Polynomial-time trace reconstruction in the smoothed complexity model

Xi Chen* Anindya De† Chin Ho Lee‡ Rocco A. Servedio§ Sandip Sinha¶

Abstract

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 δ 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 $\{0,1\}^n$ [6, 19, 7, 8, 4] and in the average-case setting where x is drawn uniformly at random from $\{0,1\}^n$ [21, 9, 8, 4].

This paper studies trace reconstruction in the *smoothed analysis* setting, in which a "worst-case" string x^{worst} is chosen arbitrarily from $\{0,1\}^n$, and then a perturbed version $\boldsymbol{x}$ of x^{worst} is formed by independently replacing each coordinate by a uniform random bit with probability σ. The problem is to reconstruct $\boldsymbol{x}$ 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 $\text{poly}(n)$ running time and traces and succeeds with high probability in reconstructing the string $\boldsymbol{x}$. This stands in contrast with the worst-case version of the problem, for which the best known sample complexity is $\exp(\tilde{O}(n^{1/5}))$ [5], a recent improvement on $\exp(O(n^{1/3}))$ [6, 19].

Our approach is based on reconstructing $\boldsymbol{x}$ 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 $\text{poly}(n)$-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.

*Columbia University. Supported by NSF grants CCF-1703925 and IIS-1838154. Email: xichen@cs.columbia.edu

†University of Pennsylvania.Supported by NSF grants CCF-1926872 and CCF-1910534. Email: anindyad@cis.upenn.edu

‡Columbia University. Supported by the Croucher Foundation and the Simons Collaboration on Algorithms and Geometry. Email: c.h.lee@columbia.edu

§Columbia University. Supported by NSF grants CCF-1814873, IIS-1838154, CCF-1563155, and by the Simons Collaboration on Algorithms and Geometry. Email: rocco@cs.columbia.edu

¶Columbia University. Supported by NSF grants CCF-1714818, CCF-1822809, IIS-1838154, CCF-1617955, CCF-1740833, and by the Simons Collaboration on Algorithms and Geometry. Email: sandip@cs.columbia.edu

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 δ and outputs the concatenation of 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 $\boldsymbol{z} \sim \text{Del}_\delta(x)$, is the string $\boldsymbol{z}$ 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 [12]. The problem was studied on and off throughout the 2000s [16, 15, 3, 13, 23, 11, 17], and has seen a renewed surge of recent interest over the past few years [6, 19, 21, 9, 7, 4, 1, 2, 14, 18, 10] 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. [3], who gave a $\text{poly}(n)$-time algorithm that uses $\text{poly}(n)$ traces and succeeds when the deletion rate δ is very small, at most $n^{-1/2-\varepsilon}$ for any $\varepsilon > 0$. In [11] Holenstein et al. gave an algorithm that runs in $\exp(\tilde{O}(n^{1/2}))$ time us-

ing $\exp(\tilde{O}(n^{1/2}))$ traces and succeeds for any δ bounded away from 1 by a constant. Simultaneous and independent works of De et al. [6] and Nazarov and Peres [19] gave an algorithm that improves the running time and sample complexity of [11] to $\exp(O(n^{1/3}))$. In this same constant-δ regime, successively stronger lower bounds on the required sample complexity were given by [17, 8], culminating in a $\tilde{\Omega}(n^{3/2})$ lower bound due to Chase [4].

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 [3, 13, 23] gave efficient algorithms that succeed for trace reconstruction of almost all $x \in \{0,1\}^n$ for various $o_n(1)$ deletion rates δ, and [11] gave an algorithm that runs in $\mathrm{poly}(n)$ time using $\mathrm{poly}(n)$ traces when δ is at most some sufficiently small constant. More recent results of Peres and Zhai [21] and Holden et al. [9, 10], which build on worst-case trace reconstruction results of [6, 19], substantially improve on this, with [9, 10] giving an algorithm which uses $\exp(O(\log^{1/3} n))$ traces to reconstruct a random $x \in \{0,1\}^n$ in $n^{1+o_n(1)}$ time when the deletion rate is any constant bounded away from 1.

Summarizing the results described above, the current $\exp(O(n^{1/3}))$ 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 [22], 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 σ-perturbation to a worst-case input instance; here σ 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 σ 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^{worst} is chosen arbitrarily from $\{0,1\}^n$, and then a randomly perturbed version $\boldsymbol{x}$ of the string x^{worst} is formed by independently replacing each coordinate of x^{worst} by a uniform random bit with probability σ. The goal is to reconstruct $\boldsymbol{x}$ given access to independent traces drawn from $\mathrm{Del}_\delta(\boldsymbol{x})$. 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.

As our main result, we give an algorithm for the smoothed trace reconstruction problem. For any initial string x^{worst}, our algorithm can recover a $1 - 1/\mathrm{poly}(n)$ fraction of perturbed strings $\boldsymbol{x}$ obtained from x^{worst} (for any $\mathrm{poly}(n)$) 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 1.1 (Polynomial time smoothed trace reconstruction) *Let $0 < \delta, \eta, \tau < 1$ and $0 < \sigma \leq 1$. Let x^{worst} be an arbitrary and unknown string in $\{0,1\}^n$ and let $\boldsymbol{x}$ be formed from x^{worst} by independently replacing each bit of x^{worst} with a uniform random bit from $\{0,1\}$ with probability σ.*

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

$$\left(\frac{n}{\eta}\right)^{O\left(\frac{1}{\sigma(1-\delta)} \log \frac{2}{1-\delta}\right)} \log \frac{1}{\tau}.$$

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 [11, 6, 19] for deletion rates bounded

away from 1 are "mean-based," meaning that they only use (estimates of) the n expected values $\mathbf{E}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}[\boldsymbol{y}_i]$, $i = 1,\dots,n$. The two papers [6, 19] 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 [6, 19] 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 [11, 6, 19] 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 [21, 9, 10] work by aligning individual traces (and are not mean-based). The analysis builds off of some of the structural results established in [6, 19], but also employs sophisticated probabilistic arguments which heavily depend on the randomness of the source string x being reconstructed.

As noted in [9, 10], 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^{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^{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 [9, 10] even to the setting in which the starting string x^{worst} is a *known* arbitrary string, let alone to our setting in which x^{worst} is both arbitrary and unknown.

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 $x = (x_1,\dots,x_n)$ is a sequence of contiguous characters of x, i.e. a $(b-a+1)$-character string $(x_a, x_{a+1},\dots,x_b)$ for some $1 \le a \le b \le n$.

Reconstruction from subwords. Given a length $1 \le k \le n$, let us write $\mathrm{subword}(x,k)$ 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

$$\mathrm{subword}\big(0^{n/4}1^{n/4-1}0^{n/4}1^{n/4+1},k\big) \quad \text{and} \quad \mathrm{subword}\big(0^{n/4}1^{n/4+1}0^{n/4}1^{n/4-1},k\big)$$

are identical for every $k \le n/4-1$. 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.

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 $x^{\mathrm{worst}} \in \{0,1\}^n$, with high probability a random σ-perturbation $\boldsymbol{x}$ of x^{worst} is such that $\mathrm{subword}(\boldsymbol{x},k)$ *does* uniquely identify $\boldsymbol{x}$ within $\{0,1\}^n$ even if k is relatively small. Moreover, there is an efficient algorithm to reconstruct $\boldsymbol{x}$ from $\mathrm{subword}(\boldsymbol{x},k)$. This is captured by the following result, which we prove in Section 3:

Lemma 1.1 (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* $\mathrm{subword}(x,k)$ *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* $\mathrm{poly}(n)$ *time and has the following property: for any $x^{\mathrm{worst}} \in \{0,1\}^n$, if $\boldsymbol{x}$ is a random σ-perturbation of x^{worst} (i.e. $\boldsymbol{x}$ is obtained by independently replacing each bit of x^{worst} with a uniform random bit with probability σ), then with probability at least $1-\eta$ the output of* `Reconstruct-from-subword-deck` *on input* $\mathrm{subword}(\boldsymbol{x},k)$ *is the string $\boldsymbol{x}$.*

The subword deck reconstruction problem. Lemma 1.1 naturally motivates the algorithmic problem of *subword deck reconstruction*: given access to independent traces drawn from $\mathrm{Del}_\delta(x)$ 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 1.2 (Reconstructing the k-subword deck of x) *Let $0 < \delta,\tau < 1$. There is an algorithm* `Reconstruct-subword-deck` *which takes as input a parameter $1 \le k \le n$ and access to independent traces*

of an unknown source string $x \in \{0,1\}^n$. The running time of `Reconstruct-subword-deck`*, as well as the number of traces it uses, is*

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

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

Theorem 1.1 follows immediately from Lemma 1.1 and Theorem 1.2. We note that Theorem 1.2 dominates the overall running time of Theorem 1.1, and that Theorem 1.2 works for arbitrary strings.

The algorithm in Lemma 1.1 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 $\{0,\dots,n-1\}$ as $x = (x_0,\dots,x_{n-1})$.)

Definition 1 *Given a k-bit string $w = (w_0,\dots,w_{k-1}) \in \{0,1\}^k$, a k-bit string $(w_1,\dots,w_{k-1},b)$ 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 $\boldsymbol{x}$ is obtained by a random σ-perturbation of x^{worst}, then $\boldsymbol{x}$ has useful *local uniqueness properties*. More precisely, for $k = O(\log(n/\tau)/\sigma)$, a simple probabilistic argument shows that with high probability $\boldsymbol{x}[n-k : n-1]$ is the unique element of subword$(\boldsymbol{x},k)$ with no right-extension in subword$(\boldsymbol{x},k)$. Consequently, we can identify $\boldsymbol{x}[n-k : n-1]$ from the k-subword deck subword$(\boldsymbol{x},k)$ of $\boldsymbol{x}$. This argument can be extended inductively without much difficulty to in fact identify the whole of $\boldsymbol{x}$.

In contrast, Theorem 1.2 is substantially more challenging. The structural results that underlie Theorem 1.2 are based on two different sets of analytic arguments. The first argument only works when $\delta \le 1/2$ 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$(\boldsymbol{x},k)$ can be obtained by computing the multiplicity of occurrences of each $w \in \{0,1\}^k$ in the set subword$(\boldsymbol{x},k)$; we denote this multiplicity by $\#(w,\boldsymbol{x})$. The first key step is to define a univariate polynomial (in the variable ζ) $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ which has the following two key properties: (i) $\mathrm{SW}_{\boldsymbol{x},w}(0) = \#(w,\boldsymbol{x})$, and (ii) using traces from $\mathrm{Del}_\delta(\boldsymbol{x})$, we have an unbiased estimator for $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ for $\zeta = \delta$. Next, observe that given traces from $\mathrm{Del}_\delta(\boldsymbol{x})$, we can trivially simulate traces from $\mathrm{Del}_{\delta'}(\boldsymbol{x})$ for any $\delta' \ge \delta$, and hence we can get an unbiased estimator for $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ for any $\zeta \in [\delta,1]$. Recall, though, that our goal is to estimate $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ at $\zeta = 0$ and thus items (i) and (ii) above do not give us an unbiased estimator for $\mathrm{SW}_{\boldsymbol{x},w}(0)$.

The most obvious idea at this point would be to do polynomial interpolation and use estimates for $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ for $\zeta \in [\delta,1]$ to infer $\mathrm{SW}_{\boldsymbol{x},w}(0)$. Unfortunately, directly applying Lagrange interpolation is too naive an approach: to accurately estimate $\mathrm{SW}_{\boldsymbol{x},w}(0)$, it turns out that we need $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ for $\zeta \in [\delta,1]$ up to error $\pm 2^{-\Theta(n)}$. However, to estimate $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ up to error $\pm\kappa$, at least $\mathrm{poly}(1/\kappa)$ traces from $\mathrm{Del}_\delta(\boldsymbol{x})$ are needed (Lemma 6.1). Thus, directly applying Lagrange interpolation would require a sample complexity that grows like $2^{\Theta(n)}$, which is too expensive.

Our approach is to forego Lagrange interpolation and instead (in essence) interpolate using tools from complex analysis. In particular, we prove a new structural result (Theorem 6.3) about polynomials whose constant coefficient is not too small and whose coefficients have magnitude bounded from above by a parameter m (which is set to be n^k in our application given that every coefficient of $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ is bounded by n^k). This result implies that $\mathrm{SW}_{\boldsymbol{x},w}(0)$ (which must be an integer given that $\mathrm{SW}_{\boldsymbol{x},w}(0) = \#(w,\boldsymbol{x})$) is uniquely determined by the values of $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ in the interval $\zeta \in [\delta,1]$ if these values are given up to error $n^{-O(k/(1-\delta))}$; see Theorem 6.2. Thus, in principle we can determine $\mathrm{SW}_{\boldsymbol{x},w}(0)$ by estimating $\mathrm{SW}_{\boldsymbol{x},w}(\zeta)$ for values of $\zeta \in [\delta,1]$ to error $\pm n^{-O(k/(1-\delta))}$. This essentially implies that $\mathrm{SW}_{\boldsymbol{x},w}(0)$ can be determined using $\approx n^{O(k/(1-\delta))}$ traces from $\mathrm{Del}_\delta(\boldsymbol{x})$. (Note though that this sample complexity is not quite as good as is claimed in Theorem 1.2. We refine the above argument, using stronger coefficient bounds on $\mathrm{SW}_{\boldsymbol{x},w}$ and other ideas described at the beginning of Section 7, to get Theorem 1.2 in its full strength as stated earlier.)

In closing this subsection, we emphasize that while Theorem 6.2 is about the behavior of polynomials on the real line, its proof crucially uses tools from complex analysis such as Jensen's formula and the Hadamard three circle theorem. We further note that while we

have sketched above how $\mathrm{SW}_{\boldsymbol{x},w}(0)$ can be determined in principle, this does not necessarily give an efficient algorithm. To get an efficient algorithm, we use an approach based on linear programming.

1.5 Discussion and future work We view this paper as a first exploration, establishing that the algorithmic framework of smoothed analysis can be fruitfully brought to bear on the trace reconstruction problem. There are many interesting questions and directions for future work, some of which we highlight below.

One natural question is to establish strong sample complexity lower bounds for smoothed trace reconstruction. Currently the best lower bound we are aware of for this framework is $\tilde{\Omega}(\log^{5/2} n)$ for average-case trace reconstruction due to [4]. Can an $n^{\Omega(1)}$ lower bound be established for the smoothed model?

Another natural goal is to quantitatively strengthen our algorithmic result. In the regime of $\sigma = c/n$ with c a small constant, the smoothed problem reduces to the worst-case problem, and in the regime $\sigma = 1$ it reduces to the average-case problem; however, the running times of our algorithm in these regimes do not match the state-of-the-art running times for the corresponding problems that are provided in [6, 19] and in [9, 10] respectively. As a concrete first question along these lines, is it possible to improve the sample complexity of our algorithm from its current $n^{1/\sigma}$ dependence on the perturbation rate to a dependence more like $n^{1/\sigma^{1/3}}$?

2 Preliminaries

Notation. Given a nonnegative integer n, we write $[n]$ to denote $\{1, \dots, n\}$. Given integers $a \le b$ we write $[a : b]$ to denote $\{a, \dots, b\}$. It will be convenient for us to index a binary string $x \in \{0,1\}^n$ using $[0 : n-1]$ as $x = (x_0, \dots, x_{n-1})$. Given such a string x and integers $0 \le a < b \le n-1$, we write $x[a : b]$ to denote the *subword* $(x_a, x_{a+1}, \dots, x_b)$. 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 $\mathbb{Z}_{\ge 0}$. For a vector $\alpha = (\alpha_1, \dots, \alpha_\ell) \in \mathbb{Z}^\ell_{\ge 0}$, we write $|\alpha|$ to denote $\alpha_1 + \alpha_2 + \cdots + \alpha_\ell$, and write $\alpha!$ to denote $\alpha_1!\alpha_2! \cdots \alpha_\ell!$.

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 $(x_a, x_{a+1}, \dots, x_{a+k-1})$ 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 $w_i \ne *$.

Distributions. We use bold font letters to denote probability distributions and random variables, which should be clear from the context. We write "$\boldsymbol{x} \sim \mathbf{X}$" to indicate that random variable $\boldsymbol{x}$ is distributed according to distribution $\mathbf{X}$.

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 $\mathrm{Del}_\delta(x)$ to denote the distribution of the string that results from passing x through the δ-deletion channel (so the distribution $\mathrm{Del}_\delta(x)$ is supported on $\{0,1\}^{\le n}$), and we refer to a string in the support of $\mathrm{Del}_\delta(x)$ as a *trace* of x. Recall that a random trace $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ is obtained by independently deleting each bit of x with probability δ 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 $x^{\mathrm{worst}} \in \{0,1\}^n$, a *$\sigma$-perturbation of x^{worst}* is a random string $\boldsymbol{x} \in \{0,1\}^n$ obtained by independently setting each coordinate $\boldsymbol{x}_i$ to be x^{worst}_i with probability $1-\sigma$ and to be uniformly random with the remaining probability σ. Equivalently, $\boldsymbol{x}$ is a random string that is $(1-\sigma)$-correlated with x^{worst}; in the notation of Chapter 2 of [20], we may write this as "$\boldsymbol{x} \sim N_{1-\sigma}(x^{\mathrm{worst}})$."

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

3 Reconstructing perturbed worst-case strings from their subword decks: Proof of Lemma 1.1

In this section we prove Lemma 1.1:

Restatement of Lemma 1.1 (Reconstructing perturbed strings from their subword decks)**.** *Let $0 < \sigma, \eta < 1$. There is a deterministic algorithm* `Reconstruct-from-subword-deck` *which takes*

[1] For simplicity in this work we assume that the deletion probability δ is known to the reconstruction algorithm. We note that it is possible to obtain a high-accuracy estimate of δ simply by measuring the average length of traces received from the deletion channel.

as input the k-subword deck $\text{subword}(x,k)$ *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 string* $x^{\text{worst}} \in \{0,1\}^n$*, if* $\boldsymbol{x}$ *is a random* σ*-perturbation of* x^{worst} *(i.e.* $\boldsymbol{x}$ *is obtained by independently replacing each bit of* x^{worst} *with a uniform random bit with probability* σ*), then with probability at least* $1-\eta$ *the output of* `Reconstruct-from-subword-deck` *on input* $\text{subword}(\boldsymbol{x},k)$ *is the string* $\boldsymbol{x}$*.*

The idea of Lemma 1.1 is very simple: a probabilistic argument shows that for any worst-case string x^{worst}, a random σ-perturbation introduces enough variability into $\boldsymbol{x} \sim N_{1-\sigma}(x^{\text{worst}})$ so that the k-subwords comprising the k-subword deck of $\boldsymbol{x}$ can be easily pieced together in a unique way to yield $\boldsymbol{x}$ 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 $\ell = n-k$.
3. While $\ell > 0$, 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 ℓ 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 2 *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 1.1, it remains only to establish the following claim:

Claim 3.1 *Fix any string* $x^{\text{worst}} \in \{0,1\}^n$*. Then for* $k = O(\log(n/\eta)/\sigma)$

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x} \sim N_{1-\sigma}(x^{\text{worst}})} \left[\boldsymbol{x} \text{ is } k\text{-good} \right] \geq 1-\eta.$$

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

$$\begin{aligned}
&\mathbf{Pr}\left[E(\boldsymbol{x})\right] \\
&\leq \mathbf{Pr}\left[\exists\, i,j \text{ such that } \boldsymbol{x}[i:i+k-2] = \boldsymbol{x}[j:j+k-2]\right] \\
&\text{(by a union bound)} \\
&\leq \sum_{0 \leq i < j \leq n-k+1} \mathbf{Pr}\left[\boldsymbol{x}[i:i+k-2] = \boldsymbol{x}[j:j+k-2]\right].
\end{aligned}$$

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

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

$$\mathbf{Pr}\left[\boldsymbol{x}_i = \boldsymbol{x}_j\right] \cdot \prod_{\ell=1}^{k-2} \mathbf{Pr}\left[\boldsymbol{x}_{i+\ell} = \boldsymbol{x}_{j+\ell} \,\middle|\, \boldsymbol{x}_{i+h} = \boldsymbol{x}_{j+h} \text{ for all } h = 0,\ldots,\ell-1\right].$$

The first factor $\mathbf{Pr}\left[\boldsymbol{x}_i = \boldsymbol{x}_j\right]$ is at most $1-\sigma/2$ because for any fixed value b of $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ agrees with b after the perturbation with probability at most $1-\sigma/2$. The upper bound of $1-\sigma/2$ holds for every other factor in the product. For the ℓth factor, we note that for any fixed values of $\boldsymbol{x}_i,\ldots,\boldsymbol{x}_{j+\ell-1}$ that satisfy the conditioning part $\boldsymbol{x}_{i+h} = \boldsymbol{x}_{j+h}$ for all $h = 0,\ldots,\ell-1$, $\boldsymbol{x}_{j+\ell}$ agrees with the fixed value of $\boldsymbol{x}_{i+\ell}$ with probability at most $1-\sigma/2$.

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

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

This finishes the proof of the claim. ∎

4 Reconstructing the k-subword deck: Towards proof of Theorem 1.2

The remaining task to establish the main result, Theorem 1.1, is to prove Theorem 1.2 (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 1.2 (Reconstructing the k-subword deck). *Let $0 < \delta, \tau' < 1$. There is an algorithm* `Reconstruct-subword-deck` *which takes as input a parameter $1 \le k \le n$ and access to independent traces of an unknown source string $x \in \{0,1\}^n$. The running time of* `Reconstruct-subword-deck`*, as well as the number of traces it uses, is*

$$\left(n/\rho^k\right)^{O(1/\rho)} \log(1/\tau').$$

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

The main algorithmic ingredient that underlies Theorem 1.2 is an algorithm for a closely related but slightly simpler problem. This algorithm, which we call `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 $\mathrm{subword}(x,k)$ (note that this multiplicity can be zero if w is not present as a subword of x):

Theorem 4.1 *Let $0 < \delta, \tau < 1$ and let $\rho = (1-\delta)/2$. There is an algorithm* `Multiplicity` *which takes as input a string $w \in \{0,1\}^k$ and access to independent traces of an unknown source string $x \in \{0,1\}^n$.* `Multiplicity` *runs in $\left(n/\rho^k\right)^{O(1/\rho)} \log(1/\tau)$ time and uses $\left(n/\rho^k\right)^{O(1/\rho)} \log(1/\tau)$ many traces from $\mathrm{Del}_\delta(x)$, and has the following property: For any unknown source string $x \in \{0,1\}^n$, with probability at least $1-\tau$ the output of* `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 1.2 from Theorem 4.1:

Proof. (*Proof of Theorem 1.2 using Theorem 4.1.*) Let $\ell = \lfloor \log n \rfloor$. We first consider the case that $k \le \ell$. In this case `Reconstruct-subword-deck` simply runs `Multiplicity`(w) once for each of the 2^k strings $w \in \{0,1\}^k$, with the confidence parameter "τ" for each run of `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 $2^k \left(n/\rho^k\right)^{O(1/\rho)} \log\left(2^k/\tau'\right) = \left(n/\rho^k\right)^{O(1/\rho)} \log(1/\tau')$ and the sample complexity is $\left(n/\rho^k\right)^{O(1/\rho)} \log(1/\tau')$.

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, `Reconstruct-subword-deck` runs `Multiplicity` on each of the 2^ℓ strings $w \in \{0,1\}^\ell$ with confidence parameter $\tau'/(2nk)$, thereby obtaining the ℓ-subword deck $\mathrm{subword}(x,\ell)$. 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 $w \in \mathrm{subword}(x, \ell+j-1)$, the algorithm runs `Multiplicity`$(w0)$ and `Multiplicity`$(w1)$, 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 $\mathrm{subword}(x, \ell+j)$ if its $(\ell+j-1)$-bit prefix is present in $\mathrm{subword}(x, \ell+j-1)$. Since there are at most $n + 2n(k-\ell) < 2kn$ many runs of `Multiplicity` overall, the running time of `Reconstruct-subword-deck` is at most $O(kn) \cdot \left(n/\rho^k\right)^{O(1/\rho)} \log(2kn/\tau') = \left(n/\rho^k\right)^{O(1/\rho)} \log(1/\tau')$ and the sample complexity is at most $\left(n/\rho^k\right)^{O(1/\rho)} \log(1/\tau')$, and Theorem 1.2 is proved. ∎

Thus, in the rest of the paper, we focus on proving Theorem 4.1.

4.1 The subword polynomial

The following "subword polynomial" plays an important role in our approach:

Definition 3 *Given $x \in \{0,1\}^n$ and $w = (w_0, \ldots, w_{k-1}) \in \{0,1\}^k$, let $\mathrm{SW}_{x,w}(\zeta)$ be the following univariate polynomial of degree $n-k$:*

$$\mathrm{SW}_{x,w}(\zeta) := \sum_{\substack{\alpha \in \mathbb{Z}_{\ge 0}^{k-1} \\ |\alpha| \le n-k}} \#\left(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1},\, x\right) \cdot \zeta^{|\alpha|}.$$

In words, the degree-ℓ coefficient of the subword polynomial $\mathrm{SW}_{x,w}$ is the number of ways that w arises as a substring of x with a total of exactly ℓ extraneous additional characters interspersed among the characters of w. In particular, we have that the constant term of $\mathrm{SW}_{x,w}$ (i.e. $\mathrm{SW}_{x,w}(0)$, since $0^0 = 1$) is equal to $\#(w,x)$, the frequency of w as a subword of x, which is what Theorem 4.1 aims to estimate efficiently from traces of x.

4.2 Outline of our approach

We prove Theorem 4.1 by giving two different algorithms depending on

the value of the deletion rate δ. The first of these algorithms, $\texttt{Multiplicity}_{\text{small-}\delta}$, gives a simple and direct approach to compute the value $\mathrm{SW}_{x,w}(0) = \#(w, x)$; however this approach requires the deletion rate δ 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, $\texttt{Multiplicity}_{\text{large-}\delta}$, 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 analysis (albeit one that works only for $\delta < 1/2$) may wish to focus on $\texttt{Multiplicity}_{\text{small-}\delta}$ (Section 5). Readers who are interested in a more involved approach that succeeds for all $\delta < 1$ may wish to focus on $\texttt{Multiplicity}_{\text{large-}\delta}$ (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 5.1 and Theorem 6.1, 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 5.1 and Theorem 6.1 to achieve a $\mathrm{poly}(n) \cdot \exp(O(k))$ running time and sample complexity, and thereby complete the proof of Theorem 4.1.

5 $\texttt{Multiplicity}'_{\text{small-}\delta}$: An algorithm for deletion rate $\delta < 1/2$

In this section we prove Theorem 5.1, a weaker version of Theorem 4.1. 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 5.1 works when $\delta \le 1/2$; we only require $\delta < 1/2$ later in Section 7.1 to achieve the improved running time and sample complexity in Theorem 4.1 based on a similar approach (the running time achieved in that section will depend on how close δ is to $1/2$).

Theorem 5.1 *Let $0 < \delta \le 1/2$. There is an algorithm $\texttt{Multiplicity}'_{\text{small-}\delta}$ 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 $\tau > 0$. $\texttt{Multiplicity}'_{\text{small-}\delta}$ draws $n^{O(k)} \cdot \log(1/\tau)$ traces from $\mathrm{Del}_\delta(x)$, runs in time $n^{O(k)} \cdot \log(1/\tau)$, and has the following property: For any unknown source string $x \in \{0,1\}^n$, with probability at least $1 - \tau$ the output of $\texttt{Multiplicity}'_{\text{small-}\delta}$ is the multiplicity of w in* $\mathrm{subword}(x, k)$ *(i.e. the number of occurrences of w as a subword of x).*

In Section 7 we will build on Theorem 5.1 to give a stronger version that has $\mathrm{poly}(n) \cdot \exp(O(k))$ 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 $\mathrm{SW}_{x,w}(\zeta)$ in Theorem 5.2, 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 $\mathrm{SW}_{x,w}(\zeta)$ allows one to evaluate $\mathrm{SW}_{x,w}(\zeta)$ at $\zeta = 0$ up to a small error (say, ± 0.1) using traces of x (see Corollary 5.1) when $\delta \le 1/2$. Given that $\mathrm{SW}_{x,w}(0)$ is an integer, the result can be rounded to obtain the exact value of $\mathrm{SW}_{x,w}(0)$; this finishes the proof of Theorem 5.1.

We remark that the expression for $\mathrm{SW}_{x,w}(\zeta)$ given in Theorem 5.2 works for any $\zeta \in \mathbb{C}$, when viewing $\mathrm{SW}_{x,w}(\zeta)$ as a polynomial over $\mathbb{C}$, and may be useful for subsequent work. Indeed Corollary 5.1 shows that $\mathrm{SW}_{x,w}(\zeta)$ 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 complex disc with center δ and radius $1 - \delta$. We need $\delta \le 1/2$ so that $0 \in B_{1-\delta}(\delta)$.

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

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

Theorem 5.2 *Let x, k and w be as above. Then for all $\zeta \in \mathbb{C}$ we have*

$$\mathrm{SW}_{x,w}(\zeta) = \frac{1}{(1-\delta)^k} \sum_{\substack{\alpha \in \mathbb{Z}_{\ge 0}^{k-1} \\ |\alpha| \le n-k}} \mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \Big[\#(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1}, \boldsymbol{y}) \Big] \cdot \left(\frac{\zeta - \delta}{1 - \delta} \right)^{|\alpha|}.$$

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

Corollary 5.1 (Corollary of Theorem 5.2) *Let x, k, w be as above, and let $\varepsilon > 0$. Then, given access*

to traces $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$, there exists an algorithm which, given as input any $\zeta \in B_{1-\delta}(\delta)$, evaluates $\mathrm{SW}_{x,w}(\zeta)$ up to error $\pm\varepsilon$ with success probability at least $1-\tau$. The algorithm takes

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

many traces and running time.

Recall that $\mathrm{SW}_{x,w}(0) = \#(w,x)$. When $\delta \le 1/2$, the disc $B_{1-\delta}(\delta)$ contains the origin. Therefore, setting $\varepsilon = 1/3$ in Corollary 5.1 directly implies an algorithm $\texttt{Multiplicity}'_{\text{small-}\delta}$ that uses $((n/(1-\delta))^{O(k)} \cdot \log(1/\tau) = n^{O(k)} \cdot \log(1/\tau)$ traces and running time to evaluate $\mathrm{SW}_{x,w}(0)$ up to an error of $\varepsilon = 1/3$, which succeeds with probability at least $1-\tau$. It then rounds the result to the nearest integer to obtain $\mathrm{SW}_{x,w}(0) = \#(w,x)$ given that the latter is an integer. This finishes the proof of Theorem 5.1.

Proof. (Proof of Corollary 5.1) The algorithm simply draws

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

many traces $\boldsymbol{y}_1, \dots, \boldsymbol{y}_s$ of x and uses them to compute an empirical estimate $\tilde{E}_\alpha$ of

$$E_\alpha := \mathop{\mathbf{E}}_{\boldsymbol{y}\sim\mathrm{Del}_\delta(x)}\left[\#(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1},\, \boldsymbol{y})\right]$$

for each $\alpha \in \mathbb{Z}_{\ge 0}^{k-1}$ with $|\alpha| \le n-k$. This is done by computing $\#(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1}, \boldsymbol{y}_i)$ for each α and $\boldsymbol{y}_i$ (in time polynomial in n), and then taking the average over $\boldsymbol{y}_1, \dots, \boldsymbol{y}_s$ for each α. Given that the number of α's is at most n^k, the overall running time is $s \cdot n^k \cdot \mathrm{poly}(n)$, as stated in Corollary 5.1.

Given that $\#(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1}, \boldsymbol{y})$ in (5.1) is between 0 and n, it follows from our choice of s, a Chernoff bound and a union bound, that with probability at least $1-\tau$, every empirical estimate $\tilde{E}_\alpha$ satisfies

$$|\tilde{E}_\alpha - E_\alpha| \le \varepsilon \cdot \left(\frac{1-\delta}{n}\right)^k. \tag{5.1}$$

Using

$$\left|\frac{\zeta-\delta}{1-\delta}\right| \le 1$$

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

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

and the estimate is correct up to error

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

where the inequality holds by Equation (5.1) given that the number of α's is no more than n^k. ∎

5.2 Generalized deletion polynomial and the proof of Theorem 5.2 In this subsection we prove Theorem 5.2. We first introduce a more general class of polynomials, the (x,f)-deletion-channel polynomials (see Definition 4), of which $\mathrm{SW}_{x,w}$ is a special case. We then prove an extension of Theorem 5.2 (see Theorem 5.3) which applies to every (x,f)-deletion channel polynomial; Theorem 5.2 follows as a direct corollary. While we don't need the full generality of Theorem 5.3 to prove Theorem 5.2, working with this new class of polynomials makes our proofs cleaner. We also believe that Theorem 5.3 in the general form may be useful for subsequent analysis.

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

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

and the $|\gamma|$-th order partial derivatives of P

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

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

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

Definition 4 *Given $f\colon \{0,1\}^k \to \mathbb{C}$ and a string $x \in \{0,1\}^n$, the (x,f)-*deletion-channel polynomial *$P_{x,f}\colon \mathbb{C}^k \to \mathbb{C}$ is defined by*

$$P_{x,f}(\xi) := \sum_{\substack{\gamma\in\mathbb{Z}_{\ge 0}^k \\ |\gamma|\le n-k}} f(x_{\gamma_1}, x_{\gamma_1+\gamma_2+1}, \dots, x_{\gamma_1+\cdots+\gamma_k+(k-1)})\cdot\xi^\gamma.$$

We call $P_{x,f}$ the (x,f)-deletion-channel polynomial because by choosing $k=1$ and $f\colon \{0,1\} \to \{0,1\}$ to be the 1-bit identity function $\mathrm{id}(x) = x$, we have that

$$P_{x,\mathrm{id}}(\xi) = \sum_{i=0}^{n-1} x_i \xi^i$$

is the deletion-channel polynomial defined in [6].

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 5.2 follows as a direct corollary.

Theorem 5.3 *For any $\xi \in \mathbb{C}^k$, we have*

$$P_{x,f}(\xi) = \frac{1}{(1-\delta)^k} \sum_{\substack{\beta \in \mathbb{Z}^k_{\geq 0} \\ |\beta| \leq n-k}} \mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \left[f(\boldsymbol{y}_{\beta_1}, \ldots, \boldsymbol{y}_{\beta_1+\cdots+\beta_k+k-1}) \right] \cdot \left(\frac{\xi - \vec{\delta}}{1-\delta} \right)^{\beta}.$$

Proof. (Proof of Theorem 5.2 assuming Theorem 5.3) Given $x \in \{0,1\}^n$ and $w \in \{0,1\}^k$ for some $k \in [n]$ as in the statement of Theorem 5.2, we take $f : \{0,1\}^k \to \{0,1\}$ to be the indicator function of w:

$$f(b_1, b_2, \ldots, b_k) = \mathbf{1}\left[(b_1, b_2, \ldots, b_k) = w\right].$$

Using this f we get the following connection between $\mathrm{SW}_{x,w}(\zeta)$ and $P_{x,f}(1, \zeta, \zeta, \ldots, \zeta)$:

$$\begin{aligned}
&\mathrm{SW}_{x,w}(\zeta) \\
&= \sum_{\substack{\alpha \in \mathbb{Z}^{k-1}_{\geq 0} \\ |\alpha| \leq n-k}} \#\left(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1},\, x\right) \cdot \zeta^{|\alpha|} \\
&= \sum_{\substack{\alpha \in \mathbb{Z}^{k-1}_{\geq 0} \\ |\alpha| \leq n-k}} \sum_{i=0}^{n-k-|\alpha|} f(x_i, x_{i+\alpha_1+1}, x_{i+\alpha_1+\alpha_2+2}, \ldots, x_{i+|\alpha|+k-1})\, 1^i\, \zeta^{|\alpha|} \\
&= P_{x,f}(1, \zeta, \zeta, \cdots, \zeta)
\end{aligned}$$

Applying Theorem 5.3 on $P_{x,f}(1, \zeta, \zeta, \ldots, \zeta)$, we have

$$\begin{aligned}
&\mathrm{SW}_{x,w}(\zeta) \\
&= \frac{1}{(1-\delta)^k} \sum_{\substack{\alpha \in \mathbb{Z}^{k-1}_{\geq 0} \\ |\alpha| \leq n-k}} \sum_{i=0}^{n-k-|\alpha|} \mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \left[f(\boldsymbol{y}_i, \boldsymbol{y}_{i+\alpha_1+1}, \ldots, \boldsymbol{y}_{i+|\alpha|+k-1}) \right] \cdot \left(\frac{\zeta - \delta}{1-\delta} \right)^{|\alpha|} \\
&= \frac{1}{(1-\delta)^k} \sum_{\substack{\alpha \in \mathbb{Z}^{k-1}_{\geq 0} \\ |\alpha| \leq n-k}} \mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \left[\#(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1},\, \boldsymbol{y}) \right] \cdot \left(\frac{\zeta - \delta}{1-\delta} \right)^{|\alpha|}
\end{aligned}$$

where the last step follows by linearity of expectation. This concludes the proof of Theorem 5.2. ∎

We now prove Theorem 5.3. The high-level idea is to relate the expectation of f over traces of x drawn from the deletion channel to partial derivatives of polynomial $P_{x,f}$ at $\vec{\delta}$, and then apply Taylor's expansion to $P_{x,f}$ at the point $\vec{\delta}$.

Claim 5.1 *Let $\beta \in \mathbb{Z}^k_{\geq 0}$ with $|\beta| \leq n-k$. We have*

$$\mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \left[f(\boldsymbol{y}_{\beta_1}, \ldots, \boldsymbol{y}_{\beta_1+\cdots+\beta_k+(k-1)}) \right] = (1-\delta)^k \cdot \frac{(1-\delta)^{|\beta|}}{\beta!} \cdot D^\beta P_{x,f}(\vec{\delta}).$$

To get some intuition, consider the special case of $k = 1$ (so $P_{x,f}$ is univariate) and $f = \mathrm{id}$. Then it is straightforward to verify that

$$\mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \left[\boldsymbol{y}_0 \right] = (1-\delta) \sum_{i=0}^{n-1} x_i \delta^i = (1-\delta) \cdot P_{x,\mathrm{id}}(\delta),$$

and

$$\begin{aligned}
\mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \left[\boldsymbol{y}_1 \right] &= (1-\delta) \sum_{i=1}^{n-1} x_i \binom{i}{1} (1-\delta) \delta^{i-1} \\
&= (1-\delta)^2 \sum_{i=1}^{n-1} x_i i \delta^{i-1} \\
&= (1-\delta)^2 \cdot D^1 P_{x,\mathrm{id}}(\delta).
\end{aligned}$$

Proof. (Proof of Claim 5.1) For a fixed $\gamma \in \mathbb{Z}^k_{\geq 0}$ with $|\gamma| \leq n-k$, we write

$$\gamma \to \beta, \text{ or equivalently } (\gamma_1, \gamma_2, \ldots, \gamma_k) \to (\beta_1, \beta_2, \ldots, \beta_k),$$

to denote the event that the $(\gamma_1, \gamma_1+\gamma_2+1, \ldots, \gamma_1+\cdots+\gamma_k+(k-1))$ positions of x become the $(\beta_1, \beta_1+\beta_2+1, \ldots, \beta_1+\cdots+\beta_k+(k-1))$ positions of $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$ respectively. For this to occur, each of $x_{\gamma_1}, x_{\gamma_1+\gamma_2+1}, \ldots, x_{\gamma_1+\cdots+\gamma_k+(k-1)}$ must be present in $\boldsymbol{y}$, which happens with probability $(1-\delta)^k$. Further, for each x_{γ_i} to become $\boldsymbol{y}_{\beta_i}$, exactly β_i out of the γ_i bits between (and including) positions $\gamma_1+\cdots+\gamma_{i-1}+i$ and $\gamma_1+\cdots+\gamma_i+(i-1)$ of x must be retained. So, the

probability of this event is

(5.2)

$$\begin{aligned}
&\mathbf{Pr}[\gamma \to \beta] \\
&= (1-\delta)^k \prod_{i=1}^{k} \binom{\gamma_i}{\beta_i} (1-\delta)^{\beta_i} \delta^{\gamma_i - \beta_i} \\
&= (1-\delta)^k \prod_{i=1}^{k} \frac{\gamma_i(\gamma_i-1)\cdots(\gamma_i-\beta_i+1)}{\beta_i!} (1-\delta)^{\beta_i} \delta^{\gamma_i-\beta_i} \\
&= (1-\delta)^k \cdot \left(\prod_{i=1}^{k} \frac{(1-\delta)^{\beta_i}}{\beta_i!} \right) \cdot \prod_{i=1}^{k} \Big(\gamma_i(\gamma_i-1)\cdots \\
&\qquad\qquad (\gamma_i-\beta_i+1) \cdot \delta^{\gamma_i-\beta_i} \Big) \\
&= (1-\delta)^k \cdot \frac{(1-\delta)^{|\beta|}}{\beta!} \cdot \prod_{i=1}^{k} \frac{d^{\beta_i}}{d\delta^{\beta_i}} \delta^{\gamma_i}.
\end{aligned}$$

As a result, we have that

$$\begin{aligned}
&\mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \Big[f(\boldsymbol{y}_{\beta_1}, \ldots, \boldsymbol{y}_{\beta_1+\cdots+\beta_k+(k-1)}) \Big] \\
&\quad = \sum_{|\gamma| \le n-k} f(x_{\gamma_1}, \ldots, x_{\gamma_1+\cdots+\gamma_k+(k-1)}) \cdot \mathbf{Pr}[\gamma \to \beta] \\
&\quad = (1-\delta)^k \cdot \frac{(1-\delta)^{|\beta|}}{\beta!} \sum_{|\gamma| \le n-k} f(x_{\gamma_1}, \ldots, \\
&\qquad\qquad x_{\gamma_1+\cdots+\gamma_k+(k-1)}) \cdot \prod_{i=1}^{k} \frac{d^{\beta_i}}{d\delta^{\beta_i}} \delta^{\gamma_i} \\
&\quad = (1-\delta)^k \cdot \frac{(1-\delta)^{|\beta|}}{\beta!} \cdot D^\beta P_{x,f}(\vec{\delta}).
\end{aligned}$$

(Equation (5.2))

This finishes the proof of Claim 5.1. ∎

Proof. (Proof of Theorem 5.3) Since $P_{x,f}$ is a polynomial of degree at most $n-k$, applying Taylor's expansion to $P_{x,f}$ at the point $\vec{\delta}$ and using Claim 5.1, we get that

$$\begin{aligned}
&(1-\delta)^k \cdot P_{x,f}(\xi) \\
&\quad = (1-\delta)^k \sum_{|\beta| \le n-k} \frac{D^\beta P_{x,f}(\vec{\delta})}{\beta!} \cdot (\xi - \vec{\delta})^\beta \\
&\quad = \sum_{|\beta| \le n-k} \mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \Big[f(\boldsymbol{y}_{\beta_1}, \ldots, \\
&\qquad\qquad \boldsymbol{y}_{\beta_1+\cdots+\beta_k+k-1}) \Big] \cdot \left(\frac{\xi - \vec{\delta}}{1-\delta} \right)^\beta.
\end{aligned}$$

∎

6 $\texttt{Multiplicity}'_{\text{large-}\delta}$: An algorithm for deletion rate $\delta < 1$

In this section we prove a weaker version of Theorem 4.1, 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 6.1 *Let $0 < \tau, \delta < 1$. There is an algorithm $\texttt{Multiplicity}'_{\text{large-}\delta}$ which takes as input a string $w \in \{0,1\}^k$ and access to independent traces of an unknown source string $x \in \{0,1\}^n$. $\texttt{Multiplicity}'_{\text{large-}\delta}$ runs in $\left(\frac{n^{1/(1-\delta)}}{1-\delta}\right)^{O(k)} \log\left(\frac{1}{\tau}\right)$ time and uses $\left(\frac{n^{1/(1-\delta)}}{1-\delta}\right)^{O(k)} \log\left(\frac{1}{\tau}\right)$ many traces from $\mathrm{Del}_\delta(x)$, and has the following property: For any unknown source string $x \in \{0,1\}^n$, with probability at least $1-\tau$ the output of $\texttt{Multiplicity}'_{\text{large-}\delta}$ is $\#(w,x)$, the multiplicity of w in $\mathrm{subword}(x,k)$ (equivalently, the value $\mathrm{SW}_{x,w}(0)$).*

Looking ahead, in Section 7 we will build on the proof of Theorem 6.1 to give a stronger version that has polynomial running time and sample complexity when $k \approx \log n$.

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 $\mathrm{SW}_{x,w}(0)$ by at least $1/2$, then q must "differ noticeably" from $\mathrm{SW}_{x,w}$ 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 $\mathrm{Del}_\delta(x)$.)

Theorem 6.2 *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 $|\mathrm{SW}_{x,w}(0) - q(0)| \ge 1/2$, and $0 \le q_\ell \le n^k$ for all $\ell \in \{0,1,\cdots,n-k\}$. Then*

$$\sup_{\zeta \in [\delta, (\delta+1)/2]} \big|\mathrm{SW}_{x,w}(\zeta) - q(\zeta)\big| \ge n^{-O\left(\frac{k}{1-\delta}\right)} \tag{6.3}$$

for any $\delta \in (0,1)$.

Theorem 6.2 is an easy consequence of the following more general theorem:

Theorem 6.3 *Let $1 \le n \le m$. Let $p(z) = \sum_{\ell=0}^{n} p_\ell z^\ell$ be a polynomial of degree at most n with real coefficients such that $|p_0| \ge 1/2$, and $|p_\ell| \le m$ for all ℓ. Then we have*

$$\sup_{\zeta \in [\delta, (\delta+1)/2]} \big|p(\zeta)\big| \ge m^{-O(1/(1-\delta))} \tag{6.4}$$

for any $\delta \in (0,1)$.

To obtain Theorem 6.2 from Theorem 6.3, set $p = \mathrm{SW}_{x,w} - q$. By the condition of Theorem 6.2 we have that $|p_0| = |\mathrm{SW}_{x,w}(0) - q_0| \geq 1/2$. Writing $(\mathrm{SW}_{x,w})_\ell$ for the degree-ℓ coefficient of $\mathrm{SW}_{x,w}$, from the discussion following Definition 3 it is immediate that $0 \leq (\mathrm{SW}_{x,w})_\ell \leq \binom{n}{k} \leq n^k$, and hence $|p_\ell| = |(\mathrm{SW}_{x,w})_\ell - q_\ell| \leq n^k$. Thus we can invoke Theorem 6.3 with $m = n^k$ to obtain Theorem 6.2.

In Section 6.1 we present and analyze the algorithm $\texttt{Multiplicity}'_{\text{large-}\delta}$ (which is based on linear programming) and prove Theorem 6.1 assuming Theorem 6.2. The proof of Theorem 6.3, which is based on complex analysis, is given in Section 6.2.

6.1 Proof of Theorem 6.1 assuming Theorem 6.2

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

Lemma 6.1 *Let $x \in \{0,1\}^n$, $w \in \{0,1\}^k$ and let $\varepsilon > 0$. Then, given traces $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$, there exists an algorithm, which for any $\delta' \in [\delta, 1]$, evaluates $\mathrm{SW}_{x,w}(\delta')$ 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 $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$, we can sample $\boldsymbol{y} \sim \mathrm{Del}_{\delta'}(x)$ for any $\delta' \geq \delta$ with no overhead. The algorithm simply draws

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

traces $\boldsymbol{y}^1, \cdots, \boldsymbol{y}^s \sim \mathrm{Del}_{\delta'}(x)$, and returns the estimator

$$\frac{1}{s(1-\delta')^k} \sum_{j=1}^{s} \#(w, \boldsymbol{y}^j).$$

Correctness. Observe that the expected number of w in a randomly trace $\boldsymbol{y} \sim \mathrm{Del}_{\delta'}(x)$ is given by

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\boldsymbol{y}\sim\mathrm{Del}_{\delta'}(x)}[\#(w,\boldsymbol{y})] \\
&\quad = \sum_{\substack{\alpha\in\mathbb{Z}_{\geq 0}^{k-1} \\ |\alpha|\leq n-k}} \#(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots \\
&\qquad w_{k-2} *^{\alpha_{k-1}} w_{k-1},\, x) \cdot \delta'^{|\alpha|} \cdot (1-\delta')^k.
\end{aligned}$$

This follows from the fact that every occurrence of w as a subword of trace y can be uniquely identified with a subsequence $(i_1 \leq \ldots \leq i_k)$ such that (i) $x_{i_1} = w_1 \wedge \ldots \wedge x_{i_k} = w_k$. (ii) positions $i_1, \ldots, i_k$ are not deleted in $\boldsymbol{y}$. (iii) every position in $[i_1, \ldots, i_k] \setminus \{i_1, \ldots, i_k\}$ is deleted in the trace $\boldsymbol{y}$. However, by Definition 3, it follows that

$$\mathop{\mathbf{E}}_{\boldsymbol{y}\sim\mathrm{Del}_{\delta'}(x)}[\#(w,\boldsymbol{y})] = \mathrm{SW}_{x,w}(\delta') \cdot (1-\delta')^k. \tag{6.5}$$

Now for any $\boldsymbol{y} \sim \mathrm{Del}_{\delta'}(x)$, $\#(w,\boldsymbol{y})$ is an integer between 0 and n. So, the output is an estimate of $\mathbf{E}_{\boldsymbol{y}\sim\mathrm{Del}_{\delta'}(x)}[\#(w,\boldsymbol{y})]/(1-\delta')^k$ up to $\pm\varepsilon$. Using (6.5), we get the claim. ∎

Inputs
$w \in \{0,1\}^k$
access to independent traces drawn from $\mathrm{Del}_\delta(x)$ for an unknown string $x \in \{0,1\}^n$
error parameter $\tau \in (0,1)$

Output
$\#(w,x)$ or "fail"

Algorithm description

1. Let $\kappa := n^{-O(k/(1-\delta))}$ be the RHS of Equation (6.3) in Theorem 6.2, let $\Delta := \kappa/(2n^{k+2})$, and let
$$S := \{\delta, \delta+\Delta, \delta+2\Delta, \ldots, \delta+L\Delta\}$$
such that L is the largest integer with $\delta + L\Delta \leq (\delta+1)/2$. (Note that $|S| = O(1/\Delta)$.)
2. For each $\zeta \in S$, compute the empirical estimate $\widehat{\mathrm{SW}}_{x,w}(\zeta)$ of $\mathrm{SW}_{x,w}(\zeta)$ up to accuracy $\kappa/5$ with correctness probability $1 - \tau/|S|$ using Lemma 6.1. (We reuse traces from $\mathrm{Del}_\delta(x)$ for each $\zeta \in S$.)
3. Set up a linear program as follows:
   (a) Variables are $q_0, \ldots, q_{n-k} \in [0, n^k]$.
   (b) Constraints are: For each $\zeta \in S$,
   $$\left|\sum_{\ell=0}^{n-k} q_\ell \zeta^\ell - \widehat{\mathrm{SW}}_{x,w}(\zeta)\right| \leq \kappa/5.$$
4. Return "fail" if the above linear program has no solution.
5. Otherwise solve the linear program and return the nearest integer to q_0.

Figure 1: Description of the algorithm $\texttt{Multiplicity}'_{\text{large-}\delta}$.

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

$$\text{for every } \zeta \in S, \quad \left|\widehat{\mathrm{SW}}_{x,w}(\zeta) - \mathrm{SW}_{x,w}(\zeta)\right| \le \kappa/5.$$

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

To see that the linear program is feasible, we let $p_0, \ldots, p_{n-k}$ denote the coefficients of $\mathrm{SW}_{x,w}$, so $\mathrm{SW}_{x,w}(\zeta) = \sum_{\ell=0}^{n-k} p_\ell \zeta^\ell$. From the discussion after Theorem 6.3, every p_ℓ 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 $\zeta \in S$,

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

Next we let $q_0, \ldots, q_{n-k}$ be any feasible solution to the linear program and assume for a contradiction that $|q_0 - \mathrm{SW}_{x,w}(0)| \ge 1/2$. Let q be the polynomial $q(\zeta) = \sum_{\ell=0}^{n-k} q_\ell \zeta^\ell$. Given that $0 \le q_\ell \le n^k$ for every ℓ (as required by the linear program), Theorem 6.2 implies (using the choice of κ in line 1 of the algorithm) that

$$\sup_{\zeta \in [\delta, (\delta+1)/2]} \left|\mathrm{SW}_{x,w}(\zeta) - q(\zeta)\right| \ge \kappa. \tag{6.6}$$

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

$$\left|\mathrm{SW}_{x,w}(\zeta) - q(\zeta)\right| \ge \kappa/2,$$

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

$$\begin{aligned}
&\left|\sum_{\ell=0}^{n-k} q_\ell \zeta^\ell - \widehat{\mathrm{SW}}_{x,w}(\zeta)\right| \\
&= \left|q(\zeta) - \widehat{\mathrm{SW}}_{x,w}(\zeta)\right| \\
&\ge \left|q(\zeta) - \mathrm{SW}_{x,w}(\zeta)\right| - \left|\mathrm{SW}_{x,w}(\zeta) - \widehat{\mathrm{SW}}_{x,w}(\zeta)\right| \\
&> \kappa/5.
\end{aligned}$$

Claim 6.2 (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_ℓ has $|s_\ell| \le m$. Suppose $|s(t_0)| \ge \kappa$ for some $t_0 \in [\delta, (\delta+1)/2]$. Then there exists an integer k such that $t' = \delta + k\Delta \in [\delta, (\delta+1)/2]$ and $|s(t')| \ge \kappa/2$, where $\Delta = \kappa/(2mn^2)$.*

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

$$|t'^\ell - t_0^\ell| \le |t' - t_0| \cdot \sum_{i=0}^{\ell-1} \left|t'^i t_0^{\ell-1-i}\right| \le \Delta\ell \le \Delta n.$$

Since $|s_\ell| \le m$ and $\Delta = \kappa/(2mn^2)$, we have

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

Therefore

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

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

We now analyze the complexity of the algorithm. Note that for all $\zeta \in S$, we have $1 - \zeta \ge (1-\delta)/2$. By Lemma 6.1, the sample complexity is

$$\begin{aligned}
&n^{O(1)} \cdot \left(\frac{2}{1-\delta}\right)^{O(k)} \cdot \left(\frac{5}{\kappa}\right)^2 \cdot \log\left(\frac{|S|}{\tau}\right) \\
&\qquad = \left(\frac{n^{1/(1-\delta)}}{1-\delta}\right)^{O(k)} \log\left(\frac{1}{\tau}\right).
\end{aligned} \tag{6.7}$$

The running time of the algorithm is (6.7) 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 (6.7) 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 (6.7). This proves the claimed upper bounds on the running time and sample complexity, and concludes the proof of Theorem 6.1 assuming Theorem 6.2.

6.2 Proof of Theorem 6.3 In this subsection we prove Theorem 6.3. For convenience we define $\rho := 1 - \delta \in (0,1)$, and we restate the theorem below in terms of ρ:

Restatement of Theorem 6.3: *Let $1 \le n \le m$. Let $p(z) = \sum_{i=0}^{n} p_i z^i$ be a polynomial of degree at most n with real coefficients such that $|p_0| \ge 1/2$, and $|p_i| \le m$ for all i. Then for any $\rho \in (0,1)$,*

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

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 [24] on f, which tells us that

$$2 \log \left(\sup_{z \in C_2} |f(z)| \right) \leq \log \left(\sup_{z \in C_1} |f(z)| \right) + \log \left(\sup_{z \in C_3} |f(z)| \right). \tag{6.8}$$

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:

Fact 6.1 *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 $[1 - \rho, 1 - \rho/2]$.

(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 $z \in C_3$, $w(z)$ 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 (6.8) will be useful to us because of the following simple claim, which is immediate from Fact 6.1, Item (1):

Claim 6.3

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

Given Equation (6.8) and Claim 6.3, in order to lower bound $\sup_{\zeta \in [1-\rho, 1-\rho/2]} |p(\zeta)|$, it suffices to upper bound $\sup_{z \in C_3} |f(z)|$ and to lower bound $\sup_{z \in C_2} |f(z)|$. We do this in the following claims:

Claim 6.4

$$\sup_{z \in C_3} |f(z)| \leq m \cdot (n + 1).$$

Proof. By Fact 6.1, Item (3) above, we have $E_3 \subseteq B_1(0)$ and so

$$\sup_{z \in C_3} |f(z)| = \sup_{z \in E_3} |p(z)| \leq \sup_{z \in B_1(0)} |p(z)|,$$

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

Claim 6.5

$$\sup_{z \in C_2} |f(z)| \geq m^{-O(1/\rho)}.$$

Proof. Applying Jensen's formula [25] to p on the closed origin-centered disk of radius $1 - 3\rho/4$, we get that

$$\mathbf{E}_{\boldsymbol{z}}[\ln |p(\boldsymbol{z})|] \geq \ln |p(0)| \geq \ln(1/2) = -\ln 2. \tag{6.9}$$

Here $\boldsymbol{z}$ 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

$$\mathcal{A} = \{z \in \mathbb{C} : |z| = 1 - 3\rho/4 \text{ and } |\arg(z)| \leq 3\rho/16\}.$$

Let $c_{\max,\mathcal{A}} = \max_{z \in \mathcal{A}} |p(z)|$ and $\theta^* = 3\rho/16$ (note that θ^*/π 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 (6.9) that

$$\begin{aligned} -\ln 2 &\leq \left(1 - \frac{\theta^*}{\pi}\right) \ln\left(m(n+1)\right) + \frac{\theta^*}{\pi} \cdot \ln c_{\max,\mathcal{A}} \\ &\leq \ln\left(m(n+1)\right) + \frac{\theta^*}{\pi} \cdot \ln c_{\max,\mathcal{A}}. \end{aligned}$$

Thus,

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

and hence

$$c_{\max,\mathcal{A}} \geq \left(2m(n+1)\right)^{-\pi/\theta^*}.$$

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 and Fact 6.1 Item (2), we conclude that

$$\sup_{z\in C_2}|f(z)| = \sup_{z\in E_2}|p(z)| \geq \sup_{z\in\mathcal{A}}|p(z)| = c_{\max,\mathcal{A}} \geq m^{-O(1/\rho)},$$

and Claim 6.5 is proved. ∎

Proof. (Proof of Theorem 6.3) We combine Claims 6.3 to 6.5 in Equation (6.8) to get that

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

Exponentiating both sides finishes the proof of Theorem 6.3. ∎

7 Improved algorithms: Proof of Theorem 4.1

In this section we give improved algorithms strengthening the quantitative bounds given in Theorem 5.1 and Theorem 6.1 and thereby complete the proof of Theorem 4.1.

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 6.1 in this overview. Recall the definition of the subword polynomial $\mathrm{SW}_{x,w}$ from Definition 3:

$$\mathrm{SW}_{x,w}(\zeta) := \sum_{\substack{\alpha\in\mathbb{Z}_{\geq 0}^{k-1}\\ |\alpha|\leq n-k}} \#\left(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2\cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1},\, x\right)\cdot\zeta^{|\alpha|}.$$

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

$$\gamma_\ell = \sum_{\substack{\alpha\in\mathbb{Z}_{\geq 0}^{k-1}\\ |\alpha|=\ell}} \#\left(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2\ldots w_{k-2} *^{\alpha_{k-1}} w_{k-1},\, x\right)$$

is the degree-ℓ coefficient, for each $0\leq\ell\leq n-k$. In the proofs of Corollary 5.1 in Section 5 and Theorem 6.2 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

$$0\leq\gamma_\ell\leq m_\ell := n\binom{\ell+k-2}{k-2}. \tag{7.10}$$

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 $\alpha_1,\cdots,\alpha_{k-1}$ that sum to ℓ. The second insight is that since our approaches only involve evaluating $\mathrm{SW}_{x,w}(\zeta)$ on non-negative real inputs ζ 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 5 *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 $p^{\leq d}$) to be*

$$p^{\leq d}(\zeta) = \sum_{\ell=0}^{d} p_\ell\,\zeta^\ell.$$

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

Consider any polynomial q with a constant term which is an integer different from $\mathrm{SW}_{x,w}(0)$. 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 (7.10). 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 $\mathrm{SW}_{x,w}$ and q must differ by at least

$$\left(\frac{1}{n}\left(\frac{1-\delta}{2}\right)^k\right)^{O(1/(1-\delta))}$$

(see Equation (7.16)) at some point in the interval $[\delta,(\delta+1)/2]$. This result is stronger than the analogous $\approx n^{-O(k/(1-\delta))}$ lower bound established in Theorem 6.2, 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.

[2] We ignore the dependence on δ for the overview here; see (7.11) and (7.15) for exact choices of d.

In Section 7.1, we implement these ideas to strengthen Theorem 5.1 when $\delta < 1/2$. In Section 7.2, we do the same to derive a stronger analogue of Theorem 6.2, 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 5.1 for deletion rate $\delta < 1/2$

In this subsection we strengthen Theorem 5.1 for deletion rate $\delta < 1/2$ as follows:

Theorem 7.1 *Let $0 < \delta < 1/2$. There is an algorithm $\texttt{Multiplicity}_{\textit{small-}\delta}$ 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 $\tau > 0$. $\texttt{Multiplicity}_{\textit{small-}\delta}$ draws $\mathrm{poly}(n) \cdot (1/2-\delta)^{-O(k)} \cdot \log(1/\tau)$ traces from $\mathrm{Del}_\delta(x)$, runs in time $\mathrm{poly}(n) \cdot (1/2-\delta)^{-O(k)} \cdot \log(1/\tau)$, and has the following property: For any unknown source string $x \in \{0,1\}^n$, with probability at least $1-\tau$ the output of $\texttt{Multiplicity}_{\textit{small-}\delta}$ is the multiplicity of w in $\mathrm{subword}(x,k)$ (i.e. the number of occurrences of w as a subword of x).*

Recall Theorem 5.2, which relates the subword polynomial value at any point $\zeta \in \mathbb{C}$ to traces drawn from the deletion channel using Taylor series:

$$\begin{aligned}&\mathrm{SW}_{x,w}(\zeta)\\ &= \frac{1}{(1-\delta)^k} \sum_{\substack{\alpha \in \mathbb{Z}_{\geq 0}^{k-1} \\ |\alpha| \leq n-k}} \mathbf{E}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \Big[\#(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots \\ &\qquad\qquad w_{k-2} *^{\alpha_{k-1}} w_{k-1}, \boldsymbol{y}) \Big] \cdot \left(\frac{\zeta - \delta}{1-\delta} \right)^{|\alpha|}.\end{aligned}$$

As in Section 7.1, our goal is to evaluate $\mathrm{SW}_{x,w}(0) = \#(w,x)$ 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:

$$p(\xi) := (1-\delta)^k \cdot \mathrm{SW}_{x,w}\big(\delta + \xi(1-\delta)\big).$$

We have that $\mathrm{SW}_{x,w}(0) = (1-\delta)^{-k} p(-\delta/(1-\delta))$, so estimating $\mathrm{SW}_{x,w}(0)$ up to error $\pm 1/3$ is equivalent to estimating $p(-\delta/(1-\delta))$ up to error $\pm(1-\delta)^k/3$. 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. Let $\theta = 1/2 - \delta$, so that $\delta/(1-\delta) \leq 2\delta = 1 - 2\theta$.

Lemma 7.1 *Let $\delta < 1/2$, and let p and θ be as above. Then by setting*

$$d := \frac{C}{\theta}\left(k \ln \frac{C}{\theta} + \ln n\right) \tag{7.11}$$

with $C = e^2$, we have

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

Before proving Lemma 7.1, we show that it implies Theorem 7.1.

Proof. (Proof of Theorem 7.1 assuming Lemma 7.1) Consider $p^{\leq d}$, the d-low-degree-part of p, where d is as given by Lemma 7.1. For all ξ with $|\xi| \leq 1-2\theta$,

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

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

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

$$E_\alpha := \mathop{\mathbf{E}}_{\boldsymbol{y} \sim \mathrm{Del}_\delta(x)} \Big[\#(w_0 *^{\alpha_1} w_1 *^{\alpha_2} w_2 \cdots w_{k-2} *^{\alpha_{k-1}} w_{k-1}, \boldsymbol{y}) \Big]$$

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

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

Each E_α is between 0 and n and using the same argument as that following Equation (7.10), we have

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

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

Claim 7.2 *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*

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

Proof. Using $d \geq k$ and the approximation $k! \geq$

$\sqrt{2\pi k}(k/e)^k \geq (k/e)^k$, we have

$$\begin{aligned}\binom{d+k}{k} &\leq \frac{(2d)^k}{(k/e)^k} = \exp\left(k\ln\frac{2ed}{k}\right)\\ &\leq \exp\left(k\left(2+\ln\frac{C}{\theta}+\ln\left(\ln\frac{C}{\theta}+\frac{\ln n}{k}\right)\right)\right)\\ &\leq \exp\left(k\left(2+\ln\frac{C}{\theta}+\ln\frac{C}{\theta}+\frac{\ln n}{k}\right)\right) \qquad (7.12)\\ &\leq n\cdot\left(\frac{C}{\theta}\right)^{3k}, \qquad (7.13)\end{aligned}$$

where (7.12) used $\ln a \leq a$, (7.13) used $2 < \ln(C/\theta)$ since $C \geq e^2$. ∎

Plugging in Claim 7.2, we have $M \leq n/\theta^{O(k)}$ using $\theta < 1/2$. The algorithm just draws s (to be specified) traces $\boldsymbol{y} \sim \mathrm{Del}_\delta(x)$, computes an empirical estimate $\tilde{E}_\alpha$ of E_α for each $\alpha \in S_d$ so that

$$\left|\tilde{E}_\alpha - E_\alpha\right| \leq \frac{0.2}{2^k M}.$$

with probability at least $1-\tau$. This can be achieved by setting the number of traces to be

$$s := O\left(\left(M^2 2^k\right)^2\right)\cdot\log\left(\frac{M}{\tau}\right) = \left(\frac{n}{\theta^k}\right)^{O(1)}\cdot\log\frac{1}{\tau}$$

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

$$\sum_{\alpha\in S_d}\tilde{E}_\alpha\cdot\left(\frac{-\delta}{1-\delta}\right)^{|\alpha|}$$

is an estimate that deviates by at most $2^{-k}/5$. Combined with the observations at the beginning of the proof, this implies that we can estimate $\mathrm{SW}_{x,w}(0) = \#(w,x)$ up to error $\pm 1/3$, and hence our output (the nearest integer to our estimate of $\mathrm{SW}_{x,w}(0)$) is $\#(w,x)$ with probability at least $1-\tau$.

The runtime is governed by the time required to compute estimates $\tilde{E}_\alpha$. We can bound it by

$$\begin{aligned}s\cdot n^{O(1)}\cdot|S_d| &\leq \left(\frac{n}{\theta^k}\right)^{O(1)}\cdot\log\frac{1}{\tau}\\ &= n^{O(1)}\cdot\left(\frac{1}{1/2-\delta}\right)^{O(k)}\cdot\log\frac{1}{\tau}.\end{aligned}$$

This finishes the proof of the theorem. ∎

Proof. (Proof of Lemma 7.1) We are interested in $|p^{>d}(\xi)|$ over $|\xi| \leq 1-2\theta$, which is trivially bounded by

$$\begin{aligned}|p^{>d}(\xi)| &\leq \sum_{\ell=d+1}^{n-k} n\binom{\ell+k-2}{k-2}\cdot(1-2\theta)^\ell\\ &\leq \sum_{\ell=d}^{n-k} n\binom{\ell+k}{k}\cdot(1-2\theta)^\ell.\end{aligned}$$

First, we show that terms in the sum on the right hand side above decreases with ℓ so it suffices to bound the term with $\ell = d$ multiplied by n. To see this, observe that

$$\left|\frac{\binom{\ell+k}{k}}{\binom{\ell+k-1}{k}}\cdot(1-2\theta)\right| = \frac{\ell+k}{\ell}\cdot(1-2\theta) \leq 1+\frac{k}{\ell}-2\theta < 1,$$

whenever $\ell > k/2\theta$, which holds for all $\ell > d$ given our choice of d. So,

$$\begin{aligned}\sup_{|\xi|\leq 1-2\theta}\left|p^{>d}(\xi)\right| &\leq n^2\binom{d+k}{k}(1-2\theta)^d\\ &\leq n^2\binom{d+k}{k}e^{-2\theta d}.\end{aligned}$$

We have $e^{-2\theta d} = n^{-2C}\cdot(C/\theta)^{-2Ck}$, and so plugging in Claim 7.2 we have

$$n^2\cdot\left(n\cdot\left(C/\theta\right)^{3k}\right)\cdot e^{-2\theta d} \leq n^{3-2C}\cdot\left(C/\theta\right)^{(3-2C)k} \leq \frac{1}{n2^k}$$

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

7.2 Improvement of Theorem 6.2 for deletion rate $\delta < 1$ Our main technical result is the following, which is a strengthening of Theorem 6.2:

Theorem 7.2 *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 $|\mathrm{SW}_{x,w}(0) - q(0)| \geq 1/2$ and $0 \leq q_\ell \leq m_\ell$ for all $\ell \in \{0,1,\cdots,n-k\}$. Then for any $\delta \in (0,1)$,*

$$\sup_{\zeta\in[\delta,(\delta+1)/2]}\left|\mathrm{SW}_{x,w}(\zeta) - q(\zeta)\right| \geq \left(\frac{1}{n}\left(\frac{1-\delta}{2}\right)^k\right)^{O\left(\frac{1}{1-\delta}\right)}. \qquad (7.14)$$

Let $p(z) = \mathrm{SW}_{x,w}(z) - q(z) = \sum_{\ell=0}^{n-k} p_\ell z^\ell$. Let $c > 0$ be the constant hidden in the exponent of the RHS of Equation (6.4) in Theorem 6.3. Let $\theta = (1-\delta)^2/2$. We will choose the threshold on the degree to be

$$d := \frac{C}{\theta}\left(k\ln\frac{C}{\theta}+\ln n\right) \qquad (7.15)$$

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

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

for all $\ell \le d$. We invoke Theorem 6.3 on $p^{\le d}$ to conclude that

$$\sup_{\zeta \in [\delta,(\delta+1)/2]} |p^{\le d}(\zeta)| \ge \left(n\binom{d+k}{k}\right)^{-c/(1-\delta)}. \tag{7.16}$$

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

Lemma 7.3 *Let p and d be as above. Then*

$$\sup_{\zeta \in [\delta,(\delta+1)/2]} |p^{>d}(\zeta)| \le \frac{1}{n} \cdot \left(n\binom{d+k}{k}\right)^{-c/(1-\delta)}. \tag{7.17}$$

Before proving this lemma, we show that it implies Theorem 7.2.

Proof. (Proof of Theorem 7.2 using Lemma 7.3) Since $p = p^{\le d} + p^{>d}$, we use Lemma 7.3 and (7.16) to get

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

Plugging in Claim 7.2 with our choice of d, we have

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

which concludes the proof of Theorem 7.2 using Lemma 7.3. ∎

Proof. (Proof of Lemma 7.3) This proof is similar to that of Lemma 7.1. First we show that the maximum possible contribution to $p^{>d}(\zeta)$, when $\zeta \in [\delta, (\delta+1)/2]$, arises from the degree-d term in p:

$$\begin{aligned}\left|\frac{\binom{\ell+k}{k}}{\binom{\ell+k-1}{k}} \cdot \zeta\right| &= \frac{\ell+k}{\ell} \cdot |\zeta| \\ &\le \left(1 + \frac{k}{\ell}\right)\left(1 - \frac{1-\delta}{2}\right) \\ &\le 1 + \frac{k}{\ell} - \frac{1-\delta}{2} < 1\end{aligned}$$

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

$$\begin{aligned}\sup_{|\zeta| \le (\delta+1)/2} |p^{>d}(\zeta)| &\le n^2\binom{d+k}{k}\left(1 - \frac{1-\delta}{2}\right)^d \\ &\le n^2\binom{d+k}{k} \cdot \exp\left(-\frac{(1-\delta)d}{2}\right).\end{aligned}$$

It suffices to show that

$$n^2\binom{d+k}{k} \cdot \exp\left(-\frac{(1-\delta)d}{2}\right) \le \frac{1}{n}\left(n\binom{d+k}{k}\right)^{-\frac{c}{1-\delta}}$$

or equivalently,

$$n^{3+\frac{2c}{1-\delta}} \cdot \binom{d+k}{k}^{1+\frac{c}{1-\delta}} \cdot \exp\left(-\frac{(1-\delta)d}{2}\right) \le 1. \tag{7.18}$$

By our choice of d we have

$$\exp\left(-\frac{(1-\delta)d}{2}\right) \le n^{-\frac{C}{1-\delta}} \cdot (C/\theta)^{-\frac{kC}{1-\delta}}.$$

Using Claim 7.2 again, the left hand side of Equation (7.18) is at most

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

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

7.3 The algorithm of Theorem 4.1 Armed with Theorem 7.2 in place of Theorem 6.2, the algorithm $\texttt{Multiplicity}_{\text{large-}\delta}$ giving Theorem 4.1 and its analysis are very similar to the algorithm $\texttt{Multiplicity}'_{\text{large-}\delta}$ 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 κ to be the RHS of Equation (7.14):

$$\kappa := \left(\frac{1}{n}\left(\frac{1-\delta}{2}\right)^k\right)^{O(1/(1-\delta))}.$$

  With this choice of κ, it follows from the proof of Theorem 7.2 that the RHS of Equation (7.17) in Lemma 7.3 can be bounded from above by 0.01κ.

- Later in Line 1, we now set

$$\Delta := \frac{\kappa}{2d^2\, m_d} = \frac{\kappa}{2d^2 \cdot n\binom{d+k-2}{k-2}},$$

  where d is as given in Equation (7.15) (the idea is that now we are using the sharper coefficient bound $m_\ell \le m_d$ given by Equation (7.10) rather than the cruder n^k 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

$$\text{for every } \zeta \in S, \quad \left|\widehat{\mathrm{SW}}_{x,w}(\zeta) - \mathrm{SW}_{x,w}(\zeta)\right| \le \kappa/5.$$

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

Now we show that every feasible solution $q_0, \cdots, q_{n-k}$ to the linear program must satisfy $|q_0 - \mathrm{SW}_{x,w}(0)| < 1/2$; 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 $|q_0 - \mathrm{SW}_{x,w}(0)| \ge 1/2$. Let $q(\zeta) = \sum_\ell q_\ell \zeta^\ell$ and define the polynomial $p = \mathrm{SW}_{x,w} - q$, with coefficients p_ℓ. We invoke Theorem 7.2 to get that $|p(\zeta^*)| \ge \kappa$ for some $\zeta^* \in [\delta, (\delta+1)/2]$. By Lemma 7.3 (and the remark below the choice of κ),

$$|p(\zeta) - p^{\le d}(\zeta)| = |p^{>d}(\zeta)| \le 0.01\kappa \tag{7.19}$$

for all $\zeta \in [\delta, (\delta+1)/2]$. As a result, we have $|p^{\le d}(\zeta^*)| \ge 0.99\kappa$. Applying Claim 6.2 with $s = p^{\le d}$, $n = d$, $t_0 = \zeta^*$, $m = m_d$ and our choice of Δ, there exists a $\zeta' \in S$ such that $|p^{\le d}(\zeta')| \ge 0.495\kappa$ and thus, $|p(\zeta')| \ge |p^{\le d}(\zeta')| - |p^{>d}(\zeta')| \ge 0.485\kappa$. Hence, recalling that $p = \mathrm{SW}_{x,w} - q$, we have

$$\begin{aligned}\left|\widehat{\mathrm{SW}}_{x,w}(\zeta') - q(\zeta')\right| &\ge |p(\zeta')| - \left|\widehat{\mathrm{SW}}_{x,w}(\zeta') - \mathrm{SW}_{x,w}(\zeta')\right| \\ &\ge 0.285\kappa > \kappa/5.\end{aligned}$$

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 $1 - \zeta \ge (1-\delta)/2$. So, by Lemma 6.1, the sample complexity is at most

$$s = \frac{n^{O(1)}}{\kappa^2}\left(\frac{2}{1-\delta}\right)^{O(k)} \log\left(\frac{|S|}{\tau}\right) \tag{7.20}$$

$$= \left(n\left(\frac{2}{1-\delta}\right)^k\right)^{O(1/(1-\delta))} \log\frac{1}{\tau}. \tag{7.21}$$

The running time is dominated by the time required to compute $\widehat{SW}_{x,w}(\zeta)$ for each $\zeta \in S$. The running time for each ζ can be bounded by (7.20) and the same expression can be used to bound the overall running time given the bound on $|S|$ above.

References

[1] 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. 1

[2] 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. 1

[3] T. Batu, S. Kannan, S. Khanna, and A. McGregor. Reconstructing strings from random traces. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 910–918, 2004. 1, 1.1

[4] Z. Chase. New lower bounds for trace reconstruction. *CoRR*, abs/1905.03031, 2019. (document), 1, 1.1, 1.5

[5] Zachary Chase. New upper bounds for trace reconstruction, 2020. (document)

[6] 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. (document), 1, 1.1, 1.3, 1.5, 5.2

[7] 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. (document), 1

[8] N. Holden and R. Lyons. Lower bounds for trace reconstruction. *CoRR*, abs/1808.02336, 2018. (document), 1.1

[9] Nina Holden, Robin Pemantle, and Yuval Peres. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. *CoRR*, abs/1801.04783, 2018. (document), 1, 1.1, 1.3, 1.5

[10] Nina Holden, Robin Pemantle, Yuval Peres, and Alex Zhai. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. *CoRR*, abs/1801.04783, 2020. 1, 1.1, 1.3, 1.5

[11] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. 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. 1, 1.1, 1.3

[12] V. V. Kalashnik. Reconstruction of a word from its fragments. *Computational Mathematics and Computer Science (Vychislitel'naya matematika i vychislitel'naya tekhnika), Kharkov*, 4:56–57, 1973. 1

[13] 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. 1, 1.1

[14] 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. 1

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

[16] Vladimir Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. *Journal of Combinatorial Theory Series A*, 93(2):310–332, 2001. 1

[17] Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace reconstruction revisited. In *Proceedings of the 22nd Annual European Symposium on Algorithms*, pages 689–700, 2014. 1, 1.1

[18] S. Narayanan. Population recovery from the deletion channel: Nearly matching trace reconstruction bounds. *CoRR*, abs/2004.06828, 2020. 1

[19] Fedor Nazarov and Yuval Peres. Trace reconstruction with $\exp\left(o(n^{1/3})\right)$ samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1042–1046, 2017. (document), 1, 1.1, 1.3, 1.5

[20] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. 2

[21] Yuval Peres and Alex Zhai. Average-case reconstruction for the deletion channel: subpolynomially many traces suffice, 2017. Available at https://arxiv.org/abs/1708.00854. (document), 1, 1.1, 1.3

[22] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing (STOC) 2001*, pages 296–305. ACM, 2001. 1.2

[23] 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. 1, 1.1

[24] Wikipedia contributors. Hadamard three-circle theorem. Wikipedia, The Free Encyclopedia, Accessed June 28, 2020. https://en.wikipedia.org/wiki/Hadamard_three-circle_theorem. 6.2

[25] Wikipedia contributors. Jensen's formula. Wikipedia, The Free Encyclopedia, Accessed June 28, 2020. https://en.wikipedia.org/wiki/Jensen 6.2