonnegative integer $n$, we write $[n]$ to denote $\{1, \ldots, n\}$. Given integers $a \leq b$, we write $[a : b]$ to denote $\{a, \ldots, b\}$. It will be convenient for us to index a binary string $x \in \{0,1\}^n$ using $[0 : n - 1]$ as $x = (x_0, \ldots, x_{n-1})$. Given such a string $x$ and integers $0 \leq a \leq b \leq n - 1$, we write $x[a:b]$ to denote the subword $(x_a, x_{a+1}, \ldots, x_b)$. We write $\ln$ to denote natural logarithm and $\log$ to denote logarithm to the base 2.

We denote the set of non-negative integers by $\mathbb{Z}_{\geq 0}$. For a vector $\alpha = (\alpha_1, \ldots, \alpha_\ell) \in \mathbb{Z}_{\geq 0}^\ell$, we write $|\alpha|$ to denote $\alpha_1 + \alpha_2 + \cdots + \alpha_\ell$, and write $\alpha!$ to denote $\alpha_1! \cdot \alpha_2! \cdot \cdots \cdot \alpha_\ell!$. We write $\omega$ to denote

**Subword deck.** Fix a string $x \in \{0,1\}^n$ and an integer $k \in [n]$. A $k$-subword of $x$ is a (contiguous) subword of $x$ of length $k$, given by $(x_a, x_{a+1}, \ldots, x_{a+k-1})$ for some $a \in \{0 : n - k\}$. For a string $w \in \{0,1\}^k$, let $\#(w,x)$ denote the number of occurrences of $w$ as a subword of $x$. We define the $k$-subword deck of $x$, denoted subword$(x,k)$, to be the $(n - k + 1)$-size (unordered) multiset of all $k$-subwords of $x$. We also extend the notation of $\#(w,x)$ to
strings \( w \in \{0, 1, *\}^k \), where \(*\) is the wildcard symbol: \(#(w, x)\) is the sum of \(#(w', x)\) over all \( w' \in \{0, 1\}^k \) with \( w'_i = w_i \) for every \( w_i \neq * \).

**Distributions.** We use bold font letters to denote probability distributions and random variables, which should be clear from the context. We write \( x \sim X \) to indicate that random variable \( x \) is distributed according to distribution \( X \).

**Deletion channel and traces.** Throughout this paper the parameter \( \delta : 0 < \delta < 1 \) denotes the deletion probability. Given a string \( x \in \{0, 1\}^n \), we write \( \text{Del}_\delta(x) \) to denote the distribution of the string that results from passing \( x \) through the \( \delta \)-deletion channel (so the distribution \( \text{Del}_\delta(x) \) is supported on \( \{0, 1\}^{\leq n} \)), and we refer to a string in the support of \( \text{Del}_\delta(x) \) as a trace of \( x \). Recall that a random trace \( y \sim \text{Del}_\delta(x) \) is obtained by independently deleting each bit of \( x \) with probability \( \delta \) and concatenating the surviving bits.\(^1\)

**Perturbation and smoothed analysis.** The perturbation model we consider corresponds to the standard notion of perturbation of an \( n \)-bit string which arises in the analysis of Boolean functions. Given an \( n \)-bit string \( x^{\text{worst}} \in \{0, 1\}^n \), a \( \sigma \)-perturbation of \( x^{\text{worst}} \) is a random string \( x \in \{0, 1\}^n \) obtained by independently setting each coordinate \( x_i \) to be \( x_i^{\text{worst}} \) with probability \( 1 - \sigma \) and to be uniformly random with the remaining probability \( \sigma \). Equivalently, \( x \) is a random string that is \((1 - \sigma)\)-correlated with \( x^{\text{worst}} \), in the notation of Chapter 2 of [O’D14], we may write this as \( x \sim N_{1-\sigma}(x^{\text{worst}}) \).

We recall that in the smoothed analysis framework, an initial string \( x^{\text{worst}} \in \{0, 1\}^n \) is selected (in what may be thought of as an adversarial manner), and then a \( \sigma \)-perturbation \( x \) of \( x^{\text{worst}} \) is drawn at random from \( N_{1-\sigma}(x^{\text{worst}}) \), and the algorithm runs on instance \( x \). The goal is to develop algorithms which, for every \( x^{\text{worst}} \in \{0, 1\}^n \), succeed with high probability on the perturbed instance \( x \sim N_{1-\sigma}(x^{\text{worst}}) \).

3 RECONSTRUCTING PERTURBED WORST-CASE STRINGS FROM THEIR SUBWORD DECKS: PROOF OF LEMMA 2

In this section we prove Lemma 2:

**Restatement of Lemma 2** (Reconstructing perturbed strings from their subword decks). Let \( 0 < \sigma, \eta < 1 \). There is a deterministic algorithm \( \text{Reconstruct-from-subword-deck} \) which takes as input the \( k \)-subword deck \( \text{subword}(x, k) \) of a string \( x \in \{0, 1\}^n \), where \( k = O(\log(n/\eta)/\sigma) \), and outputs either a string in \( \{0, 1\}^n \) or “fail.” The algorithm \( \text{Reconstruct-from-subword-deck} \) runs in \( \text{poly}(n) \) time and has the following property: For any string \( x^{\text{worst}} \in \{0, 1\}^n \), if \( x \) is a random \( \sigma \)-perturbation of \( x^{\text{worst}} \) (i.e., \( x \) is obtained by independently replacing each bit of \( x^{\text{worst}} \) with a uniform random bit with probability \( \sigma \)), then with probability at least \( 1 - \eta \) the output of \( \text{Reconstruct-from-subword-deck} \) on input subword \( (x, k) \) is the string \( x \).

The idea of Lemma 2 is very simple: a probabilistic argument shows that for any worst-case string \( x^{\text{worst}} \), a random \( \sigma \)-perturbation introduces enough variability into \( x \sim N_{1-\sigma}(x^{\text{worst}}) \) so that the \( k \)-subwords comprising the \( k \)-subword deck of \( x \) can be easily pieced together in a unique way to yield \( x \) by a simple greedy algorithm. We now provide details.

Given \( \text{subword}(x, k) \) of a string \( x \in \{0, 1\}^n \), we use the following greedy algorithm to recover \( x \):

1. We will store the output in \( y \), a string of length \( n \).
2. Let \( w \in \text{subword}(x, k) \) be a string that fails to have a right-extension in \( \text{subword}(x, k) \). (Note the only \( k \)-subword of \( x \) that can fail to have a right-extension in \( \text{subword}(x, k) \) is \( x[n-k:n-1] \).) If no such \( w \) exists, return fail; otherwise set \( y[n-k:n-1] = w \) and \( \ell = n-k \).

\(^1\)For simplicity in this work we assume that the deletion probability \( \delta \) is known to the reconstruction algorithm. We note that it is possible to obtain a high-accuracy estimate of \( \delta \) simply by measuring the average length of traces received from the deletion channel.
(3) When \( \ell > 0 \), do the following: Find \( w \in \text{subword}(x, k) \) as a left-extension of \( y[\ell : \ell + k - 1] \). (Note that if \( y \) agrees with \( x \) so far, then such a left-extension must exist.) If \( w \) is not unique (counted with multiplicity), return fail; otherwise set \( y_{\ell-1} = w_0 \) and decrement \( \ell \) by 1.

(4) When \( \ell = 0 \), return \( y \).

It is clear from the description of the greedy algorithm above and comments therein that either it returns fail or there is no ambiguity (in filling in the last \( k \) bits and extending from there bit by bit) and \( x \) is recovered correctly as \( y \) at the end. We use the following definition to capture strings \( x \) on which the greedy algorithm succeeds:

**Definition 5.** An \( n \)-bit string \( x \) is said to be \( k \)-good if

(i) for every \( j \in [n-k] \), there is exactly one string in \( \text{subword}(x, k) \) (counted with multiplicity) that is a left-extension of the subword \( x[j : j + k - 1] \); and

(ii) the subword \( x[n-k : n-1] \) does not have a right-extension in \( \text{subword}(x, k) \).

To prove Lemma 2, it remains only to establish the following claim:

**Claim 6.** Fix any string \( w_{\text{worst}} \in \{0, 1\}^n \). Then for \( k = O(\log(n/\eta)/\sigma) \)

\[
\Pr_{x \sim N_{1-\sigma}(w_{\text{worst}})} \left[ x \text{ is } k \text{-good} \right] \geq 1 - \eta.
\]

**Proof.** Let \( E(x) \) be the event that \( x \) is not \( k \)-good. We observe that for \( E(x) \) to occur, there must exist indices \( 0 \leq i < j \leq n-k+1 \) such that the \((k-1)\)-subwords of \( x \) starting at positions \( i \) and \( j \) are equal, i.e., \( x[i : i+k-2] = x[j : j+k-2] \). In particular, we have the following (where here and subsequently all probabilities are over the random draw of \( x \sim N_{1-\sigma}(w_{\text{worst}}) \)):

\[
\Pr\left[ E(x) \right] \leq \Pr \left[ \exists i, j \text{ such that } x[i : i+k-2] = x[j : j+k-2] \right] \\
\leq \sum_{0 \leq i < j \leq n-k+1} \Pr \left[ x[i : i+k-2] = x[j : j+k-2] \right].
\]

(by a union bound)

Let \( E_{i,j}(x) \) denote the event that \( x[i : i+k-2] = x[j : j+k-2] \). To prove the claim, it suffices to show that \( \Pr[E_{i,j}(x)] \leq \eta/n^2 \) for each fixed pair \( 1 \leq i < j \leq n-k+1 \).

To this end, we write the probability of \( E_{i,j}(x) \) as

\[
\Pr \left[ x_i = x_j \right] \prod_{\ell=1}^{k-2} \Pr \left[ x_{i+\ell} = x_{j+\ell} \mid x_{i+h} = x_{j+h} \text{ for all } h = 0, \ldots, \ell-1 \right].
\]

The first factor \( \Pr \left[ x_i = x_j \right] \) is at most \( 1 - \sigma/2 \) because for any fixed value \( b \) of \( x_i \), \( x_j \) agrees with \( b \) after the perturbation with probability at most \( 1 - \sigma/2 \). We claim that the upper bound of \( 1 - \sigma/2 \) holds for every other factor in the product. For the \( \ell \)th factor, we note that for any fixed values of \( x_{i} \), \( x_{j+1} \) that satisfy the conditioning part \( x_{i+h} = x_{j+h} \) for all \( h = 0, \ldots, \ell-1 \), \( x_{i+\ell} \) agrees with the fixed value of \( x_{i+\ell} \) with probability at most \( 1 - \sigma/2 \).

Thus, by setting \( k = C \log(n/\eta)/\sigma \) for some large enough constant \( C \), we have

\[
\Pr \left[ E_{i,j}(x) \right] \leq (1 - \sigma/2)^{k-1} \leq \exp \left( -\Omega \left( \frac{\log \eta}{\eta} \right) \right) \leq \frac{\eta}{n^2}.
\]

This finishes the proof of the claim. \( \square \)

ACM Trans. Algor.
4 RECONSTRUCTING THE k-SUBWORD DECK: PROOF OF Theorem 3

The remaining task to establish the main result, Theorem 1, is to prove Theorem 3 (restated below), which gives an efficient algorithm to reconstruct the k-subword deck of an arbitrary source string \( x \in \{0,1\}^n \) given access to independent traces of \( x \). Throughout this section, let \( \rho = (1 - \delta)/2 \).

**Restatement of Theorem 3** (Reconstructing the k-subword deck). Let \( 0 < \delta, \tau' < 1 \). There is an algorithm \( \text{Reconstruct-subword-deck} \) which takes as input a parameter \( 1 \leq k \leq n \) and access to independent traces of an unknown source string \( x \in \{0,1\}^n \). The running time of \( \text{Reconstruct-subword-deck} \), as well as the number of traces it uses, is \( \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(1/\tau') \).

\( \text{Reconstruct-subword-deck} \) has the following property: For any unknown source string \( x \in \{0,1\}^n \), with probability at least \( 1 - \tau' \), \( \text{Reconstruct-subword-deck} \) outputs subword \( x, k \).

The main algorithmic ingredient that underlies Theorem 3 is an algorithm for a closely related but slightly simpler problem. This algorithm, which we call \( \text{Multiplicity} \), takes as input a string \( w \in \{0,1\}^k \) and access to independent traces from an unknown source string \( x \), and it outputs \#(w, x), the multiplicity of \( w \) in the \( (n - k + 1) \)-element multiset subword \( x, k \) (note that this multiplicity can be zero if \( w \) is not present as a subword of \( x \)).

**Theorem 7.** Let \( 0 < \delta, \tau < 1 \) and let \( \rho = (1 - \delta)/2 \). There is an algorithm \( \text{Multiplicity} \) which takes as input a string \( w \in \{0,1\}^k \) and access to independent traces of an unknown source string \( x \in \{0,1\}^n \). \( \text{Multiplicity} \) runs in \( \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(1/\tau) \) time and uses \( \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(1/\tau) \) many traces from \( \text{Del}_\delta(x) \), and has the following property: For any unknown source string \( x \in \{0,1\}^n \), with probability at least \( 1 - \tau \) the output of \( \text{Multiplicity} \) is \#(w, x) (i.e. the number of occurrences of \( w \) as a subword of \( x \)).

A standard “branch-and-bound” argument gives Theorem 3 from Theorem 7:

**Proof of Theorem 3 using Theorem 7.** Let \( \ell = \lceil \log n \rceil \). We first consider the case that \( k \leq \ell \). In this case the algorithm \( \text{Reconstruct-subword-deck} \) simply runs \( \text{Multiplicity}(w) \) once for each of the \( 2^k \) strings \( w \in \{0,1\}^k \), with the confidence parameter “\( \tau' \)” for each run of \( \text{Multiplicity} \) set to \( \tau'/2^k \). Since we can reuse the same traces for each of the \( 2^k \) runs, in this case the running time is \( 2^k \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(2^k/\tau') = \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(1/\tau') \) and the sample complexity is \( \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(1/\tau') \).

Next we consider the case that \( k > \ell \). To avoid an exponential running time dependence on \( k \), the algorithm uses a simple “branch-and-prune” approach. In the first stage, similar to the previous paragraph, \( \text{Reconstruct-subword-deck} \) runs \( \text{Multiplicity} \) on each of the \( 2^\ell \) strings \( w \in \{0,1\}^\ell \) with confidence parameter \( \tau'/2(2nk) \), thereby obtaining the \( \ell \)-subword deck subword \( x, \ell \). It then executes \( k - \ell \) many successive stages \( j = 1, 2, \ldots, k - \ell \), where in stage \( j \) the algorithm determines the \( (\ell + j) \)-subword deck of \( x \) using the \( (\ell + j - 1) \)-subword deck of \( x \). It does this in each stage as follows: for each of the (at most \( n \)) distinct strings \( w \in \text{subword}(x, \ell + j - 1) \), the algorithm runs \( \text{Multiplicity}(w_0) \) and \( \text{Multiplicity}(w_1) \), each with confidence parameter \( \tau'/(2nk) \).

The correctness of this approach follows from the trivial fact that an \( (\ell + j) \)-bit string can only be present in subword \( x, \ell + j \) if its \( (\ell + j - 1) \)-bit prefix is present in subword \( x, \ell + j - 1 \). Since there are at most \( n + 2n(\ell - k) < 2kn \) many runs of \( \text{Multiplicity} \) overall, the running time of \( \text{Reconstruct-subword-deck} \) is at most \( O(kn) \cdot \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(2kn/\tau') = \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(2/\tau') \) and the sample complexity is at most \( \left( \frac{n}{\rho^k} \right)^{O(1/\rho)} \log(2/\tau') \), and Theorem 3 is proved.

Thus, in the rest of the paper, we focus on proving Theorem 7.
4.1 The subword polynomial
The following "subword polynomial" plays an important role in our approach:

Definition 8. Given $x \in \{0, 1\}^n$ and $w = (w_0, \ldots, w_{k-1}) \in \{0, 1\}^k$, let $SW_{x,w}(\zeta)$ be the following univariate polynomial of degree $n-k$:

$$SW_{x,w}(\zeta) := \sum_{\substack{\alpha \in \mathbb{Z}_{\geq 0}^k \cap \{0\}^{n-k} \subseteq \mathbb{N}^k \cap \{0\}^{n-k}}} \# (w_0 *^{a_1} w_1 *^{a_2} w_2 \cdots *^{a_{k-1}} w_{k-1}, x) \cdot \zeta^{|\alpha|},$$

where $*$ stands for a wild-card character.

In words, the degree-$\ell$ coefficient of the subword polynomial $SW_{x,w}$ is the number of ways that $w$ arises as a substring of $x$ with a total of exactly $\ell$ extraneous additional characters interspersed among the characters of $w$. In particular, we have that the constant term of $SW_{x,w}$ (i.e. $SW_{x,w}(0)$, since $0^k = 1$) is equal to $\#(w, x)$, the number of occurrences of $w$ as a subword of $x$, which is what Theorem 7 aims to estimate efficiently from traces of $x$.

4.2 Outline of our approach
We prove Theorem 7 by giving two different algorithms depending on the value of the deletion rate $\delta$. The first of these algorithms, $\text{Multiplicity}_{\text{small-}\delta}$, gives a simple and direct approach to compute the value $SW_{x,w}(0) = \#(w, x)$; however this approach requires the deletion rate $\delta$ to be less than $1/2$. This approach is based on analyzing a new object, the "generalized deletion polynomial," that we believe may be useful for subsequent work. The second of these algorithms, $\text{Multiplicity}_{\text{large-}\delta}$, gives a different and somewhat more involved algorithm (involving linear programming and a new extremal result on polynomials, proved using complex analysis) that can be used for any deletion rate $\delta < 1$.

Readers who are interested in a simple approach (albeit one that works only for $\delta < 1/2$) may wish to focus on $\text{Multiplicity}_{\text{small-}\delta}$ (Section 5). Readers who are interested in a more involved approach that succeeds for all $\delta < 1$ may wish to focus on $\text{Multiplicity}_{\text{large-}\delta}$ (Section 6). The two algorithms and analyses are each self-contained; each may be read independently of the other.

For each of the two algorithms, we first give a simpler version of the analysis which establishes a quantitatively weaker version of the result, with an $n^{O(k)}$ running time and sample complexity (ignoring the dependence on other parameters); see the statements of Theorem 9 and Theorem 15, at the beginnings of Section 5 and Section 6 respectively, for detailed statements of these weaker versions. In Section 7 we quantitatively strengthen both Theorem 9 and Theorem 15 to achieve a $\text{poly}(n) \cdot \exp(O(k))$ running time and sample complexity, and thereby complete the proof of Theorem 7.

5 MULTICLICY_{\text{small-}\delta}: AN ALGORITHM FOR DELETION RATE $\delta < 1/2$
In this section we prove Theorem 9, a weaker version of Theorem 7. It gives an algorithm that has $n^{O(k)}$ running time and sample complexity (ignoring the dependence on other parameters) and works when $\delta < 1/2$. Actually, Theorem 9 works when $\delta \leq 1/2$; we only require $\delta < 1/2$ later in Section 7.1 to achieve the improved running time and sample complexity in Theorem 7 based on a similar approach (the running time achieved in that section will depend on how close $\delta$ is to $1/2$).

Theorem 9. Let $0 < \delta \leq 1/2$. There is an algorithm $\text{Multiplicity}_{\text{small-}\delta}'$ which takes as input a string $w \in \{0, 1\}^k$, access to independent traces of an unknown source string $x \in \{0, 1\}^n$, and a parameter $\tau > 0$. $\text{Multiplicity}_{\text{small-}\delta}'$ draws $n^{O(k)} \cdot \log(1/\tau)$ traces from $\text{Del}_\delta(x)$, runs in time $n^{O(k)} \cdot \log(2/\tau)$, and has the following property: For any
unknown source string \( x \in \{0,1\}^n \), with probability at least \( 1 - \tau \) the output of \( \text{Multiplicity}_{\text{small-}\delta} \) is the multiplicity of \( w \) in \( \text{subword}(x, k) \) (i.e., the number of occurrences of \( w \) as a subword of \( x \)).

In Section 7 we will build on Theorem 9 to give a stronger version that has \( \text{poly}(n) \cdot \exp(O(k)) \) running time and sample complexity (ignoring the dependence on other parameters) for \( \delta < 1/2 \).

The rest of this section is organized as follows. In Section 5.1, we give an equivalent expression for \( \text{SW}_{x,w}(\zeta) \) in Theorem 10, which relates the subword polynomial to traces drawn from the deletion channel. The proof uses the generalized deletion polynomial and is presented in Section 5.2. This new expression for \( \text{SW}_{x,w}(\zeta) \) allows one to evaluate \( \text{SW}_{x,w}(\zeta) \) at \( \zeta = 0 \) up to a small error (say, \( +0.1 \)) using traces of \( x \) (see Corollary 11) when \( \delta \leq 1/2 \). Given that \( \text{SW}_{x,w}(0) \) is an integer, the result can be rounded to obtain the exact value of \( \text{SW}_{x,w}(0) \); this finishes the proof of Theorem 9.

We remark that the expression for \( \text{SW}_{x,w}(\zeta) \) given in Theorem 10 works for any \( \zeta \in \mathbb{C} \), when viewing \( \text{SW}_{x,w}(\zeta) \) as a polynomial over \( \mathbb{C} \), and may be useful for subsequent work. Indeed Corollary 11 shows that \( \text{SW}_{x,w}(\zeta) \) can be evaluated at any \( \zeta \in B_{1-\delta}(\delta) \) up to a small error using traces of \( x \), where \( B_{1-\delta}(\delta) \) denotes the closed complex disc with center \( \delta \) and radius \( 1 - \delta \). We need \( \delta \leq 1/2 \) so that \( 0 \in B_{1-\delta}(\delta) \).

### 5.1 Evaluating \( \text{SW}_{x,w}(\zeta) \) for \( \zeta \in B_{1-\delta}(\delta) \) using traces of \( x \)

In the rest of this section we consider \( \text{SW}_{x,w}(\zeta) \) as a polynomial over complex numbers. The main technical ingredient in the algorithm \( \text{Multiplicity}_{\text{small-}\delta} \) is the following theorem, which relates the subword polynomial to traces drawn from the deletion channel:

**Theorem 10.** Let \( x, k \) and \( w \) be as above. Then for all \( \zeta \in \mathbb{C} \) we have

\[
\text{SW}_{x,w}(\zeta) = \frac{1}{(1-\delta)^k} \sum_{\alpha \in \mathbb{Z}^{n-k}_+} \mathbb{E}_{y \sim \text{Del}_\delta(x)} \left[ \#(w_0 \ast^{\alpha_1} w_1 \ast^{\alpha_2} w_2 \cdots w_{k-2} \ast^{\alpha_{k-1}} w_{k-1}, y) \right] \cdot \left( \frac{\zeta - \delta}{1-\delta} \right)^{|\alpha|}.
\]

Before proving Theorem 10 in Section 5.2 we use it to obtain the following corollary.

**Corollary 11 (Corollary of Theorem 10).** Let \( x, k, w \) be as above, and let \( \epsilon > 0 \). Then, given access to traces \( y \sim \text{Del}_\delta(x) \), there exists an algorithm which, given as input any \( \zeta \in B_{1-\delta}(\delta) \), evaluates \( \text{SW}_{x,w}(\zeta) \) up to error \( \pm \epsilon \) with success probability at least \( 1 - \tau \). The algorithm takes

\[
\left( \frac{n}{1-\delta} \right)^{O(k)} \cdot \frac{1}{\epsilon^2} \cdot \log \left( \frac{1}{\tau} \right)
\]

many traces and running time.

Recall that \( \text{SW}_{x,w}(0) = \#(w, x) \). When \( \delta \leq 1/2 \), the disc \( B_{1-\delta}(\delta) \) contains the origin. Therefore, setting \( \epsilon = 1/3 \) in Corollary 11 directly implies an algorithm \( \text{Multiplicity}_{\text{small-}\delta} \) that uses \( (n/(1-\delta)^{O(k)}) \cdot \log(1/\tau) = n^{O(k)} \cdot \log(1/\tau) \) traces and running time to evaluate \( \text{SW}_{x,w}(0) \) up to an error of \( \epsilon = 1/3 \), which succeeds with probability at least \( 1 - \tau \). It then rounds the result to the nearest integer to obtain \( \text{SW}_{x,w}(0) = \#(w, x) \) given that the latter is an integer. This finishes the proof of Theorem 9.

**Proof of Corollary 11.** The algorithm simply draws

\[
s = \left( \frac{n}{1-\delta} \right)^{O(k)} \cdot \frac{1}{\epsilon^2} \cdot \log \left( \frac{1}{\tau} \right)
\]

many traces \( y_1, \ldots, y_s \) of \( x \) and uses them to compute an empirical estimate \( \tilde{E}_a \) of

\[
E_a := \mathbb{E}_{y \sim \text{Del}_\delta(x)} \left[ \#(w_0 \ast^{\alpha_1} w_1 \ast^{\alpha_2} w_2 \cdots w_{k-2} \ast^{\alpha_{k-1}} w_{k-1}, y) \right]
\]

(1)
for each \( \alpha \in \mathbb{Z}^{k-1}_{\geq 0} \) with \(|\alpha| \leq n - k\). This is done by computing \(#(w_0 ^{\alpha_1} w_1 ^{\alpha_2} w_2 \cdots w_{k-2} ^{\alpha_{k-1}} w_{k-1}, y_i)\) for each \( \alpha \) and \( y_i \) (in time polynomial in \( n \)), and then taking the average over \( y_1, \ldots, y_k \) for each \( \alpha \). Given that the number of \( \alpha \)'s is at most \( n^k \), the overall running time is \( s \cdot n^k \cdot \text{poly}(n) \), as stated in Corollary 11.

Given that \(#(w_0 ^{\alpha_1} w_1 ^{\alpha_2} w_2 \cdots w_{k-2} ^{\alpha_{k-1}} w_{k-1}, y)\) in (1) is between 0 and \( n \), it follows from our choice of \( s \), a standard Chernoff bound and a union bound, that with probability at least \( 1 - \tau \), every empirical estimate \( \tilde{E}_\alpha \) satisfies

\[
|\tilde{E}_\alpha - E_\alpha| \leq \varepsilon \cdot \left( \frac{1 - \delta}{n} \right)^k.
\]

Using

\[
\left| \frac{\xi - \delta}{1 - \delta} \right| \leq 1
\]

when \( \xi \in B_{1-\delta}(\delta) \), we can use \( \tilde{E}_\alpha \) to obtain an estimate of \( SW_{x,w}(\xi) \):

\[
\frac{1}{(1 - \delta)^k} \sum_\alpha \tilde{E}_\alpha \cdot \left( \frac{\xi - \delta}{1 - \delta} \right)^{|\alpha|}
\]

and the estimate is correct up to error

\[
\frac{1}{(1 - \delta)^k} \sum_\alpha |\tilde{E}_\alpha - E_\alpha| \leq \varepsilon,
\]

where the inequality holds by Equation (2) given that the number of \( \alpha \)'s is no more than \( n^k \). \( \square \)

### 5.2 Generalized deletion polynomial and the proof of Theorem 10

In this subsection we prove Theorem 10. We first introduce a more general class of polynomials, the \((x,f)\)-deletion-channel polynomials (see Definition 12), of which \(SW_{x,w}\) is a special case. We then prove an extension of Theorem 10 (see Theorem 13) which applies to every \((x,f)\)-deletion channel polynomial; Theorem 10 follows as a direct corollary. While we don’t need the full generality of Theorem 13 to prove Theorem 10, working with this new class of polynomials makes our proofs cleaner. We also believe that Theorem 13 in the general form may be useful for subsequent analysis.

The following notation will be convenient for us. Given vectors \( \gamma \in \mathbb{Z}^k_{\geq 0} \) and \( \xi \in \mathbb{C}^k \), and a polynomial \( P(z_1, \ldots, z_k) \) from \( \mathbb{C}^k \) to \( \mathbb{C} \), we define

\[
\xi^\gamma = \xi_1^{\gamma_1} \cdots \xi_k^{\gamma_k}
\]

and

\[
D^\gamma P = \frac{\partial |\gamma| P}{\partial z_1^{\gamma_1} \cdots \partial z_k^{\gamma_k}}.
\]

Recall that \( \gamma \# = \gamma_1 \cdots \gamma_k \) and \( |\gamma| = \gamma_1 + \cdots + \gamma_k \). For \( \nu \in \mathbb{C} \), we will denote the vector \( (\nu, \nu, \ldots, \nu) \in \mathbb{C}^k \) by \( \overline{\nu} \), where the dimension \( k \) will be clear from context.

We define the class of \((x,f)\)-deletion-channel polynomials:

**Definition 12.** Given \( f : \{0,1\}^k \to \mathbb{C} \) and a string \( x \in \{0,1\}^n \), the \((x,f)\)-deletion-channel polynomial \( P_{x,f} : \mathbb{C}^k \to \mathbb{C} \) is defined by

\[
P_{x,f}(\xi) := \sum_{\gamma \in \mathbb{Z}^k_{\geq 0}} f(x_{\gamma_1}, x_{\gamma_1+\gamma_2}, \ldots, x_{\gamma_1+\cdots+\gamma_k+(k-1)}) \cdot \xi^\gamma.
\]
We call $P_{x,f}$ the $(x,f)$-deletion-channel polynomial because by choosing $k = 1$ and $f : \{0,1\} \to \{0,1\}$ to be the 1-bit identity function $\text{id}(x) = x$, we have that
\[
P_{x,\text{id}}(\xi) = \sum_{i=0}^{n-1} x_i \xi^i
\]
is the deletion-channel polynomial defined in [DOS17].

The next theorem shows that under a change of variables, the coefficients of $P_{x,f}$ with respect to the new variables can be expressed in terms of the expectation of $f$ over traces of $x$ drawn from the deletion channel. We state it and then show that Theorem 10 follows as a direct corollary.

**Theorem 13.** For any $\xi \in C^k$, we have
\[
P_{x,f}(\xi) = \frac{1}{(1 - \delta)^k} \sum_{\beta \in \mathbb{Z}_{\geq 0}^k, |\beta| \leq n-k} E_{y - \text{Del}_\delta(x)} \left[ f(y_{\beta_1}, \ldots, y_{\beta_k + \ldots + \beta_k + (k-1)}) \right] \cdot \left( \frac{\xi - \delta}{1 - \delta} \right)^{|\beta|}
\]

**Proof of Theorem 10 assuming Theorem 13.** Given $x \in \{0,1\}^n$ and $w \in \{0,1\}^k$ for some $k \in [n]$ as in the statement of Theorem 10, we take $f : \{0,1\}^k \to \{0,1\}$ to be the indicator function of $w$:
\[
f(b_1, b_2, \ldots, b_k) = 1 \iff (b_1, b_2, \ldots, b_k) = w.
\]
Using this $f$ we get the following connection between $SW_{x,w}(\xi)$ and $P_{x,f}(1, \xi, \xi, \ldots, \xi)$:
\[
SW_{x,w}(\xi) = \sum_{\alpha \in \mathbb{Z}_{\geq 0}^k, |\alpha| \leq n-k} \sum_{i=0}^{n-k-|\alpha|} f(x_i, x_i + \alpha_i, x_i + \alpha_i + \alpha_{i+1}, \ldots, x_i + |\alpha| + k - 1) \cdot \xi_i^{|\alpha|}
\]
Applying Theorem 13 on $P_{x,f}(1, \xi, \xi, \ldots, \xi)$, we have
\[
SW_{x,w}(\xi) = \frac{1}{(1 - \delta)^k} \sum_{\alpha \in \mathbb{Z}_{\geq 0}^k, |\alpha| \leq n-k} \sum_{i=0}^{n-k-|\alpha|} E_{y - \text{Del}_\delta(x)} \left[ f(y_i, y_{i+\alpha_i}, \ldots, y_{i+|\alpha| + k - 1}) \right] \cdot \left( \frac{\xi - \delta}{1 - \delta} \right)^{|\alpha|}
\]
where the last step follows by linearity of expectation. This concludes the proof of Theorem 10. □

We now prove Theorem 13. The high-level idea is to relate the expectation of $f$ over traces of $x$ drawn from the deletion channel to partial derivatives of the polynomial $P_{x,f}$ at $\delta$, and then apply Taylor’s expansion to $P_{x,f}$ at the point $\delta$.

**Claim 14.** Let $\beta \in \mathbb{Z}_{\geq 0}^k$ with $|\beta| \leq n-k$. We have
\[
E_{y - \text{Del}_\delta(x)} \left[ f(y_{\beta_1}, \ldots, y_{\beta_k + \ldots + \beta_k + (k-1)}) \right] = (1 - \delta)^k \cdot \frac{(1 - \delta)^{|\beta|}}{|\beta|!} \cdot D^\beta P_{x,f}(\delta).
\]
To get some intuition, consider the special case of \(k = 1\) (so \(P_{x,f}\) is univariate) and \(f = \text{id}\). Then it is straightforward to verify that

\[
E_{y \sim \text{Del}_\delta(x)}[y_0] = (1 - \delta) \sum_{i=0}^{n-1} x_i \delta^i = (1 - \delta) \cdot P_{x,\text{id}}(\delta),
\]

and

\[
E_{y \sim \text{Del}_\delta(x)}[y_1] = (1 - \delta) \sum_{i=1}^{n} x_i (1 - \delta) \delta^{i-1} = (1 - \delta)^2 \sum_{i=1}^{n-1} x_i \delta^{i-1} = (1 - \delta)^2 \cdot D^1 P_{x,\text{id}}(\delta).
\]

**Proof of Claim 14.** For a fixed \(y \in \mathbb{Z}_k^k\) with \(|y| \leq n - k\), we write

\[
y \rightarrow \beta,
\]

or equivalently \((y_1, y_2, \ldots, y_k) \rightarrow (\beta_1, \beta_2, \ldots, \beta_k)\),

to denote the event that the \((y_1, y_1+y_2+1, \ldots, y_1+\cdots+y_k+(k-1))\) positions of \(x\) become the \((\beta_1, \beta_1+\beta_2+1, \ldots, \beta_1+\cdots+\beta_k+(k-1))\) positions of \(y \sim \text{Del}_\delta(x)\) respectively. For this to occur, each of \(x_{y_1}, x_{y_1+y_2+1}, \ldots, x_{y_1+\cdots+y_k+(k-1)}\) must become \(y_{\beta_i}\), exactly \(\beta_i\) out of the \(y_i\) bits between (and including) positions \(y_1 + \cdots + y_{i-1} + i\) and \(y_1 + \cdots + y_i + (i-1)\) of \(x\) must be retained. So, the probability of this event is

\[
Pr[y \rightarrow \beta] = (1 - \delta)^k \prod_{i=1}^{k} \binom{y_i}{\beta_i} (1 - \delta)^{\beta_i} \cdot \delta^{y_i - \beta_i}.
\]

\[
= (1 - \delta)^k \prod_{i=1}^{k} y_i (y_i - 1) \cdot \cdots \cdot (y_i - \beta_i + 1) \cdot \frac{(1 - \delta)^{\beta_i} \cdot \delta^{y_i - \beta_i}}{\beta_i!}.
\]

\[
= (1 - \delta)^k \cdot \frac{(1 - \delta)^{|\beta|}}{\beta!} \cdot \prod_{i=1}^{k} \frac{d^{\beta_i}}{d \delta^{\beta_i}} \delta^{y_i}.
\]

(Equation 3)

As a result, we have that

\[
E_{y \sim \text{Del}_\delta(x)}[f(y_{\beta_1}, \ldots, y_{\beta_1+\cdots+\beta_k+(k-1)})] = \sum_{|y| \leq n-k} f(x_{y_1}, \ldots, x_{y_1+\cdots+y_k+(k-1)}) \cdot Pr[y \rightarrow \beta]
\]

\[
= (1 - \delta)^k \cdot \frac{(1 - \delta)^{|\beta|}}{\beta!} \cdot \prod_{i=1}^{k} \frac{d^{\beta_i}}{d \delta^{\beta_i}} \delta^{y_i}
\]

(Equation 3)

\[
= (1 - \delta)^k \cdot \frac{(1 - \delta)^{|\beta|}}{\beta!} \cdot D^\beta P_{x,f}(\delta).
\]

This finishes the proof of Claim 14. \(\square\)
Proof of Theorem 13. Since $P_{x,f}$ is a polynomial of degree at most $n-k$, applying Taylor’s expansion to $P_{x,f}$ at the point $\delta$ and using Claim 14, we get that

$$(1-\delta)^k \cdot P_{x,f}(\xi) = (1-\delta)^k \sum_{|\beta|\leq n-k} \frac{D^{|\beta|}P_{x,f}(\delta)}{\beta!} \cdot (\xi-\delta)^{|\beta|} \cdot$$

$$= \sum_{|\beta|\leq n-k} E_{y=\text{Del}_2(x)} \left[ f(y_{\beta_1}, \ldots, y_{\beta_k+\ldots+\beta_{k+1}}) \right] \cdot \left( \frac{\xi-\delta}{1-\delta} \right)^{|\beta|}. \quad \blacksquare$$

6 MULTIPOLICY'_large-$\delta$: AN ALGORITHM FOR DELETION RATE $\delta < 1$

In this section we prove a weaker version of Theorem 7, giving an algorithm that works for any deletion rate $\delta < 1$ but has quasipolynomial running time and sample complexity when $k \approx \log n$ (as will be the case in our ultimate application):

**Theorem 15.** Let $0 < \tau, \delta < 1$. There is an algorithm Multiplicity'_large-$\delta$, which takes as input a string $w \in \{0,1\}^k$ and access to independent traces of an unknown source string $x \in \{0,1\}^n$. Multiplicity'_large-$\delta$ runs in $\left( \frac{n^{1/(1-\delta)}}{1-\delta} \right)^{O(k)} \log \left( \frac{1}{\delta} \right)$ time and uses $\left( \frac{n^{1/(1-\delta)}}{1-\delta} \right)^{O(k)} \log \left( \frac{1}{\delta} \right)$ many traces from Del$_2(x)$, and has the following property: For any unknown source string $x \in \{0,1\}^n$, with probability at least $1-\tau$ the output of Multiplicity'_large-$\delta$ is $\#(w,x)$, the multiplicity of $w$ in subword$(x,k)$ (equivalently, the value SW$_{x,w}(0)$).

Looking ahead, in Section 7 we will build on the proof of Theorem 15 to give a stronger version that has polynomial running time and sample complexity when $k = \log n$.

The following result is central to our analysis. Informally, it says that if $q$ is a polynomial with “not-too-large” coefficients and a constant term which is bounded away from SW$_{x,w}(0)$ by at least 1/2, then $q$ must “differ noticeably” from SW$_{x,w}$ over a particular interval. (Looking ahead, for our purposes it is crucially important that this interval corresponds to a range of deletion probabilities for which it is easy to estimate the polynomial’s value given access to traces drawn from Del$_2(x)$.)

**Theorem 16.** Fix strings $x \in \{0,1\}^n$, $w \in \{0,1\}^k$ for some $k \in [n]$. Let $q(z) = \sum_{\ell=0}^{n-k} q_\ell z^\ell$ be any polynomial such that $|\text{SW}_{x,w}(0) - q(0)| \geq 1/2$, and $0 \leq q_\ell \leq n^k$ for all $\ell \in \{0,1,\ldots,n-k\}$. Then

$$\sup_{z \in [\delta, (\delta+1)/2]} |\text{SW}_{x,w}(z) - q(z)| \geq n^{-O(k/(1-\delta))}, \quad \text{for any } \delta \in (0,1). \quad (4)$$

Theorem 16 is an easy consequence of the following more general theorem:

**Theorem 17.** Let $1 \leq n \leq m$. Let $p(z) = \sum_{\ell=0}^{n} p_\ell z^\ell$ be a polynomial of degree at most $n$ with real coefficients such that $|p_0| \geq 1/2$, and $|p_\ell| \leq m$ for all $\ell$. Then we have

$$\sup_{z \in [\delta, (\delta+1)/2]} |p(z)| \geq m^{-O(1/(1-\delta))}, \quad \text{for any } \delta \in (0,1). \quad (5)$$

To obtain Theorem 16 from Theorem 17, set $p = \text{SW}_{x,w} - q$. By the condition of Theorem 16 we have that $|p_0| = |\text{SW}_{x,w}(0) - q(0)| \geq 1/2$. Writing $(\text{SW}_{x,w})_\ell$ for the degree-$\ell$ coefficient of SW$_{x,w}$, from the discussion following Definition 8 it is immediate that $0 \leq (\text{SW}_{x,w})_\ell \leq \left( \frac{n}{k} \right) \leq n^k$, and hence $|p_\ell| = |(\text{SW}_{x,w})_\ell - q_\ell| \leq n^k$. Thus we can invoke Theorem 17 with $m = n^k$ to obtain Theorem 16.

In Section 6.1 we present and analyze the algorithm Multiplicity'_large-$\delta$ (which is based on linear programming) and prove Theorem 15 assuming Theorem 16. The proof of Theorem 17, which is based on complex analysis, is given in Section 6.2.
6.1 Proof of Theorem 15 assuming Theorem 16

6.1.1 Estimating $SW_{x,w}(\delta')$ for $\delta' \geq \delta$. The following easy lemma gives an unbiased estimator for $SW_{x,w}(\delta')$, for all $\delta' \geq \delta$, given traces from Del$_\delta(x)$.

**Lemma 18.** Let $x \in \{0,1\}^n$, $w \in \{0,1\}^k$ and let $\varepsilon > 0$. Then, given traces $y \sim$ Del$_\delta(x)$, there exists an algorithm, which for any $\delta' \in [\delta,1]$, evaluates $SW_{x,w}(\delta')$ up to error $\pm \varepsilon$ with success probability at least $1 - \tau$. The algorithm takes

$$n^{O(1)} \cdot \left(\frac{1}{1 - \delta'}\right)^{O(k)} \cdot \frac{1}{\varepsilon^2} \cdot \log \left(\frac{1}{\tau}\right)$$

many traces and running time.

**Proof.** First of all, observe that given $y \sim$ Del$_\delta(x)$, we can sample $y \sim$ Del$_{\delta'}(x)$ for any $\delta' \geq \delta$ with no overhead. Next, observe that the expected number of $w$ in a random trace $y \sim$ Del$_{\delta'}(x)$ is given by

$$E_{y \sim \text{Del}_{\delta'}(x)}[\#(w,y)] = \sum_{\alpha \in \mathbb{Z}_{\geq 0}^k \atop |\alpha| \leq n-k} \#(w_0 \ast_{\alpha_1} w_1 \ast_{\alpha_2} \cdots \ast_{\alpha_k} w_{k-1}, x) \cdot \delta^{\alpha} \cdot (1 - \delta')^k.$$ 

This follows from the fact that every occurrence of $w$ as a subword of trace can be uniquely identified with a subsequence $(i_1 \leq \cdots \leq i_k)$ such that (i) $x_{i_1} = w_1$, \ldots, $x_{i_k} = w_k$, (ii) positions $i_1, \ldots, i_k$ are not deleted in $y$, (iii) every position in $[i_1 : i_k] \setminus \{i_1, \ldots, i_k\}$ is deleted in the trace $y$. However, by Definition 8, it follows that

$$E_{y \sim \text{Del}_{\delta'}(x)}[\#(w,y)] = SW_{x,w}(\delta') \cdot (1 - \delta')^k.$$

Now for any $y \sim$ Del$_{\delta'}(x)$, $\#(w,y)$ is an integer between 0 and $n$. Thus, the usual analysis of the standard empirical estimator via a Chernoff bound will use

$$n^{O(1)} \cdot \left(\frac{1}{1 - \delta'}\right)^{O(k)} \cdot \frac{1}{\varepsilon^2} \cdot \log \left(\frac{1}{\tau}\right)$$

many traces and running time and returns an estimate of $E_{y \sim \text{Del}_{\delta'}(x)}[\#(w,y)]$ up to $\pm \varepsilon \cdot (1 - \delta')^k$. Using (6), we get the claim. \hfill \Box

6.1.2 The Multiplicity'$_{\text{large-}\delta}$ algorithm and its analysis. We present the algorithm Multiplicity'$_{\text{large-}\delta}$ in Figure 1. For its correctness we will take enough traces so that with probability at least $1 - \tau$, we have that

for every $\zeta \in S$, 

$$|\bar{SW}_{x,w}(\zeta) - SW_{x,w}(\zeta)| \leq \kappa/5.$$ 

We finish the proof by showing that when this happens, the linear program in lines 3(a) and 3(b) is feasible, and furthermore, $|q_0 - SW_{x,w}(0)| < 1/2$ in any feasible solution $(q_0, \ldots, q_{n-k})$ (when this happens, the closest integer to $q_0$ is exactly $SW_{x,w}(0)$).

To see that the linear program is feasible, we let $p_0, \ldots, p_{n-k}$ denote the coefficients of $SW_{x,w}$, so $SW_{x,w}(\zeta) = \sum_{\ell=0}^{n-k} p_{\ell} \zeta^\ell$. From the discussion after Theorem 17, every $p_{\ell}$ lies between 0 and $n^k$. As a result, $p_0, \ldots, p_{n-k}$ is a feasible solution to the linear program because for every $\zeta \in S$,

$$\left|\sum_{\ell=0}^{n-k} p_{\ell} \zeta^\ell - \bar{SW}_{x,w}(\zeta)\right| = \left|SW_{x,w}(\zeta) - \bar{SW}_{x,w}(\zeta)\right| \leq \kappa/5.$$ 

Next we let $q_0, \ldots, q_{n-k}$ be any feasible solution to the linear program and assume for a contradiction that $|q_0 - SW_{x,w}(0)| \geq 1/2$. Let $q$ be the polynomial $q(\zeta) = \sum_{\ell=0}^{n-k} q_\ell \zeta^\ell$. Given that $0 \leq q_\ell \leq n^k$ for every $\ell$ (as required

ACM Trans. Algor.
The following claim (with proof for each $\ell$ coefficient $\kappa$ by the linear program), Theorem 16 implies (using the choice of $\kappa'$) that

$$\sup_{\zeta \in [\delta, (\delta + 1)/2]} \left| \sum_{\ell=0}^{n-k} q_\ell \xi^\ell - SW_{x, \omega}(\zeta) - q(\zeta) \right| \geq \kappa. \tag{7}$$

The following claim (with $s = SW_{x, \omega} - q$ and $m = n^k$) shows that there exists a $\zeta \in S$ such that

$$\left| \sum_{\ell=0}^{n-k} q_\ell \xi^\ell - SW_{x, \omega}(\zeta) \right| \geq \kappa/5,$$

a contradiction to the assumption that $q_0, \ldots, q_{n-k}$ is a feasible solution because

$$\left| \sum_{\ell=0}^{n-k} q_\ell \xi^\ell - SW_{x, \omega}(\zeta) \right| = \left| q(\zeta) - \sum_{\ell=0}^{n-k} q_\ell \xi^\ell \right| \geq \left| q(\zeta) - SW_{x, \omega}(\zeta) \right| - \left| SW_{x, \omega}(\zeta) - \sum_{\ell=0}^{n-k} q_\ell \xi^\ell \right| > \kappa/5.$$

Claim 19 (Searching over $S$ suffices). Let $s(t) = s_0 + s_1 t + \cdots + s_n t^n$ be a polynomial such that every coefficient $s_\ell$ has $|s_\ell| \leq m$. Suppose $|s(t_0)| \geq \kappa$ for some $t_0 \in [\delta, (\delta + 1)/2]$. Then there exists an integer $j$ such that

$$t' = \delta + j \Delta \in [\delta, (\delta + 1)/2] \text{ and } |s(t')| \geq \kappa/2,$$

where $\Delta = \kappa/(2mn^2)$.

Proof. Let $j$ be an integer such that $t' := \delta + j \Delta \in [\delta, (\delta + 1)/2]$ and $|t' - t_0| \leq \Delta$. Since $|t_0| \leq 1$ and $|t'| \leq 1$, for each $\ell \in \{1, \ldots, n\}$ we have that

$$|t'^\ell - t_0^\ell| \leq |t' - t_0| \cdot \sum_{i=0}^{\ell-1} |t'^i t_0^{\ell-1-i}| \leq \Delta \ell \leq \Delta n.$$
Since $|s_\ell| \leq m$ and $\Delta = \kappa/(2mn^2)$, we have

$$|s_\ell t - s_\ell t_0| = |s_\ell| \cdot |t' - t_0| \leq mn\Delta = \kappa/(2n).$$

Therefore

$$|s(t') - s(t_0)| \leq \sum_{\ell=1}^n |s_\ell t - s_\ell t_0| \leq \kappa/2.$$

It follows from the triangle inequality that $|s(t')| \geq |s(t_0)| - |s(t') - s(t_0)| \geq \kappa/2$. □

We now analyze the complexity of the algorithm. Note that for all $\zeta \in S$, we have $1 - \zeta \geq (1 - \delta)/2$. By Lemma 18, the sample complexity is

$$n^{O(1)} \cdot \left(\frac{2}{1 - \delta}\right)^{O(k)} \cdot \left(\frac{5}{\kappa}\right)^2 \cdot \log \left(\frac{|S|}{\tau}\right) = \left(\frac{n^{1/(1 - \delta)}}{1 - \delta}\right)^{O(k)} \cdot \log \left(\frac{1}{\tau}\right). \tag{8}$$

The running time of the algorithm is (8) multiplied by $|S|$ plus the time needed to solve the linear program. The former can still be bounded by the same expression on the RHS of (8) above. The latter can be bounded by poly(n) multiplied by the number of bits needed to describe the linear program, which can also be bounded by the RHS of (8). This proves the claimed upper bounds on the running time and sample complexity, and concludes the proof of Theorem 15 assuming Theorem 16.

6.2 Proof of Theorem 17

In this subsection we prove Theorem 17. As mentioned earlier, the proof is inspired by similar proofs in [BE97, BEK99], in particular the proof of Theorem 5.1 of [BEK99].

For convenience we define $\rho := 1 - \delta \in (0, 1)$, and we restate the theorem below in terms of $\rho$:

**Restatement of Theorem 17**: Let $1 \leq n \leq m$. Let $p(z) = \sum_{l=0}^n p_l z^l$ be a polynomial of degree at most $n$ with real coefficients such that $|p_0| \geq 1/2$, and $|p_l| \leq m$ for all $l$. Then for any $\rho \in (0, 1)$,

$$\sup_{\zeta \in [1 - \rho, 1 - \rho/2]} |p(\zeta)| \geq m^{-O(1/\rho)}.$$

The proof uses the Hadamard three-circle theorem, along with other standard results in complex analysis. Consider the mapping $w: \mathbb{C} \to \mathbb{C}$ given by

$$w(z) = 1 - \frac{3\rho}{4} + \frac{\rho}{8} \left(z + \frac{1}{z}\right).$$

We observe that the map $w(z)$ is meromorphic with only one pole at $z = 0$. Define radii

$$r_1 = 1; \quad r_2 = 2; \quad r_3 = 4.$$

For $i = 1, 2, 3$, let $C_i \subset \mathbb{C}$ be the circle centered at the origin with radius $r_i$. Consider the map $f: \mathbb{C} \to \mathbb{C}$ given by $f(z) = p(w(z))$. Like $w(\cdot)$, $f$ is meromorphic with only one pole at $z = 0$. The idea of the proof is to use the Hadamard three-circle theorem (see [Wik20a] or [Ull08], pp. 386-387) on $f$, which tells us that

$$2\log \left(\sup_{z \in C_2} |f(z)|\right) \leq \log \left(\sup_{z \in C_1} |f(z)|\right) + \log \left(\sup_{z \in C_3} |f(z)|\right). \tag{9}$$

Now, we will analyze each term in the above inequality. We first record some facts about the behaviour of $w$ over each circle $C_i$ that are immediate from the definition and a routine calculation:

**Fact 20.** Let $w, C_1, C_2$ and $C_3$ be as defined above.

1. When $z$ ranges over $C_1$, $w(z)$ ranges over the real line segment $[1 - \rho, 1 - \rho/2]$. ACMTTrans. Algor.
(2) When $z$ ranges over $C_2$, $w(z)$ ranges over the ellipse $E_2$ in the complex plane which is centered at the real value $1 - 3\rho/4$ and is the locus of all points $z = x + iy$ satisfying
\[
\left(\frac{x - (1 - 3\rho/4)}{5\rho/16}\right)^2 + \left(\frac{y}{3\rho/16}\right)^2 = 1.
\]

(3) Similarly, when $z$ ranges over $C_3$, $w(z)$ ranges over the ellipse $E_3$ in the complex plane which is centered at the real value $1 - 3\rho/4$ and is the locus of all points $z = x + iy$ satisfying
\[
\left(\frac{x - (1 - 3\rho/4)}{17\rho/32}\right)^2 + \left(\frac{y}{15\rho/32}\right)^2 = 1.
\]

Moreover, the ellipse $E_3$ is completely contained in the unit disk $B_1(0)$.

Equation (9) will be useful to us because of the following simple claim, which is immediate from Fact 20, Item (1):

**Claim 21.**
\[
\sup_{z \in C_1} |f(z)| = \sup_{\zeta \in [1-\rho,1-\rho/2]} |p(\zeta)|.
\]

Given Equation (9) and Claim 21, in order to lower bound $\sup_{z \in C_1} |f(z)|$ and to lower bound $\sup_{z \in C_2} |f(z)|$, it suffices to upper bound $\sup_{z \in C_2} |f(z)|$ and to lower bound $\sup_{z \in C_1} |f(z)|$. We do this in the following claims:

**Claim 22.**
\[
\sup_{z \in C_2} |f(z)| \leq m \cdot (n + 1).
\]

**Proof.** By Fact 20, Item (3) above, we have $E_2 \subseteq B_1(0)$ and so
\[
\sup_{z \in C_2} |f(z)| = \sup_{z \in E_2} |f(z)| \leq \sup_{z \in B_1(0)} |p(z)|.
\]

The bounds on the coefficients of $p$ immediately imply that $\sup_{z \in B_1(0)} |p(z)| \leq m \cdot (n + 1)$. \(\square\)

**Claim 23.**
\[
\sup_{z \in C_1} |f(z)| \geq m^{-O(1/\rho)}.
\]

**Proof.** Applying Jensen’s formula (see e.g. [Wik20b] or [Ahl79], pp. 207-208) to $p$ on the closed origin-centered disk of radius $1 - 3\rho/4$, we get that
\[
E_2 \left[ \ln |p(z)| \right] \geq \ln |p(0)| \geq \ln(1/2) = -\ln 2.
\]

(10)

Here $z$ is taken to be a uniform random point on the circle $C$ of radius $1 - 3\rho/4$ centered at the origin.

Now, consider the arc
\[
\mathcal{A} = \{ z \in C : |z| = 1 - 3\rho/4 \text{ and } \arg(z) \leq 3\rho/16 \}.
\]

Let $c_{max, \mathcal{A}} \equiv \max_{z \in \mathcal{A}} |p(z)|$ and $\theta^* = 3\rho/16$ (note that $\theta^*/\pi$ is the fraction of $C$ that lies in $\mathcal{A}$). Now since $|p(z)| \leq m(n+1)$ for all $z \in B_{1-3\rho/4}(0) \setminus \mathcal{A}$ (because of the coefficient bound on $p$), we have by Equation (10) that
\[
-\ln 2 \leq \left( 1 - \frac{\theta^*}{\pi} \right) \ln(m(n+1)) + \frac{\theta^*}{\pi} \ln c_{max, \mathcal{A}} \leq \ln(m(n+1)) + \frac{\theta^*}{\pi} \ln c_{max, \mathcal{A}}.
\]

Thus,
\[
\ln c_{max, \mathcal{A}} \geq \frac{-\pi \cdot \ln(2m(n+1))}{\theta^*},
\]

and hence
\[
c_{max, \mathcal{A}} \geq (2m(n+1))^{-\pi/\theta^*}.
\]
Next, we observe that the arc $\mathcal{A}$ is entirely in the interior of the ellipse $E_2$. (To see this, observe that the center of the arc is the real value $1 - 3\rho/4$, which coincides with the center of the ellipse, and that every point on the arc is within distance less than $3\rho/16$ from the center of the arc (ellipse). Since $3\rho/16$ is the length of the semi-minor axis of the ellipse, it follows that every point in the arc is within the ellipse.) We further recall that $m \geq n$ and that $\theta^* = \Theta(\rho)$. Using these facts along with the maximum modulus principle (see Section 3.4 of [Ahl79]) and Fact 20 Item (2), we conclude that

$$\sup_{z \in C_1} |f(z)| = \sup_{z \in E_2} |p(z)| \geq \sup_{z \in \mathcal{A}} |p(z)| = c_{\max, \mathcal{A}} \geq m^{-O(1/\rho)},$$

and Claim 23 is proved. \hfill \Box

**Proof of Theorem 17.** We combine Claims 21, 22 and 23 in Equation (9) to get that

$$\log \sup_{\xi \in [1-\rho,1-\rho/2]} |f(\xi)| = \log \sup_{z \in C_1} |f(z)| \geq -O(1/\rho) \log m - \log(m(n+1)) \geq -O(1/\rho) \log m.$$  

Exponentiating both sides finishes the proof of Theorem 17. \hfill \Box

### 7 IMPROVED ALGORITHMS: PROOF OF Theorem 7

In this section we give improved algorithms strengthening the quantitative bounds given in Theorem 9 and Theorem 15 and thereby complete the proof of Theorem 7.

First we describe the main ideas underlying the improved algorithms. Both algorithms benefit from the same insights, so we will just describe the improvement of Theorem 15 in this overview. Recall the definition of the subword polynomial $SW_{x,w}$ from Definition 8:

$$SW_{x,w}(\xi) := \sum_{a \in Z_{n-k}^{k-1}} \# \left( w_0 * a_1^w w_1 * a_2^w w_2 \ldots w_{k-2} * a_{k-1}^w w_{k-1}, x \right) \cdot \xi^{|a|}.$$

Grouping terms of the same degree together, we can write it as $SW_{x,w}(\xi) = \sum_{\ell \geq 0} \gamma_{\ell} \xi^\ell$, where

$$\gamma_{\ell} = \sum_{a \in Z_{n-k}^{k-1}} \# \left( w_0 * a_1^w w_1 * a_2^w w_2 \ldots w_{k-2} * a_{k-1}^w w_{k-1}, x \right)$$

is the degree-$\ell$ coefficient, for each $0 \leq \ell \leq n-k$. In the proofs of Corollary 11 in Section 5 and Theorem 16 in Section 6, we bounded these coefficients uniformly by $m = n^k$. The first insight is that in fact a sharper bound holds for these coefficients: specifically, we have

$$0 \leq \gamma_{\ell} \leq m_{\ell} := n \binom{\ell + k - 2}{k - 2}. \quad (11)$$

This is simply because there are at most $n$ choices for the position of the first character $w_0$ in $x$, and there are $\binom{\ell + k - 2}{k - 2}$ ways to choose a tuple of non-negative integers $a_1, \ldots, a_{k-1}$ that sum to $\ell$. The second insight is that since our approaches only involve evaluating $SW_{x,w}(\xi)$ on non-negative real inputs $\xi$ that are bounded below 1, we can exploit this improved coefficient bound to *truncate* the high-degree portion of the polynomial; working with the resulting (much) lower-degree polynomial leads to an overall gain in efficiency.

To explain this in more detail, we need the following definition:
Definition 24. Let \( p(\zeta) = \sum_{\ell = 0}^{n} p_{\ell} \zeta^{\ell} \) be a univariate polynomial of degree at most \( n \). For \( d \in \{0, 1, \cdots, n\} \), we define the \( d \)-low-degree part of \( p \) (denoted as \( p^{d} \)) to be
\[
p^{d}(\zeta) = \sum_{\ell = 0}^{d} p_{\ell} \zeta^{\ell}.
\]

Analogously, we define the \( d \)-high-degree part of \( p \) to be
\[
p^{>d}(\zeta) := \sum_{\ell > d} p_{\ell} \zeta^{\ell} = p(\zeta) - p^{d}(\zeta).
\]

Consider any polynomial \( q \) with a constant term which is an integer different from \( \text{SW}_{x, w}(0) \). In order for \( q \) to be a polynomial that could possibly arise from the \( k \)-subword deck of some string \( z \in \{0, 1\}^{n} \), it must also have coefficients bounded by the right hand side of Equation (11). Using these sharper bounds on the coefficients, we show that there exists a threshold degree \( d \) that is roughly\(^2\) \( O(k + \log n) \) such that
- The \( d \)-low-degree part of the polynomials \( \text{SW}_{x, w} \) and \( q \) must differ by at least
\[
\left( \frac{1}{n} \left( \frac{1 - \delta}{2} \right) \right)^{O(1/(1-\delta))}
\]
(see Equation (17)) at some point in the interval \([\delta, (\delta + 1)/2] \). This result is stronger than the analogous \( n^{-O(k/(1-\delta))} \) lower bound established in Theorem 16, which leads to savings on both time and sample complexity.
- The maximum value that the high-degree part of such polynomials attains on the relevant interval is negligible compared to the difference specified above.

Combining these two facts enables us to carry out our analysis just on the \( d \)-low-degree part, which has much smaller coefficients and thereby admits a more efficient algorithm.

In Section 7.1, we implement these ideas to strengthen Theorem 9 when \( \delta < 1/2 \). In Section 7.2, we do the same to derive a stronger analogue of Theorem 16, which reduces the sample complexity of computing \( \#(w, x) \) for general \( \delta < 1 \) significantly. Finally in Section 7.3, we obtain an LP-based algorithm to compute \( \#(w, x) \) which is faster than the corresponding algorithm in Section 6.1.

7.1 Improvement of Theorem 9 for deletion rate \( \delta < 1/2 \)

In this subsection we strengthen Theorem 9 for deletion rate \( \delta < 1/2 \) as follows:

Theorem 25. Let \( 0 < \delta < 1/2 \). There is an algorithm \text{Multiplicity}_{\text{small-} \delta} \ which takes as input a string \( w \in \{0, 1\}^{k} \), access to independent traces of an unknown source string \( x \in \{0, 1\}^{n} \), and a parameter \( \tau > 0 \). \text{Multiplicity}_{\text{small-} \delta} \ draws \( \text{poly}(n) \cdot (1/2 - \delta)^{-O(k)} \cdot \log(1/\tau) \) traces from \( \text{Del}_{\delta}(x) \), runs in time \( \text{poly}(n) \cdot (1/2 - \delta)^{-O(k)} \cdot \log(1/\tau) \), and has the following property: For any unknown source string \( x \in \{0, 1\}^{n} \), with probability at least \( 1 - \tau \) the output of \text{Multiplicity}_{\text{small-} \delta} \ is the multiplicity of \( w \) in \( \text{subword}(x, k) \) (i.e. the number of occurrences of \( w \) as a subword of \( x \)).

Recall Theorem 10, which relates the subword polynomial value at any point \( \zeta \in \mathbb{C} \) to traces drawn from the deletion channel using Taylor series:
\[
\text{SW}_{x, w}(\zeta) = \frac{1}{(1 - \delta)^{k}} \sum_{\alpha \in \mathbb{Z}_{+}^{k-1}} \sum_{\text{tau}_{\alpha} \leq n} E_{y = \text{Del}_{\delta}(x)} \left[ \#(w_{0} w_{1}^{\alpha_{1}} w_{2}^{\alpha_{2}} \cdots w_{k-2}^{\alpha_{k-2}} w_{k-1}^{\alpha_{k-1}} y) \right] \cdot \left( \frac{\zeta - \delta}{1 - \delta} \right)^{|\alpha|}.
\]

\(^2\)We ignore the dependence on \( \delta \) for the overview here; see (12) and (16) for exact choices of \( d \).
As in Section 7.1, our goal is to evaluate \( SW_{x,w}(0) = #(w,x) \) up to error 1/3 in magnitude, and return the integer nearest to our estimate. Let \( \xi = (\zeta - \delta)/(1 - \delta) \), so that \( \zeta = \delta + \xi(1 - \delta) \). Consider the polynomial \( p \) defined as follows:

\[
p(\xi) := (1 - \delta)^k \cdot SW_{x,w}(\delta + \xi(1 - \delta)).
\]

We have that \( SW_{x,w}(0) = (1 - \delta)^{-k} p(-\delta/(1 - \delta)) \), so estimating \( SW_{x,w}(0) \) up to error \( \pm 1/3 \) is equivalent to estimating \( p(-\delta/(1 - \delta)) \) up to error \( \pm 1/3 \). As \( 0 < \delta < 1/2 \), we have \( 1 - \delta > 1/2 \), and so it suffices to estimate \( p(-\delta/(1 - \delta)) \) up to error \( 2^{-k}/3 \). Moreover, we have \( |\delta/(1 - \delta)| = \delta/(1 - \delta) < 1 \). We will use these observations to bound the contribution of the high-degree-part of \( p \).

**Lemma 26.** Let \( \delta < 1/2 \), and let \( p \) and \( \theta \) be as above. Then by setting

\[
d := \frac{C}{\theta} \left( k \ln \frac{C}{\theta} + \ln n \right)
\]

with \( C = e^2 \), we have

\[
\sup_{|\xi| \leq 1 - 2\theta} |p^d(\xi)| \leq \frac{0.1}{2^k}.
\]

Before proving Lemma 26, we show that it implies Theorem 25.

**Proof of Theorem 25 assuming Lemma 26.** Consider \( p^d \), the \( d \)-low-degree-part of \( p \), where \( d \) is as given by Lemma 26. For all \( \xi \) with \( |\xi| \leq 1 - 2\theta \),

\[
|p(\xi) - p^d(\xi)| = |p^d(\xi)| \leq \frac{0.1}{2^k}.
\]

So, by the triangle inequality, in order to estimate \( p(-\delta/(1 - \delta)) \) up to error \( \pm 2^{-k}/3 \), it suffices to estimate \( p^d(-\delta/(1 - \delta)) \) up to error \( \pm 2^{-k}/5 \).

Let \( S_d \) be the set \( \{ \alpha \in \mathbb{Z}^{k-1}_{\geq 0} : |\alpha| \leq d \} \). As in Section 5.1, let

\[
E_{\alpha} := \mathbb{E}_{y \sim \text{Del}_k(x)} \left[ #(w_0 \ast \alpha_1 w_1 \ast \alpha_2 w_2 \cdots w_{k-2} \ast \alpha_{k-1} w_{k-1}, y) \right]
\]

for each \( \alpha \in S_d \). (Note that by definition, \( p^d \) only includes terms \( E_{\alpha} \) for \( |\alpha| \leq d \).) Then

\[
p^d(\xi) = \sum_{\alpha \in S_d} E_{\alpha} \cdot \xi^{|\alpha|}.
\]

Each \( E_{\alpha} \) is between 0 and \( n \) and using the same argument as that following Equation (11), we have

\[
|S_d| = M := \sum_{\ell=0}^{d} \binom{(d + k - 1)}{k - 2} = \binom{(d + k - 1)}{k - 1} \leq \binom{(d + k)}{k}
\]

and we use the following claim to bound the right hand side:

**Claim 27.** Let \( d = \frac{C}{\theta} (k \ln \frac{C}{\theta} + \ln n) \) for some \( \theta \in (0, 1] \) and \( C \geq e^2 \). Then we have

\[
\binom{(d + k)}{k} \leq n \cdot \left( \frac{C}{\theta} \right)^{3k}.
\]

ACM Trans. Algor.
We first show that terms in the sum on the right hand side above decreases with \( \ell \). By Claim 27, we have
\[
\sum_{\alpha \in S_d} E_{\alpha} = \left( \frac{\theta}{k} \right)^{O(k)} \cdot \log \left( \frac{M}{\tau} \right) = \left( \frac{n}{\theta k} \right)^{O(1)} \cdot \log \frac{1}{\tau},
\]
and a simple application of a Chernoff bound and a union bound. When this happens, it follows from the fact that \( | - \delta / (1 - \delta) | < 1 \) that
\[
\sum_{\alpha \in S_d} \hat{E}_{\alpha} \cdot \left( \frac{-\delta}{1-\delta} \right)^{|w|} \leq \left( \frac{n}{\theta k} \right)^{O(1)} \cdot \log \frac{1}{\tau} = n^{O(1)} \cdot \left( \frac{1/2 - \delta}{1/2 - \delta} \right)^{O(k)} \cdot \log \frac{1}{\tau}.
\]
This finishes the proof of the theorem. □

**Proof of Lemma 26.** We are interested in \(|p^{\geq d}(\xi)| \) over \(|\xi| \leq 1 - 2\theta\), which is trivially bounded by
\[
|p^{\geq d}(\xi)| \leq \sum_{\ell = d + 1}^{n-k} n \binom{\ell + k - 2}{k-2} \cdot (1 - 2\theta)^\ell \leq \sum_{\ell = d-1}^{n-k} n \binom{\ell + k}{k} \cdot (1 - 2\theta)^\ell.
\]
First, we show that terms in the sum on the right hand side above decreases with \( \ell \) so it suffices to bound the term with \( \ell = d \) multiplied by \( n \). To see this, observe that
\[
\left| \binom{\ell + k}{k} \right| \cdot (1 - 2\theta)^\ell = \frac{\ell + k}{\ell} \cdot (1 - 2\theta)^\ell \leq 1 + \frac{k}{\ell} - 2\theta < 1,
\]
whenever \( \ell > k/2\theta \), which holds for all \( \ell > d \) given our choice of \( d \). So,
\[
\sup_{|\xi| \leq 1 - 2\theta} |p^{\geq d}(\xi)| \leq n^2 \binom{d+k}{k} (1 - 2\theta)^d \leq n^2 \binom{d+k}{k} e^{-2\theta d}.
\]
We have $e^{-2\theta d} = n^{-2C} \cdot (C/\theta)^{-2Ck}$, and so plugging in Claim 27 we have

$$n^2 \cdot \left( n \cdot \left( \frac{C}{\theta} \right)^{3k} \right) \cdot e^{-2\theta d} \leq n^{3-2C} \cdot \left( \frac{C}{\theta} \right)^{(3-2C)k} \leq \frac{1}{n^{2k}}$$

because $3 - 2C \leq -1$ when $C = e^2$. This concludes the proof of the lemma.

\[ \square \]

7.2 Improvement of Theorem 16 for deletion rate $\delta < 1$

Our main technical result is the following, which is a strengthening of Theorem 16:

**Theorem 28.** Fix $x \in \{0, 1\}^n$ and $w \in \{0, 1\}^k$ with $k \leq n$. Let $q(z) = \sum_{\ell=0}^{n-k} q_{\ell} z^\ell$ be any polynomial such that $|SW_{x,w}(0) - q(0)| \geq 1/2$ and $0 \leq q_{\ell} \leq m_{\ell}$ for all $\ell \in \{0, 1, \cdots, n-k\}$. Then

$$\sup_{\zeta \in [\delta, (\delta+1)/2]} |SW_{x,w}(\zeta) - q(\zeta)| \geq \left( \frac{1}{n} \left( \frac{1-\delta}{2} \right)^k \right)^{O(1/(1-\delta))}, \text{ for any } \delta \in (0, 1).$$

Let $p(z) = SW_{x,w}(z) - q(z) = \sum_{\ell=0}^{n-k} p_{\ell} z^\ell$. Let $c > 0$ be the constant hidden in the exponent of the RHS of Equation (5) in Theorem 17. Let $\theta = (1-\delta)^2/2$. We will choose the threshold on the degree to be

$$d := \frac{C}{\theta} \left( k \ln \frac{C}{\theta} + \ln n \right)$$

where $C = e^2 \max(1, c)$. For this $d$, consider the $d$-low-degree part $p^{d}$. This is a polynomial of degree at most $d$ with $|p^{d}(0)| \geq 1/2$ and the degree-$\ell$ coefficient is bounded by

$$|p_{\ell}^{d}| \leq n \binom{\ell + k - 2}{k - 2} \leq n \binom{d + k - 2}{k - 2} \leq n \binom{d + k}{k}$$

for all $\ell \leq d$. We invoke Theorem 17 on $p^{d}$ to conclude that

$$\sup_{\zeta \in [\delta, (\delta+1)/2]} |p^{d}(\zeta)| \geq \left( n \binom{d + k}{k} \right)^{-c/(1-\delta)}.$$  \hspace{1cm} (17)

The following lemma upper bounds the contribution of the high-degree part $p^{>d}$ of $p$:

**Lemma 29.** Let $p$ and $d$ be as above. Then

$$\sup_{\zeta \in [\delta, (\delta+1)/2]} |p^{>d}(\zeta)| \leq \frac{1}{n} \left( n \binom{d + k}{k} \right)^{-c/(1-\delta)}.$$  \hspace{1cm} (18)

Before proving this lemma, we show that it implies Theorem 28.

**Proof of Theorem 28 using Lemma 29.** Since $p = p^{d} + p^{>d}$, we use Lemma 29 and (17) to get

$$\sup_{\zeta \in [\delta, (\delta+1)/2]} |p(\zeta)| \geq 0.9 \cdot \left( n \binom{d + k}{k} \right)^{-c/(1-\delta)}.$$

Plugging in Claim 27 with our choice of $d$, we have

$$\sup_{\zeta \in [\delta, (\delta+1)/2]} |p(\zeta)| \geq 0.9 \left( n \binom{d + k}{k} \right)^{-c/(1-\delta)} \geq \left( \frac{1}{n} \left( \frac{1-\delta}{2} \right)^k \right)^{O(1/(1-\delta))},$$

which concludes the proof of Theorem 28 using Lemma 29.  \hspace{1cm} \square

ACM Trans. Algor.
Theorem 28 in place of Theorem 16, the algorithm 

\[ |(\ell+k)\xi| = \frac{\ell+k}{\ell} \cdot |\xi| \leq \left(1 + \frac{k}{\ell}\right) \left(1 - \frac{1 - \delta}{2}\right) \leq 1 + \frac{k}{\ell} - \frac{1 - \delta}{2} < 1 \]

whenever \( \ell > 2k/(1 - \delta) \), which holds for all \( \ell > d \). So,

\[
\sup_{|\xi| \leq (\delta+1/2)} |p^{<d}(\xi)| \leq n^2 \left(\frac{d+k}{k}\right)^{1 - \delta/2} \exp\left(-\frac{(1 - \delta)d}{2}\right) \leq n^2 \left(\frac{d+k}{k}\right)^{1 - \delta/2} \exp\left(-\frac{(1 - \delta)d}{2}\right) \leq 1.
\]

By our choice of \( d \) we have

\[
\exp\left(-\frac{(1 - \delta)d}{2}\right) \leq n^{\frac{C}{1 - \delta}} \cdot \left(C/\theta\right)^{k^c \frac{2c}{1 - \delta}}.
\]

Using Claim 27 again, the left hand side of Equation (19) is at most

\[
n^{3 + \frac{2c}{1 - \delta}} \cdot \left(C/\theta\right)^{k^c \frac{2c}{1 - \delta}} \leq 1
\]

because \( 3 + \frac{3c}{1 - \delta} - \frac{C}{1 - \delta} \leq 0 \) when \( C = e^2 \max(1, c) \). This concludes the proof of the lemma. \( \square \)

7.3 The algorithm of Theorem 7

Armed with Theorem 28 in place of Theorem 16, the algorithm \( \text{Multiplicity}_{\text{large-}\delta} \) giving Theorem 7 and its analysis are very similar to the algorithm \( \text{Multiplicity}_{\text{large-}\delta} \) and its analysis given earlier in Section 6.1; we only indicate the differences here.

The algorithm changes in the following ways:

- In Line 1 of the algorithm, we now set \( \kappa \) to be the RHS of Equation (15):

\[
\kappa := \left(1 + \frac{k}{\ell}\right)^{O(1/(1 - \delta))}.
\]

With this choice of \( \kappa \), it follows from the proof of Theorem 28 that the RHS of Equation (18) in Lemma 29 can be bounded from above by 0.01\( \kappa \).

- Later in Line 1, we now set

\[
\Lambda := \frac{\kappa}{2d^2 m_d} = \frac{\kappa}{2d^2 \cdot n^\left(d+k-2\right)}.
\]

where \( d \) is as given in Equation (16) (the idea is that now we are using the sharper coefficient bound \( m_\ell \leq m_d \) given by Equation (11) rather than the cruder \( n^d \) bound used earlier).

- The coefficient bound on \( q_0, \ldots, q_{n-k} \) in Line 3(a) for the linear program is now \( q_\ell \in [0, m_\ell] \) for all \( \ell \in \{0, 1, \cdots, n-k\} \) rather than \( q_0, \ldots, q_{n-k} \in [0, n^k] \) as earlier.
With these changes to the algorithm, most of the analysis goes through unchanged. As before, we observe that with probability at least $1 - \tau$, we have

$$\text{for every } \zeta \in S, \quad |\hat{\text{SW}}_{x,w}(\zeta) - \text{SW}_{x,w}(\zeta)| \leq \kappa/5.$$  

We assume this happens henceforth. The solution which sets $q_\ell = (\text{SW}_{x,w})_\ell$, the degree-$\ell$ coefficient of $\text{SW}_{x,w}$, for all $\ell$, is clearly feasible.

Now we show that every feasible solution $q_0, \cdots, q_{n-k}$ to the linear program must satisfy $|q_0 - \text{SW}_{x,w}(0)| < 1/2$; this is the only part of the analysis that is somewhat different. Suppose for a contradiction that $q_0, \cdots, q_{n-k}$ is a feasible solution with $|q_0 - \text{SW}_{x,w}(0)| \geq 1/2$. Let $q(\zeta) = \sum_\ell q_\ell \zeta^\ell$ and define the polynomial $p = \text{SW}_{x,w} - q$, with coefficients $p_\ell$. We invoke Theorem 28 to get that $|p(\zeta^*)| \geq \kappa$ for some $\zeta^* \in [\delta, (\delta + 1)/2]$. By Lemma 29 (and the remark below the choice of $\kappa$),

$$|p(\zeta) - p^{\leq d}(\zeta)| = |p^{> d}(\zeta)| \leq 0.01\kappa \tag{20}$$

for all $\zeta \in [\delta, (\delta + 1)/2]$. As a result, we have $|p^{\leq d}(\zeta^*)| \geq 0.99\kappa$. Applying Claim 19 with $s = p^{\leq d}$, $n = d$, $t_0 = \zeta^*$, $m = m_d$, and our choice of $\Lambda$, there exists a $\zeta^* \in S$ such that $|p^{\leq d}(\zeta^*)| \geq 0.995\kappa$ and thus, $|p(\zeta^*)| \geq |p^{\leq d}(\zeta^*)| - |p^{> d}(\zeta^*)| \geq 0.485\kappa$. Hence, recalling that $p = \text{SW}_{x,w} - q$, we have

$$|\hat{\text{SW}}_{x,w}(\zeta^*) - q(\zeta^*)| \geq |p(\zeta^*) - \hat{\text{SW}}_{x,w}(\zeta^*)| \geq 0.285\kappa > \kappa/5.$$  

As $\zeta^* \in S$, the solution $q$ violates a constraint of the LP. This concludes the proof of correctness.

Now we analyze the sample complexity of the algorithm. We have

$$|S| = O(1/\Delta) = \left( n \left(\frac{2}{1-\delta}\right)^k \right)^{O(1/(1-\delta))},$$

using the bounds established in Section 7.2. Moreover, all points $\zeta \in S$ satisfy $1 - \zeta \geq (1 - \delta)/2$. So, by Lemma 18, the sample complexity is at most

$$s = \frac{n^{O(1)}}{\kappa^2} \left(\frac{2}{1-\delta}\right)^{O(k)} \log \left(\frac{|S|}{\tau}\right) = \left( n \left(\frac{2}{1-\delta}\right)^k \right)^{O(1/(1-\delta))} \log \frac{1}{\tau}. \tag{21}$$

The running time is dominated by the time required to compute $\hat{\text{SW}}_{x,w}(\zeta)$ for each $\zeta \in S$. The running time for each $\zeta$ can be bounded by (21) and the same expression can be used to bound the overall running time given the bound on $|S|$ above.

REFERENCES

[Ahl79] Lars V. Ahlfors. Complex analysis. An introduction to the theory of analytic functions of one complex variable. International Series in pure and applied Mathematics. McGraw-Hill, 1979.

[BCF19] Frank Ban, Xi Chen, Adam Freilich, Rocco A. Servedio, and Sandip Sinha. Beyond trace reconstruction: Population recovery from the deletion channel. In 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS), pages 745–768. IEEE Computer Society, 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 APPROX/RANDOM 2019, volume 145 of LIPIcs, pages 44:1–44:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[BE97] Peter Borwein and Tamás Erdélyi. Littlewood-type polynomials on subarcs of the unit circle. Indiana University Mathematics Journal, 46(4):1323–1346, 1997.

[BEK99] Peter Borwein, Tamás Erdélyi, and Géza Kós. Littlewood-type problems on $[0, 1]$. Proceedings of the London Mathematical Society, 88(2):22–46, 1999.

[BKKM04] Tugkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. In Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, pages 910–918, 2004.

[Cha21a] Zachary Chase. New lower bounds for trace reconstruction. Ann. Inst. H. Poincaré, 57(2):627–643, 2021.
[Cha21b] Zachary Chase. Separating words and trace reconstruction. In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pages 21–31. ACM, 2021.

[DOS17] Anindya De, Ryan O’Donnell, and Rocco A. Servedio. Optimal mean-based algorithms for trace reconstruction. In Proceedings of the 49th ACM Symposium on Theory of Computing (STOC), pages 1047–1056, 2017.

[DOS19] Anindya De, Ryan O’Donnell, and Rocco A. Servedio. Optimal mean-based algorithms for trace reconstruction. Annals of Applied Probability, 29(2):851–874, 2019.

[HHP18] Lisa Hartung, Nina Holden, and Yuval Peres. Trace reconstruction with varying deletion probabilities. In Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2018, New Orleans, LA, USA, January 8-9, 2018., pages 54–61, 2018.

[HL20] Nina Holden and Russell Lyons. Lower bounds for trace reconstruction. Annals of Applied Probability, 30(2):503–525, 2020.

[HMPW08] Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. Trace reconstruction with constant deletion probability and related results. In Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, pages 389–398, 2008.

[HPP18] Nina Holden, Robin Pemantle, and Yuval Peres. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. In Proceedings of the Conference On Learning Theory, COLT 2018, volume 75 of Proceedings of Machine Learning Research, pages 1799–1800, 2018.

[HPPZ20] Nina Holden, Robin Pemantle, Yuval Peres, and Alex Zhai. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. CoRR, abs/1801.04783, 2020.

[Ka73] V. V. Kalashnik. Reconstruction of a word from its fragments. Computational Mathematics and Computer Science (Vychislitel’naya matematika i vychislitel’naya tehnika), Kharkov, 4:56–57, 1973.

[KM05] Sampath Kannan and Andrew McGregor. More on reconstructing strings from random traces: Insertions and deletions. In IEEE International Symposium on Information Theory, pages 297–301, 2005.

[KMMP19] Akshay Krishnamurthy, Arya Mazumdar, Andrew McGregor, and Soumyabrata Pal. Trace reconstruction: Generalized and parameterized. In 27th Annual European Symposium on Algorithms, ESA 2019, volume 144 of LIPIcs, pages 68:1–68:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

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

[Lev01b] Vladimir Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. Journal of Combinatorial Theory Series A, 93(2):310–332, 2001.

[MPV14] Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace reconstruction revisited. In Proceedings of the 22nd Annual European Symposium on Algorithms, pages 689–700, 2014.

[Nar21] Shyam Narayanan. Improved algorithms for population recovery from the deletion channel. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, pages 1259–1278. SIAM, 2021.

[NP17] Fedor Nazarov and Yuval Peres. Trace reconstruction with exp(\(o(n^{1/3})\)) samples. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, pages 1042–1046, 2017.

[O'D14] Ryan O’Donnell. Analysis of Boolean Functions. Cambridge University Press, 2014.

[PZ17] Yuval Peres and Alex Zhai. Average-case reconstruction for the deletion channel: subpolynomially many traces suffice. In Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, pages 228–239, 2017. Full version available at https://arxiv.org/abs/1708.00854.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. ACM, 51(3):385–463, 2004.

[U08] David C. Ulrich. Complex made simple, volume 97 of Graduate Studies in Mathematics. American Mathematical Society, 2008.

[VS08] Krishnamurthy Viswanathan and Ram Swaminathan. Improved string reconstruction over insertion-deletion channels. In Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 399–408, 2008.

[Wik20a] Wikipedia contributors. Hadamard three-circle theorem. Wikipedia, The Free Encyclopedia, Accessed June 28, 2020. https://en.wikipedia.org/wiki/Hadamard_three-circle_theorem.

[Wik20b] Wikipedia contributors. Jensen’s formula. Wikipedia, The Free Encyclopedia, Accessed June 28, 2020. https://en.wikipedia.org/wiki/Jensen%27s_formula.