Randomize the Future: Asymptotically Optimal Locally Private Frequency Estimation Protocol for Longitudinal Data

Olga Ohrimenko
oohrimenko@unimelb.edu.au
The University of Melbourne
Australia

Anthony Wirth
awirth@unimelb.edu.au
The University of Melbourne
Australia

Hao Wu
whw4@student.unimelb.edu.au
The University of Melbourne
Australia

ABSTRACT
Longitudinal data tracking under Local Differential Privacy (LDP) is a challenging task. Baseline solutions that repeatedly invoke a protocol designed for one-time computation lead to linear decay in the privacy or utility guarantee with respect to the number of computations. To avoid this, the recent approach of Erlingsson et al. (2020) exploits the potential sparsity of user data that changes only infrequently. Their protocol targets the fundamental problem of frequency estimation for longitudinal binary data, with error bound scales polylogarithmically with $\log(d)$, $\log(k)$, and $\sqrt{n \cdot \log(d/k)}$, where $\varepsilon$ is the privacy budget, $d$ is the number of time periods, $k$ is the maximum number of changes of user data, and $\beta$ is the failure probability. Notably, the error bound scales polylogarithmically with $d$, but linearly with $k$.

In this paper, we break through the linear dependence on $k$ in the estimation error. Our new protocol has error $O((1/\varepsilon) \cdot (\log d) \cdot \sqrt{k \cdot n \cdot \log(d/k)})$, matching the lower bound up to a logarithmic factor. The protocol is an online one, that outputs an estimate at each time period. The key breakthrough is a new randomizer for sequential data, FutureRand, with two key features. The first is a composition strategy that correlates the noise across the non-zero elements of the sequence. The second is a pre-computation technique that enables randomizer to output the results on the fly, without knowing future inputs. Our protocol closes the error gap between existing online and offline algorithms.

CCS CONCEPTS
- Security and privacy → Privacy-preserving protocols.

KEYWORDS
Local Differential Privacy, Longitudinal Data, Frequency Estimation

1 INTRODUCTION
Frequency estimation underlines a wide range of applications in data mining and machine learning (for example, learning users’ preferences, uncovering commonly used phrases, and finding popular URLs). However, the data collected for the frequency analysis can contain sensitive personal information such as income, gender, health information, etc. To protect this information from the data collector, the local model of differential privacy (LDP) has been deployed by companies including Google [9, 10], Apple [15] and Microsoft [5]. In LDP, each user perturbs their data locally before reporting it to the (untrusted) data collector (aka server) for analysis.

Existing solutions typically focus on one-time computation. However, practical applications often involve continuous monitoring in order to discover trends over time. For example, search-engine providers keep track of popular URLs. The naive solution of repeated computation leads to a rapid degradation of privacy guarantee, that scales linearly with the number of computations [7]. But such degradation is unnecessary, when users’ data changes infrequently. For example, a list of frequently visited URLs by a user changes little everyday.

The observation of infrequent data changes is captured by Erlingsson et al. [19]. Noting that many LDP algorithms [1, 2, 9–11] require each client to send just one bit to the server, they formulated the following longitudinal data tracking problem for Boolean data.

Research Problem: Given a set of $n$ users, each holding a Boolean value that changes at most $k$ times across $d$ time periods; the server needs to report the number of users holding Boolean value 1 at each time period.

The problem is presented in an online setting; in an offline setting, the server reports an estimate only after $d$ time periods. Though here we study the problem over Boolean data, our algorithm can be adapted to solve frequency estimation and heavy hitter problems in richer domains via existing techniques [1, 2, 9–11].

For privacy budget $\varepsilon$ and failure probability $\beta$, Erlingsson et al. described a protocol that achieves maximum estimation error $O((1/\varepsilon) \cdot (\log d)^{3/2} \cdot k \cdot \sqrt{n \cdot \log(d/k)})$ [19], and scales only linearly with $k$ and not $d$. In this paper we study the following question:

Research Question: Can the estimation error for the (online) longitudinal data tracking problem above be reduced to sub-linear in $k$?

Besides improvement on previous work in terms of the error, answering this question can close the gap between error guarantees of the online algorithm and the lower bound of $O((1/\varepsilon) \cdot \sqrt{k \cdot n \cdot \log(d/k)})$ that was recently presented in [18].

Our Contributions. In this work, we provide a new LDP protocol whose error scales with the square root of the number of data changes. Specifically, it achieves error

$$O((1/\varepsilon) \cdot (\log d) \cdot \sqrt{k \cdot n \cdot \log(d/k)}).$$

Our protocol builds on a standard technique for converting the original data sequence into a sparse one [19], hierarchical aggregation scheme for releasing continual data [6], and a new randomizer, FutureRand, for sequential data. The FutureRand has two key components, a composed randomizer $R: \{-1, 1\}^k \rightarrow \{-1, 1\}^k$ for the non-zero coordinates of the input sequence, and a pre-computation technique that enables $R$ to handle online inputs. The main properties of $R$, which play vital role in establishing privacy guarantee and achieving a $\sqrt{k}$ estimation error, are stated below: assuming that $\varepsilon \leq 1$.
We present formally the longitudinal data tracking problem first locally by an \( \epsilon \)-locally differentially private if for all \( \hat{b} \in \{-1, 1\}^k \), and each sequence \( s \in \{-1, 1\}^k \), then

\[
\Pr[\hat{R}(b) = s] \in [p_{\min}', p_{\max}'].
\]

- For a given input \( b \), denote \( \tilde{b} \) is the output of \( \hat{R} \). There exists some \( \epsilon_{\text{gap}} \in \Omega(\epsilon/\sqrt{k}) \), such that for each input \( b \in \{-1, 1\}^k \), and for each \( i \in [k] \),

\[
\Pr[\tilde{b}_i = b_i] - \Pr[\tilde{b}_i = -b_i] = \epsilon_{\text{gap}} \in \Omega(\epsilon/\sqrt{k}).
\]

\( \hat{R} \) builds on the composed randomizer proposed by Bun et al. [2]. The design in [2] focused on preserving the statistical distance between the distribution of the output of the composed randomizer, and joint distribution of \( k \) independent randomized responses. This difference in the problem setting requires non-trivial changes in terms of parameters, assumptions and analysis. The proof in [2] relies extensively on the concentration and anti-concentration inequalities; in comparison, our proof avoids this and investigates the inherent structure of the problem. Finally, Bun et al.’s original design [2] applies only to offline inputs. Our paper develops the pre-computation technique for converting it into an online one.

**Organization.** Our paper is organized as follows. Section 2 introduces the problem formally. Section 3 discusses the preliminaries for our protocol. Section 4 develops an algorithmic framework for longitudinal data tracking. Section 5 introduces the FutureRand. Section 6 summarizes the related works.

## 2 PROBLEM DEFINITION

We present formally the longitudinal data tracking problem first introduced by Erlingsson et al. [19]. There is a server and a set of \( n \) users, each holding a Boolean data item that changes at most \( \log d \) times across the \( d \) time periods. Without loss of generality, we assume that \( d \) is a power of \( 2 \). For user \( u \in [n] \), denote the value sequence of its Boolean data across the \( d \) time periods as

\[ s_{tu} = (s_{tu}[1], \ldots, s_{tu}[d]) \in \{0,1\}^d. \]

For each time \( t \in [d] \), denote the number of users with value \( 1 \) by

\[ a[t] = \sum_{u \in [n]} s_{tu}[t]. \]  

**Definition 2.1 ((\( \alpha, \beta \))-Accurate Protocol).** Given \( \alpha > 0 \), and \( \beta \in (0,1) \), a server-side algorithm is called an \((\alpha, \beta)\)-accurate protocol for longitudinal data collection if it outputs at each time \( t \in [d] \), an estimate \( \hat{a}[t] \) of \( a[t] \), such that

\[ \Pr[\max_{t \in [d]} |\hat{a}[t] - a[t]| > \alpha] \leq \beta. \]

**Definition 2.2 (Differential Privacy [7]).** Let \( A : D \rightarrow Y \) be a randomized algorithm. Algorithm \( A \) is called \( \epsilon \)-differentially private if for all \( v, v' \in D \) and all (measurable) \( Y \subseteq Y \),

\[ \Pr[A(v) \in Y] \leq e^\epsilon \cdot \Pr[A(v') \in Y]. \]  

The parameter \( \epsilon \) characterizes the similarity of the output distributions of \( A(v) \) and \( A(v') \), and \( \epsilon \) is called the privacy budget.

A data-collecting protocol is called \( \epsilon \)-locally differentially private (LDP) if each user reports only a version of their data perturbed locally by an \( \epsilon \)-differentially private algorithm.

![Diagram](https://via.placeholder.com/150)

**Figure 1:** An example with \( d = 4 \) and \( k = 2 \). The left-hand side enumerates all dyadic intervals defined on \([d]\), with the nodes highlighted in purple representing the dyadic decomposition \( C(3) \) of the interval \([3]\). The right-hand side enumerates all partial sums for the discrete derivative \( X_u = (0,1,0,-1) \), with the nodes highlighted in purple representing the partial sums associated with dyadic intervals in \( C(3) \).

© Olga Ohrimenko, Anthony Wirth, Hao Wu

**Problem 2.3 (Private Longitudinal Data Collection).** Given privacy parameter \( \epsilon \), and failure probability \( \beta \), design \( \epsilon \)-locally differentially private \((\alpha, \beta)\)-accurate protocol for longitudinal data collection, that minimizes \( \alpha \).

Our goal is to achieve an \( \alpha \in O((1/\epsilon) \cdot (\log d) \cdot \sqrt{k \cdot n \cdot \log(1/\beta)}) \).

## 3 PRELIMINARIES

We use the following notation. For every \( i, j \in \mathbb{N} \), such that \( i \leq j \), \( \{i, \ldots, j\} \) denotes the set of integers \( \{i, \ldots, j\} \). For each \( j \in \mathbb{N}^+ \), set \( \{j\} \) refers to \([1, \ldots, j]\).

As the data of each user \( u \) changes at most \( k \) times, we introduce the following transformation of \( X_u \) into a sparse vector, which has at most \( k \) non-zero coordinates.

**Definition 3.1 (Data Derivative [19]).** For each user \( u \) and for all \( t \in [d] \), let \( X_u[t] = s_{tu}[t] - s_{tu}[t - 1] \), where for convenience, we assume that \( s_{tu}[0] = 0 \), so that \( X_u[1] \) is well defined. The discrete derivative of user data is

\[ X_u = (X_u[1], \ldots, X_u[d]) \in \{-1, 0, 1\}^d. \]

For example, if \( s_{tu} = (0,1,1,0) \), then \( X_u = (0,1,0,-1) \). Observe that for each \( t \in [d] \), \( s_{tu}[t] = \sum_{r \in [t]} X_u[r'] \). We are interested in the cumulative data changes over intervals whose length is a power of \( 2 \); such intervals are called dyadic.

**Definition 3.2 (Dyadic Intervals).** For each \( h \in [0 \ldots \log d] \), and each \( j \in [d/2^h] \), define the dyadic interval \( I_{h,j} = \{(j - 1) \cdot 2^h + 1, \ldots, j \cdot 2^h\} \), where \( h \) is called its order. Denote the collection of dyadic intervals of order \( h \) by

\[ ISet[h] = \{I_{h,j} : j \in [d/2^h]\}, \]

and the collection of all dyadic intervals by

\[ ISet = \cup_{h \in [0 \ldots \log d]} ISet[h]. \]

Interval \( I \) represents an element from \( ISet \).

**Example 3.3.** All possible dyadic intervals defined on interval \([4]\) include \( I_{0,1} = \{1\}, I_{0,2} = \{2\}, I_{0,3} = \{3\}, I_{0,4} = \{4\}, I_{1,1} = \{1, 2\}, I_{1,2} = \{3, 4\}, I_{2,1} = \{1, 2, 3, 4\} \).
\textbf{Definition 3.4 (Partial Sum).} For each $h \in \{0, \ldots, \log d\}$, and each $j \in [d/2^h]$, define for each user the partial sum associated with $I_{h,j}$ to be
\[ S_u(I_{h,j}) = \sum_{t \in I_{h,j}} X_u[t], \forall u \in [n]. \tag{3} \]
and their sum as
\[ S(I_{h,j}) = \sum_{u \in [n]} S_u(I_{h,j}) \tag{4} \]

Here, $h$ refers to the order of a partial sum.

\textbf{Example 3.5.} Suppose that $X_u = (0, 1, 0, -1)$. The possible partial sums are $S_u(I_{0,1}) = 0$, $S_u(I_{0,2}) = 1$, $S_u(I_{0,3}) = 0$, $S_u(I_{0,4}) = -1$, $S_u(I_{1,1}) = 1$, $S_u(I_{1,2}) = -1$, $S_u(I_{1,3}) = 0$.

Note that if $S_u(I_{h,j}) \neq 0$, then there is at least one $t \in I_{h,j}$ for which $X_u[t] \neq 0$. Since dyadic intervals of order $h$ are disjoint, and there are at most $k$ non-zero $X_u[t]$ over all $t \in [d]$, we have this observation:

\textbf{Observation 3.6.} For each $h \in [0 \ldots \log d]$, there can be at most $k$ indices $j \in [d/2^h]$ for which $S_u(I_{h,j}) \neq 0$.

\textbf{Combining Equation (3) and the fact that $X_u[t] = st_u[t] - st_u[t-1]$, and that $I_{h,j} = ((j - 1) \cdot 2^h + 1, \ldots, j \cdot 2^h)$, we observe:

\textbf{Observation 3.7.} For each $u \in [n]$, and $I_{h,j} \in \mathcal{I}$, we have
\[ S_u(I_{h,j}) = st_u[j \cdot 2^h] - st_u[(j-1) \cdot 2^h] \in (-1, 0, 1). \tag{5} \]

Recall that $st_u[t] = \sum_{t' \in [t]} X_u[t']$. It can be viewed as the cumulative data change from time 1 to time $t$. If we decompose the interval $[t]$ into a sequence of disjoint dyadic intervals, then $st_u[t]$ can be expressed as the sum of cumulative changes over these intervals.

\textbf{Fact 3.8. (Dyadic Decomposition)} For each $t \in [d]$, the interval $[t]$ can be decomposed into a minimum collection, $C(t)$, of at most $\lceil \log t \rceil$ disjoint dyadic intervals with distinct orders.

For example, the dyadic decomposition of the interval $[3]$ is given by $\{\{1, 2\}, \{3\}\}$. See Figure 1 for the illustration. In general, for $1 \leq t \leq r \leq d$, the interval $[t \ldots r]$ can also be decomposed into a minimum collection of at most $\lceil 2 \cdot \log(r - t + 1) \rceil$ disjoint dyadic intervals. But these intervals might not have distinct orders. For example, the decomposition of the interval $[2 \ldots 3]$ is given by $\{\{2\}, \{3\}\}$. Fact 3.8 holds since the interval $[t]$ starts from 1.

Via the definition of $C(t)$, we have $st_u[t] = \sum_{I_{h,j} \in C(t)} S_u(I_{h,j})$. Summing over $u \in [n]$, and combining this with Equation (1), (3) and (4), we have
\[ a[t] = \sum_{u \in [n]} \sum_{I_{h,j} \in C(t)} S_u(I_{h,j}) = \sum_{I_{h,j} \in C(t)} S(I_{h,j}) \tag{6} \]

\section{4 FRAMEWORK}

In this section, we introduce an LDP protocol that achieves our proposed error guarantee. It improves the previously known bound \[19\] by a factor of $\sqrt{k}$. The main result of the paper is summarized as follows:

\textbf{Theorem 4.1.} Assuming that $\varepsilon \leq 1$, and $\varepsilon^{-1} \cdot (\log d) \cdot \sqrt{k \cdot \ln(d/\beta)} \leq \sqrt{n}$, there is an $\varepsilon$-local differentially private protocol, such that, with probability at least $1 - \beta$, its $d$ estimates $\{\hat{a}[t]\}$ of $\{a[t]\}$ satisfy
\[ \max_{t \in [d]} |\hat{a}[t] - a[t]| \in O\left(\frac{\log d}{\varepsilon} \cdot \sqrt{k \cdot n \cdot \ln \frac{d}{\beta}}\right). \]

The assumption $\varepsilon^{-1} \cdot (\log d) \cdot \sqrt{k \cdot \ln(d/\beta)} \leq \sqrt{n}$ ensures that $\varepsilon^{-1} \cdot (\log d) \cdot \sqrt{k \cdot n \cdot \ln(d/\beta)} \leq n$, avoiding a trivial bound.

\subsection{4.1 Overview}

Our framework is split between the user (or client) and the server. Each user reports perturbed data to the server, who aggregates the data from all users and computes the required statistics while adjusting for the noise in the reports.

As a warm up consider the following (non-private) naive protocol. Here, each user $u \in [n]$ computes and reports (to the server) each partial sum $S_u(I_{h,j})$ immediately after it has all the data needed for the computation. In particular, since the last number in the dyadic interval $I_{h,j}$ is $j \cdot 2^h$, the last data point needed to compute $S_u(I_{h,j})$ is $X_u[j \cdot 2^h]$. Based on the reports of the partial sums, according to Equation (6), the server can obtain for each $t \in [d]$ a precise value of $a[t]$.

The naive protocol does not provide LDP guarantees. To this end, our algorithms build on a combination of the following two reporting techniques.

\textbf{Reporting based on Sampling.} Suppose, instead of reporting every partial sum, each user $u \in [n]$ samples an integer $h_u \in [0 \ldots \log d]$ uniformly at random, and reports only the partial sums with order $h_u$. The server no longer has exact values of the $a[t], \forall u \in [d]$. However, the server can easily construct an unbiased estimate. For each $h \in [0 \ldots \log d]$, and each $j \in [d/2^h]$, let $z_u[h, j]$ be the server’s estimate of $S_u(I_{h,j})$. If it sets $z_u[h, j] = (1 + \log d) \cdot I_{h_u = h} \cdot S_u(I_{h,j})$, where $I_{h_u = h}$ is the indicator for the event $h_u = h$, then $z_u[h, j]$ is an unbiased estimate of $S_u(I_{h,j})$. Via linearity of expectation, replacing $S_u(I_{h,j})$ in Equation (6) with $z_u[h, j]$ gives an unbiased estimator $a[t]$.\]

\textbf{Reporting with Perturbation.} In the previous paragraph, conditioned on $h_u = h$, user $u \in [n]$ reports a sequence of $L = d/2^h$ partial sums with order $h$ to the server, each taking value in $(-1, 0, 1)$ according to Equation (5). To achieve LDP with this approach, the user should invoke some randomizer $M$, to perturb their partial sums before reporting. This also requires a change in computing the estimators $z_u[h, j]$ with respect to $M$. Our randomizer, $M$, exploits sparsity: there are at most $k$ non-zero partial sums with order $h$ (Observation 3.6).

\subsection{4.2 Client Side}

\textbf{Randomizer.} Let $M$ be a client-side randomizer. Our protocol relies on a randomizer $M$ that provides the following functionalities and has three properties described below.

$M$ has an (optional) initialization phase $M.init(L, k, \varepsilon)$, with parameters the length of the input $L$, the maximum number of non-zero elements in the input $k$, and the privacy budget $\varepsilon$. In this
phase, $M$ can perform some pre-computation whose result is kept for reference later.

During the protocol, $M$ takes as input a sequence $v_1, \ldots, v_L$ where each value is in $\{-1, 0, 1\}$, corresponding to user's data, and outputs a sequence $M^{(1)}(v_1), \ldots, M^{(L)}(v_L) \in \{-1, 1\}$. The output of $M^{(j)}(v_j)$ may depend not only on the input $v_j$ and the randomness of $M^{(j)}$, but also on the pre-computation result, the past inputs $v_1, \ldots, v_{j-1}$ and outputs $M^{(1)}(v_1), \ldots, M^{(j-1)}(v_{j-1})$.

Finally, $M$ should satisfy the following three properties.

**Property I.** Given $L$, $k$ and $\epsilon \leq 1$, there exists $p_{\text{min}}, p_{\text{max}} \in (0, 1)$, such that $p_{\text{max}}/p_{\text{min}} \leq e^\epsilon$, and that for each $k$-sparse input sequence $v_1, \ldots, v_L \in \{-1, 0, 1\}$, and each sequence $w_1, \ldots, w_L \in \{-1, 1\}$, 

$$
\Pr[M^{(1)}(v_1) = w_1, \ldots, M^{(L)}(v_L) = w_L] \in [p_{\text{min}}, p_{\text{max}}].
$$

**Property II.** There exists $c_{\text{gap}} \in (0, 1)$, such that for all $j \in [L]$, if $v_j \neq 0$, 

$$
\Pr[M^{(j)}(v_j) = v_j] = \Pr[M^{(j)}(v_j) = -v_j] = c_{\text{gap}}.
$$

**Property III.** For all $j \in [L]$, if $v_j = 0$, 

$$
\Pr[M^{(j)}(v_j) = 1] = \Pr[M^{(j)}(v_j) = -1] = \frac{1}{2}. \tag{9}
$$

**Example 4.2.** The following randomizer [17] satisfies these three properties. Suppose that $M$ perturbs each coordinate independently such that: if $v_j \neq 0$, then $\Pr[M^{(j)}(v_j) = v_j] = e^{\epsilon k}/(e^{\epsilon k} + 1)$ and $\Pr[M^{(j)}(v_j) = -v_j] = 1/(e^{\epsilon k} + 1)$; if $v_j = 0$, then $M^{(j)}(v_j)$ outputs $-1$ or $1$, uniformly at random. Now, Property II and Property III are clearly satisfied with $c_{\text{gap}} = (e^{\epsilon k} - 1)/(e^{\epsilon k} + 1)$. Finally, it can be verified that Property I is satisfied, with $p_{\text{min}} = 2^{-(L-k)} \cdot (e^{\epsilon k} + 1)^{-k}$ and $p_{\text{max}} = e^\epsilon \cdot 2^{-(L-k)} \cdot (e^{\epsilon k} + 1)^{-k}$.

This simple randomizer does not perform pre-computation in the initialization phase, and for each $j \in [L]$, the result of $M^{(j)}(v_j)$ does not depend on historical inputs or outputs.

Intuitively, Property I states that, regardless of the input sequence, each sequence in $\{-1, 1\}^k$ is output by $M$ with similar probability (up to a factor of $e^\epsilon$), by which the client-side algorithm is differentially private. Property II ensures that $M$ preserves each non-zero coordinate of the input sequences with a common probability; indeed, $c_{\text{gap}}$ is the common difference between the coordinate preservation and reversal probabilities. Property III requires that $M$ outputs $1$ and $-1$ with equal probability for each zero coordinate. Based on Property II and Property III:

**Observation 4.3.** For $v_j \in \{-1, 0, 1\}$, it always holds that 

$$
\mathbb{E}[(c_{\text{gap}}^{-1}) \cdot M^{(j)}(v_j)] = v_j. \tag{10}
$$

Since $M^{(j)}(v_j)$ is in $\{-1, 1\}$; multiplying by $(c_{\text{gap}}^{-1})$ amplifies the range to $[-(c_{\text{gap}}^{-1}) \cdot (c_{\text{gap}}^{-1})]$. As will be explained in Section 4.3, this range plays a vital role in utility analysis. The smaller this range, the better utility we can obtain. Since $\epsilon \leq 1$, the randomizer in Example 4.2 guarantees $(c_{\text{gap}}^{-1}) = (e^{\epsilon k} + 1)/(e^{\epsilon k} + 1)$ in $O(k/\epsilon)$, or equivalently, $c_{\text{gap}} \in \Omega(\sqrt{k})$. Our protocol relies on a randomizer, FutureRand, with a $\sqrt{k}$-better guarantee than this naive one.

**Algorithm 1 Client $\mathcal{A}_{\text{clt}}$**

- **Input:** User Data $X_u$, Privacy parameter $\epsilon$.
  1. report $h_u \leftarrow [0 \ldots \log d]$. \textcolor{gray}{-> Send $h_u$ to the server}
  2. Set $L \leftarrow d/2^{h_u}$ and $\omega_u \leftarrow \{0\}^L$.
  3. $M.\text{init}(L, k, \epsilon)$
  4. for each time $t \in [d]$ do \textcolor{gray}{User data $X_u[t]$ arrives}
  5. if $2^{h_u}$ divides $t$ then
  6. $j \leftarrow t/2^{h_u}$
  7. Compute the $j^{(th)}$ partial sum $S_u(I_{h_u,j})$.
  8. report $\omega_u[j] \leftarrow M^{(j)}(S_u(I_{h_u,j}))$

**Theorem 4.4 (FutureRand).** There is a randomizer $M$, called FutureRand, that satisfies the above conditions with

$$
\epsilon_{\text{gap}} \in \Omega(\epsilon/\sqrt{k}). \tag{11}
$$

The design of FutureRand is non-trivial, and we defer it to Section 5. For now, we apply it to complete the design of our longitudinal data tracking protocol.

**Client-Side Algorithm.** The client-side algorithm is described in Algorithm 1. Each user $u$ first samples an integer $h_u \in [0 \ldots \log d]$ uniformly at random and reports it to the server. Then $u$ initializes a zero vector $\omega_u$, of size $L = d/2^{h_u} = \lceil |\text{Set}(h_u)| \rceil$ for the dyadic intervals with order $h_u$. They also initialize a randomizer $M$, with parameters $L, k$ and $\epsilon$.

The user reports to the server at time $t \in [d]$ if and only if $2^{h_u}$ divides $t$. In particular, the user $u$ computes partial sum $S_u(I_{h_u,j})$ associated with the $j = t/2^{h_u}$-th dyadic interval $I_{h_u,j}$, with order $h_u$. Note that each multiple of $2^{h_u}$ is the first time that $u$ has all data $X_u[t']$ needed to compute $S_u(I_{h_u,j})$, since by definition:

$$
I_{h_u,j} = \{ (j-1) \cdot 2^{h_u} + 1, \ldots, j \cdot 2^{h_u} \} = \{ t \cdot 2^{h_u} + 1, \ldots, t \}.
$$

and $S_u(I_{h_u,j}) = \sum_{t' \in I_{h_u,j}} X_u[t']$. This partial sum is then perturbed by the randomizer $M^{(j)}$. The perturbed value is stored in $\omega_u[j]$ and reported to the server.

**Privacy Guarantee.** The privacy guarantee of Algorithm 1 is an immediate consequence of Property I.

**Theorem 4.5.** $\mathcal{A}_{\text{clt}}$ is $\epsilon$-differentially private.

**Proof of Theorem 4.5.** Consider a user $u \in [n]$ with data $X_u$. For each $h \in [0 \ldots \log d]$, we have $\Pr[h_u = h] = 1/(1 + \log d)$, and, conditioned on $h_u = h$, we have $L = d/2^h$. Via Inequality (7), there exist $p_{\text{min}}, p_{\text{max}} \in (0, 1)$, s.t., $p_{\text{max}} \leq e^\epsilon \cdot p_{\text{min}}$ and that for each $w \in \{-1, 1\}^L$, it holds that 

$$
\Pr[\omega_u[1] = w_1, \ldots, \omega_u[L] = w_L | h_u = h] \in [p_{\text{min}}, p_{\text{max}}].
$$

Suppose that data $X_u$ of user $u$ is instead $X_u'$, with other inputs unchanged. Running $\mathcal{A}_{\text{clt}}$ based on $X_u'$, we have $h_u'$ and $\omega_u'$ with the same probability as above. Thus 

$$
\Pr[h_u' = h, \omega_u'[1] = w_1, \ldots, \omega_u'[L] = w_L] \leq e^\epsilon.
$$

Since $p_{\text{max}} \leq e^\epsilon \cdot p_{\text{min}}$, the ratio of outcome probabilities is 

$$
\frac{\Pr[h_u = h, \omega_u[1] = w_1, \ldots, \omega_u[L] = w_L]}{\Pr[h_u' = h, \omega_u'[1] = w_1, \ldots, \omega_u'[L] = w_L]} \leq e^\epsilon.
$$
Algorithm 2 Server $\mathcal{A}_{\text{srv}}$

**Input:** Reports $(h_u, \omega_u)$ from the users.

1. $\mathcal{U}_h \leftarrow \{u \in [n] : h_u = h\}, \forall h \in [0 \ldots \log d]$.
2. for each time $t \in [d]$ do
3. for $h \in [0 \ldots \log d]$ s.t. $2^h$ divides $t$ do
4. $j \leftarrow t/2^h$
5. $\hat{S}(I_j) \leftarrow \sum_{u \in \mathcal{U}_h} (1 + \log d) \cdot (c_{\text{gap}})^{-1} \cdot \omega_u[j]$
6. output $\hat{a}[t] \leftarrow \sum_{I_j \in \mathcal{C}(t)} \hat{S}(I_j)$

Since the output space of $\mathcal{A}_{\text{clt}}$ is discrete, it satisfies Inequality (2), and therefore is differentially private (Definition 2.2).

4.3 Server Side

**Server-Side Algorithm.** The server-side algorithm is described in Algorithm 2. At the beginning of the algorithm, according to $\mathcal{A}_{\text{clt}}$ (Algorithm 1, line 1), we assume that the server has received the $n$ sampled orders $h_1, \ldots, h_n$ one from each user. The server partitions the users into $1 + \log d$ subsets $\mathcal{U}_{h_1}, \mathcal{U}_{h_2}, \ldots$, such that $\mathcal{U}_h = \{u \in [n] : h_u = h\}$ consists of the subset of users whose sampled orders equal $h$. Observe that each user $u \in \mathcal{U}_h$ reports their perturbed partial sums $\omega_u[1], \omega_u[2], \ldots, \omega_u[d/2^h]$, at times $2^h, 2 \cdot 2^h, \ldots, d$, respectively.

Based on user reports, for each $u \in [n], h \in [0 \ldots \log d], j \in [d/2^h]$, consider this estimator $z_u[h,j]$ of $S_u(I_j)$. If $h_u = h$, then $z_u[h,j] = (1 + \log d) \cdot (c_{\text{gap}})^{-1} \cdot \omega_u[j]$, otherwise, $z_u[h,j] = 0$. Since $\omega_u[j] \in [-1,1]$, it holds that $z_u[h,j] \in (1 + \log d) \cdot (c_{\text{gap}})^{-1} \cdot [-1,1]$. Further, the expectation of $z_u[h,j]$ is given by

$$\mathbb{E}[z_u[h,j]] = \Pr[h_u = h] \cdot (1 + \log d) \cdot \mathbb{E}[(c_{\text{gap}})^{-1} \cdot \omega_u[j]].$$

Noting that $\omega_u[j] = M^{(j)}(S_u(I_j))$, and via Equation (10), we see

$$\mathbb{E}[z_u[h,j]] = S_u(I_j).$$

Via linearity of expectation, $\hat{S}(I_j) = \sum_{u \in [n]} z_u[h,j]$ is an unbiased estimator of $S(I_j)$. Keeping only the non-zero terms, and replacing $z_u[h,j]$ with its definition, we have

$$\hat{S}(I_j) = \sum_{u \in \mathcal{U}_h} (1 + \log d) \cdot (c_{\text{gap}})^{-1} \cdot \omega_u[j],$$

which justifies the update rule of $\hat{S}(I_j)$ (Algorithm 2, line 5). Finally, replacing $S(I_j)$ with $\hat{S}(I_j)$ in Equation (6) gives an unbiased estimator, $\hat{a}[t] = \sum_{I_j \in \mathcal{C}(t)} \hat{S}(I_j)$, of $a[t]$ (Algorithm 2, line 6).

**Utility Guarantee.** We conclude that $\hat{a}[t]$ is an unbiased estimator of $a[t]$; how often is it accurate?

**Lemma 4.6.** For each $t \in [d]$, assuming that $\varepsilon \leq 1$ and that $\sqrt{n} \geq (c_{\text{gap}})^{-1} \cdot \log d \cdot \sqrt{n \cdot \ln \frac{1}{\beta'}}$, with probability at least $1 - \beta'$, the estimates $\hat{a}[t]$ computed as in Algorithm 2 satisfy

$$|\hat{a}[t] - a[t]| \in O\left((c_{\text{gap}})^{-1} \cdot \log d \cdot \sqrt{n \cdot \ln \frac{1}{\beta'}}\right).$$

We first prove our main Theorem 4.1 based on Theorems 4.4 and Lemma 4.6, then justify Lemma 4.6.

**Proof of Theorem 4.1.** Via Theorem 4.4, we know that $c_{\text{gap}} \in \Omega(\varepsilon/\sqrt{k})$. Therefore, $(c_{\text{gap}})^{-1} \in O(\sqrt{k}/\varepsilon)$. Applying Lemma 4.6 with $\beta' = \beta/d$, and via union bound, we conclude that, with probability at least $1 - \beta$,

$$\max_{t \in [d]} |\hat{a}[t] - a[t]| \in O\left(\frac{\log d}{\varepsilon} \cdot \sqrt{k \cdot n \cdot \ln \frac{d}{\beta'}}\right).$$

**Proof of Lemma 4.6.** First, rewrite

$$\hat{a}[t] = \sum_{I_j \in \mathcal{C}(t)} \hat{S}(I_j) = \sum_{I_j \in \mathcal{C}(t)} \sum_{u \in [n]} z_u[h,j].$$

For each $u \in [n]$, define $Y_u = \sum_{I_j \in \mathcal{C}(t)} z_u[h,j]$. Exchanging the order of summation gives $\hat{a}[t] = \sum_{u \in [n]} Y_u$. Clearly the $Y_u$ are independent. We claim (proven below): $Y_u \in (1 + \log d) \cdot (c_{\text{gap}})^{-1} \cdot [-1,1]$. Then applying Hoeffding’s Inequality (Corollary A.2), and the fact that $\mathbb{E}[|\hat{a}[t] - a[t]|] = a[t]$, we see that with probability at most $\beta'$,

$$|\hat{a}[t] - a[t]| \geq (1 + \log d) \cdot (c_{\text{gap}})^{-1} \cdot \sqrt{2n \cdot \ln \frac{2}{\beta'}}.$$  (13)

We prove the claim, that $Y_u \in (1 + \log d) \cdot (c_{\text{gap}})^{-1} \cdot [-1,1]$. By Fact 3.8, dyadic intervals in $C(t)$ have distinct orders. By definition, $z_u[h,j]$ is non-zero exactly when $h_u = h$, so among all $I_j \in \mathcal{C}(t)$, there is at most one non-zero $z_u[h,j]$. Its value either $-(1 + \log d) \cdot (c_{\text{gap}})^{-1}$ or $(1 + \log d) \cdot (c_{\text{gap}})^{-1}$. □

5 RANDOMIZER

In this section, we present a randomized, future $\text{Rand}$, denoted briefly by $M$, that satisfies Theorem 4.4. The $\text{FutureRand}$ is based on two techniques, composition for randomized responses (for non-zero coordinates) and pre-computation for the composition.

5.1 Overview

The input to $M$ is a $k$-sparse sequence $v = (v_1, \ldots, v_L) \in \{-1,0,1\}^L$, and the output sequence $M^{(v)}(v_1, \ldots, M^{(v)}(v_L) \in \{-1,1\}^L$. Denote the support of $v$ as $\text{supp}(v) = \{j \in [L] : v_j \neq 0\}$.

**Zero Coordinates.** For each $j \notin \text{supp}(v)$, $M^{(v)}(v)$ outputs $-1$ and $1$ uniformly at random. Hence **Property III** is trivially satisfied.

**Non-Zero Coordinates.** The presentation of $\text{FutureRand}$ to handle non-zero coordinates follows three steps.

- **Offline Input with Fixed Support Size.** First we assume that all coordinates of $v$ are inputted to $M$ simultaneously, and that the $v$ contains exactly $k$ non-zero coordinates.

- **Online Input with Fixed Support Size.** We convert the protocol to online, where each coordinate $v_j$ arrives one by one, and $M$ outputs the perturbed value of $v_j$ immediately.

- **Online Input with Bounded Support Size.** We show that **Online Input with Fixed Support Size** protocol provides the same guarantees even when $v$ has support less than $k$.

For each step, we need to show that the $M$ constructed satisfies **Property I, Property II** with $c_{\text{gap}} \in \Omega(\varepsilon/\sqrt{k})$. 
5.2 Offline Input with Fixed Support Size

By assumption, \( v \) contains \( k \) non-zero coordinates. \( M \) invokes a subroutine, \( \tilde{\mathcal{R}} : \{-1, 1\}^k \rightarrow \{-1, 1\}^k \) to perturb these coordinates. \( \tilde{\mathcal{R}} \) is a composed randomizer; instead of perturbing each coordinate independently, it adds correlated noise to the coordinates. The building block of \( \tilde{\mathcal{R}} \) is a basic randomizer \( \mathcal{R} \). The pseudo-codes of both \( \mathcal{R} \) and \( \tilde{\mathcal{R}} \) are described in Algorithm 3.

**Basic Randomizer** \( \mathcal{R} \) [17]. For each \( \zeta \in \{-1, 1\} \),

\[
\mathcal{R}(\zeta) = \begin{cases} 
\zeta, & \text{w.p. } e^\frac{\varepsilon}{2} / (e^\frac{\varepsilon}{2} + 1) \\
-\zeta, & \text{w.p. } 1 / (e^\frac{\varepsilon}{2} + 1)
\end{cases},
\]

where \( \varepsilon \) depends on \( \varepsilon \), defined later.

**Composed Randomizer** \( \tilde{\mathcal{R}} \). For an input \( b \in \{-1, 1\}^k \), if it first applies \( \mathcal{R} \) independently to each coordinate of \( b \). Denote the result by \( b' = \mathcal{R}(b) \). Then \( \tilde{\mathcal{R}}(b) \) is shared by other procedures, to keep track of the pseudo-codes of both \( \mathcal{R} \) and \( \tilde{\mathcal{R}} \). The pseudo-codes of both \( \mathcal{R} \) and \( \tilde{\mathcal{R}} \) are described in Algorithm 3.

**Lemma 5.2** (Annulus). Given \( b \in \{-1, 1\}^k \), denote

\[\text{Ann}(b) = \left\{ s \in \{-1, 1\}^k : \| s \|_0 \in [\text{LB}, \text{UB}] \right\}.\]

If \( b' \notin \text{Ann}(b) \), \( \tilde{\mathcal{R}} \) replaces it with a uniform sample from \( \{-1, 1\}^k \setminus \text{Ann}(b) \). Finally, \( \tilde{\mathcal{R}} \) outputs \( b' \).

Randomizer \( \tilde{\mathcal{R}} \) has the following privacy and utility guarantees, the analysis of which is deferred to Section 5.5.

**Lemma 5.3**. For a given input \( b \), denote \( \tilde{b} = \tilde{\mathcal{R}}(b) \) the output of \( \tilde{\mathcal{R}} \). There exist \( p'_{\text{min}}, p'_{\text{max}} \in (0, 1) \) with \( p'_{\text{max}} \leq e^\varepsilon \cdot p'_{\text{min}} \), such that for each input sequence \( b \in \{-1, 1\}^k \), and each sequence \( s \in \{-1, 1\}^k \),

\[
\Pr[\tilde{\mathcal{R}}(b) = s] = [p'_{\text{min}}, p'_{\text{max}}].
\]

for \( \tilde{\mathcal{R}}(b) \) as defined in Algorithm 3.

**Algorithm 3** FutureRand \( M \), describing \( \tilde{\mathcal{R}} \)

1. **Procedure** Basic Randomizer \( \mathcal{R}(\zeta) \)
   - Input: Value \( \zeta \in \{-1, 1\} \).
   - return \( -\zeta \) w.p. \( 1 / (e^\frac{\varepsilon}{2} + 1) \) and \( \zeta \) w.p. \( e^\frac{\varepsilon}{2} / (e^\frac{\varepsilon}{2} + 1) \).
2. **Procedure** Composed Randomizer \( \tilde{\mathcal{R}}(b) \)
   - Input: Vector \( b \in \{-1, 1\}^k \).
   - Sample \( b' \leftarrow \mathcal{R}(b_1), \ldots, \mathcal{R}(b_k) \).
   - if \( b' \notin \text{Ann}(b) \) then
     - return \( b' \).
     - **Procedure** M.init\( (L, k, \varepsilon) \)
   - Input: Input length \( L \); Support size \( k \); Privacy parameter \( \varepsilon \).
   - Set \( \tilde{\varepsilon} \leftarrow \varepsilon / (5\sqrt{k}) \).
   - Set \( b \leftarrow \mathcal{R}(1^k) \).
   - Set \( \text{nnz} \leftarrow 0 \).
   - **Procedure** \( M^{(j)}(w_j) \) \( j = 1, 2, \ldots, L \)
   - Input: Value \( w_j \in [-1, 0, 1] \).
   - if \( w_j \neq 0 \) then
     - \( \text{nnz} \leftarrow \text{nnz} + 1 \).
     - return \( w_j \cdot \tilde{b}_{\text{nnz}} \).
   - else
     - return \( -1 \) or \( +1 \) uniformly at random.

and output \( s = (w_{j_1}, \ldots, w_{j_k}) \), it holds that \( \Pr[E_2] = \Pr[\tilde{\mathcal{R}}(b) = s] \in [p'_{\text{min}}, p'_{\text{max}}] \). Since \( E_1 \) and \( E_2 \) are independent, we have

\[
\Pr[E] = \Pr[E_1] \cdot \Pr[E_2] \in 2^{-(L-k)} \cdot [p'_{\text{min}}, p'_{\text{max}}],
\]

which proves Property I.

5.3 Online Input with Fixed Support Size

In this step, we modify \( M \) into an online algorithm, where each coordinate \( v_j \) arrives one by one, and \( M \) is required to perturb and output each coordinate immediately after its arrival. Since \( M \) perturbs the zero coordinates independently, we only need to take care of the non-zero coordinates. We develop a new pre-computation technique to generate the correlated noises for the non-zero elements in the initialization phase. With these noises, we perturb the non-zero coordinates as they arrive. This motivates the design of \( \text{M.init}(L, k, \varepsilon) \) and \( M^{(j)}(v_j) \), \( j \in [L] \), whose pseudo-codes are in Algorithm 3. \( \text{M.init}(L, k, \varepsilon) \). The procedure takes as parameters \( L \), the size of the input sequence; \( k \), the maximum number of non-zero elements in the input sequence; and \( \varepsilon \), the privacy parameter. It sets \( \tilde{\varepsilon} \) as \( \varepsilon / (5\sqrt{k}) \), and invokes the composed randomizer \( \tilde{\mathcal{R}} \) with the vector \( 1^k \) consisting of all ones. The returned sequence is kept as a vector \( \tilde{b} \). Finally, the procedure creates a variable \( \text{nnz} \) with value 0; the variable \( \text{nnz} \) is shared by other procedures, to keep track of the support of the input sequence received so far.

\( M^{(j)}(v_j) \). For all \( j \in [L] \), \( M^{(j)}(v_j) \) shares the same pseudo-codes. If \( v_j = 0 \), the procedure outputs \(-1 \) or \(+1 \), uniformly at random. Otherwise, it increments \( \text{nnz} \) by 1, indicating that \( v_j \) is the \((\text{nnz})^{(th)}\)
non-zero entry processed by the procedure. Then it outputs the value of $s_j$ multiplied by $b_{min}$.

**Analysis.** We need to prove that Property I and Property II hold. Consider the events $E_1$ and $E_2$ defined as before. We see that $\Pr[E_1]$ remains the same. Denote the indices in $\text{supp}(v)$ as $j_1 < \cdots < j_k$. Based on our modification, for each $i \in [k]$, we have $M(i)(v_{j_i}) = b_{i} \cdot v_{j_i}$. Hence, event $E_2$ holds only if for each $i \in [k]$, $b_{i} \cdot v_{j_i} = w_{j_i}$, equivalently, $b_{i} = w_{j_i} / v_{j_i}$. Since $\tilde{b} = \mathcal{R}(1^k)$, applying Lemma 5.2 with $s = (w_{j_1} / v_{j_1}, \ldots, w_{j_k} / v_{j_k})$, it holds that

$$\Pr[E_2] = \Pr[\tilde{b} = s] = \Pr[\tilde{\mathcal{R}}(1^k) = s] \in [p_{min}', p_{max}'].$$

It follows that Inequality (18) still holds, and therefore Property I.

Finally, observe that $M(i)(v_{j_i}) = w_{j_i}$ if $b_{i} = 1$; and $M(i)(v_{j_i}) = -w_{j_i}$ if $b_{i} = -1$. Since $\tilde{b} = \mathcal{R}(1^k)$, we have $\Pr[b_{i} = 1] - \Pr[b_{i} = -1] = \epsilon_{gap}$. Therefore, Property II holds with $\epsilon_{gap} \in \Omega(\sqrt{\kappa})$.

### 5.4 Online Input with Bounded Support Size

In this step, we relax the constraint that $\nu$ contains exactly $k$ non-zero coordinates. In particular, we show that Property I and Property II still hold, if the same protocol discussed in Section 5.3 acts on input with support less than $k$. In the case $|\text{supp}(\nu)| = k$, each bit of the pre-generated vector $\tilde{b} = \mathcal{R}(1^k)$ is used to multiply some non-zero coordinate of $b$. In the case $|\text{supp}(\nu)| < k$, however, only the first $|\text{supp}(\nu)|$ bits of $b$ are used.

**Analysis.** With a similar argument as previous section. Property II holds with $\epsilon_{gap} \in \Omega(\sqrt{\kappa})$. To verify Property I, we prove that Inequality (18) still holds. Consider the events $E_1$ and $E_2$ defined as before. We have $\Pr[E_1] = 2^{-L - |\text{supp}(\nu)|}$ as the $M(i)(v_{j_i})$ variables are independent in $(-1, 1)$ random variables for $j \notin \text{supp}(\nu)$. The event $E_2$ happens if only for each $i \in |\text{supp}(\nu)|$, it holds that $b_{i} = w_{j_i} / v_{j_i}$. Let $\mathcal{G}$ be the subset of sequence $s \in \{-1, 1\}^k$ which satisfies $s_{j_i} = w_{j_i} / v_{j_i}$, for each $i \in |\text{supp}(\nu)|$. There are $2^{k - |\text{supp}(\nu)|}$ such possible sequences, each being outputted by $\mathcal{R}(1^k)$ with probability between $[p_{min}'$, $p_{max}']$ (by Lemma 5.2). Therefore,

$$\Pr[E_2] = \Pr[\tilde{b} \in \mathcal{G}] \in 2^{k - |\text{supp}(\nu)|} \cdot [p_{min}', p_{max}'].$$

Multiplying it by $\Pr[E_1] = 2^{-L - |\text{supp}(\nu)|}$ proves Inequality (18).

### 5.5 Sketch Proofs

We conclude the technical presentation with outlines of our proofs for Lemma 5.2 and 5.3. The complete proofs for Lemma 5.2 and 5.3 are included in Appendix A.1 and A.1.2, respectively.

We use the following notation. As before, denote $p = 1/(e^\epsilon + 1)$. Note that $1 - p = e^\epsilon p$. For each $i \in [0..k]$, define

$$g(i) = p((1 - p)^{k - i} = p^k \cdot e^{\epsilon(k - i)}.$$

We see that $g$ is a decreasing function with respect to $i$. For each $b \in \{-1, 1\}^k$, denote the result of applying $\mathcal{R}$ independently to each of its coordinates as $\mathcal{R}(b) = (\mathcal{R}(b_1), \ldots, \mathcal{R}(b_k))$. For each $s \in \{-1, 1\}^k$, it is easy to verify that $\Pr[\mathcal{R}(b) = s] = g(||b - s||_0)$, where $||b||_0$ is the $l_0$ norm, and therefore $||b - s||_0$ is the number of coordinates in which $s$ differs from $b$. Since in expectation,

$$||\mathcal{R}(b) - b||_0 = kp,$$

we define

$$p_{avg} = g(kp) = e^{kp(1 - p)k - kp} = p^k \cdot e^{\epsilon(k - kp)}.$$

**Proof Outline for Lemma 5.2.** Let $b \in \{-1, 1\}^k$ be an input to $\tilde{\mathcal{R}}$. Recall that $\tilde{\mathcal{R}}$ first computes $b' = \mathcal{R}(b) = (\mathcal{R}(b_1), \ldots, \mathcal{R}(b_k))$. If $b' \in \text{Ann}(b)$, $\tilde{\mathcal{R}}$ outputs it directly. Otherwise, $\tilde{\mathcal{R}}$ outputs a uniform sample from $\{-1, 1\}^k \setminus \text{Ann}(b)$. At a high level, $\text{Ann}(b)$ consists of the $s \in \{-1, 1\}^k$ for which $\Pr[\mathcal{R}(b) = s]$ is close to $p_{avg}$, and $\tilde{\mathcal{R}}$ keeps their output probabilities. On the other hand, $\{-1, 1\}^k \setminus \text{Ann}(b)$ consists of the $s$ for which $\Pr[\mathcal{R}(b) = s]$ is much higher or lower than $p_{avg}$, and $\tilde{\mathcal{R}}$ averages their output probabilities by uniform sampling. We will show that in both cases, $\Pr[\mathcal{R}(b) = s] \approx p_{avg}$. Formally,

$$\Pr[\mathcal{R}(b) = s] \in \left[1/2^k, e^{2\sqrt{k}} \cdot p_{avg}\right], \quad \forall s \in \text{Ann}(b),$$

$$\Pr[\mathcal{R}(b) = s] \in \left[e^{-2\sqrt{k}} \cdot p_{avg}, 1/2^k\right], \quad \forall s \notin \text{Ann}(b).$$

Let $p_{min}' = e^{-2\sqrt{k}} \cdot p_{avg}$ and $p_{max}' = e^{2\sqrt{k}} \cdot p_{avg}$. Since $\tilde{e} = e^{\epsilon(2\sqrt{k})}$, $\tilde{e} = e^\epsilon \cdot \tilde{p}_{min}$. Combining Inequality (19) and Inequality (20), we know that for all $s \in \{-1, 1\}^k$, $\Pr[\mathcal{R}(b) = s] \in [p_{min}', p_{max}']$, which proves Lemma 5.2. We now prove these two inequalities separately.

**Bounding $\Pr[\mathcal{R}(b) = s]$ for $s \in \text{Ann}(b)$**. The design of $\mathcal{R}$ ensures that for each $s \in \text{Ann}(b)$, we have $\Pr[\mathcal{R}(b) = s] = \Pr[\mathcal{R}(b) = s] = g(||b - s||_0)$. Our choices of UB and LB in Equation (15) guarantee that $||b - s||_0$ is close to $kp$ if $s \in \text{Ann}(b)$, and therefore $\Pr[\mathcal{R}(b) = s]$ is close to $p_{avg}$. Next, we justify these choices.

**Choice of UB.** We set $UB = \frac{k}{2} - \ln \frac{2e^{2\sqrt{k}}}{2e^{2\sqrt{k}}}$. Since $UB$ is close to $kp$; indeed, we have $UB \in [kp, k/2]$. As $g$ is a decreasing function, this can be proven by showing that

$$g(kp) \geq 2^{-k} = g(UB) \geq g(k/2).$$

Since $||b - s||_0 \leq UB$ for each $s \in \text{Ann}(b)$, we have

$$\Pr[\mathcal{R}(b) = s] \geq g(UB) \geq 2^{-k}.$$  

As will be discussed, this property also plays an important role in upper bounding $\Pr[\mathcal{R}(b) = s]$ for $s \notin \text{Ann}(b)$.

**Choice of LB.** We pick $LB = kp - 2\sqrt{k}$ so that it is not much smaller than $kp$, and that $g(LB) = e^{2\sqrt{k}} \cdot p_{avg}$. Since $||b - s||_0 \geq LB$ for each $s \in \text{Ann}(b)$,

$$\Pr[\mathcal{R}(b) = s] = g(||b - s||_0) \leq g(LB) = e^{2\sqrt{k}} \cdot p_{avg}.$$

**Bounding $\Pr[\mathcal{R}(b) = s]$ for $s \notin \text{Ann}(b)$**. We discuss first the upper bound for this case, which is the easy part.

**Upper Bound.** As discussed, our choice of UB guarantees that each $s \in \text{Ann}(b)$ is assigned with output probability at least $2^{-k}$. Since there are $2^k$ elements in the output space $\{-1, 1\}^k$ of $\mathcal{R}(b)$, and since $\Pr[\mathcal{R}(b) = s]$ equals a common probability for each $s \notin \text{Ann}(b)$, we have

$$\Pr[\mathcal{R}(b) = s] \leq 2^{-k}, \quad \forall s \notin \text{Ann}(b).$$

**Lower Bound.** Let $UB \in [0..k] \setminus \text{LB}$, and $\mathcal{R}(b) = (\mathcal{R}(b_1), \ldots, \mathcal{R}(b_k))$. First, observe that

$$\Pr[\mathcal{R}(b) \in \text{Ann}(b)] = \sum_{i \in UB \setminus \text{LB}} i g(i).$$
As $\hat{R}$ assigns equal probability to each $s \notin \text{Ann}(b)$, it holds that
\[
P^*_\text{out} = \Pr[\hat{R}(b) = s] = \frac{\sum_{i \in \{\text{LB} \ldots \text{UB}\}} (\frac{k}{i})g(i)}{\sum_{i \in \{\text{LB} \ldots \text{UB}\}} (\frac{k}{i})}.
\] (24)

To lower bound $P^*_\text{out}$, we will partition $[\text{LB} \ldots \text{UB}]$ into three subsets, which consist of the $i$ for which $g(i)$ is significantly higher, lightly lower than, and significantly lower than $\text{pavg}$, respectively.

In particular, since $\text{LB} < k/2$ and $\text{UB} \leq k/2$, it holds that $k - \text{LB} > \text{UB}$. Hence, we can partition the set $[\text{LB} \ldots \text{UB}]$ into three subsets $[0 \ldots \text{LB} - 1], [\text{UB} + 1 \ldots k - \text{LB}]$ and $[k - \text{LB} + 1 \ldots k]$. Note that the size of the first subset equals the size of the third one. Correspondingly, we can decompose the numerator and denominator of $P^*_\text{out}$ into three parts:

\[
N_1 = \sum_{i=0}^{\text{LB}-1} (\frac{k}{i})g(i),
\]
\[
N_2 = \sum_{i=\text{LB}+1}^{k-\text{LB}} (\frac{k}{i})g(i),
\]
\[
N_3 = \sum_{i=k-\text{LB}+1}^{k} (\frac{k}{i})g(i).
\]

\[
D_1 = \sum_{i=0}^{\text{LB}-1} (\frac{k}{i}),
\]
\[
D_2 = \sum_{i=\text{LB}+1}^{k-\text{LB}} (\frac{k}{i}),
\]
\[
D_3 = \sum_{i=k-\text{LB}+1}^{k} (\frac{k}{i}).
\]

We lower bound the ratios of $N_2/D_2$ and $(N_1 + N_3)/(D_1 + D_3)$ separately.

**Lower bounding $N_2/D_2$.**

We first observe that $g(i)$ is a decreasing function, and therefore the smallest $g(i)$ in the summands of $N_2$ is lower bounded by $g(k - \text{LB}) = \text{pavg} \cdot k^{-(k-1)/2}$. Thus, $N_2/D_2 \geq \text{pavg} \cdot e^{-\varphi}$. This follows from that $g(i)$ is a decreasing function, and therefore the smallest $g(i)$ in the summands of $N_2$ is lower bounded by $g(k - \text{LB}) = \text{pavg} \cdot e^{-\varphi}$. Observe that for each $i \in [0 \ldots \text{LB} - 1]$, we have $k - i \in [k - \text{LB} + 1 \ldots k]$. The lower bound follows by pairing up each summand $(\frac{k}{i})g(i)$ in $N_1$ with $(\frac{k}{k-i})g(k-i)$ in $N_3$, and by proving that $g(i) + g(k-i) \geq \text{pavg} \cdot e^{-\varphi}$, details in the Appendix.

**Putting Together.**

Based on the lower bounds of $N_2/D_2$ and $(N_1 + N_3)/(D_1 + D_3)$, we have
\[
P^*_\text{out} = \frac{N_1 + N_2 + N_3}{D_1 + D_2 + D_3} \geq \frac{\text{pavg} \cdot e^{-\varphi}}{1 + e^{-\varphi}},
\] (25)

where the second equality follows from that for each $i \in [\text{LB} \ldots \text{UB}]$, $k - i \in [k - \text{LB} - 1 \ldots k - 1]$, and that $(\frac{k}{i}) = (\frac{k}{k-i})$. Since $\text{LB} = k - \sqrt{k} < \text{UB}$, it holds that $[\text{UB} - \sqrt{k} \ldots \text{UB} - \sqrt{k}/2] \subset [\text{LB} \ldots \text{UB}]$. Combining with that $P^*_\text{out} \leq 2^{-k}$ (Inequality (20)), we see
\[
c_\text{gap} = \frac{\text{UB} - \sqrt{k}/2}{\text{LB} - \sqrt{k}/2} \sum_{i=\text{UB}-2\sqrt{k}}^{\text{UB}} (\frac{k}{i})g(i) - 1 \geq \Omega(\frac{k}{2\sqrt{k}}).
\] (26)

It is left to prove that the right hand side is lower bounded by $\Omega(\frac{\epsilon}{\sqrt{K}})$. We will show that for each $i \in [\text{UB} - 2\sqrt{k} \ldots \text{UB} - \sqrt{k}/2]$, $g(i) - \frac{k}{2\sqrt{k}} \geq \Omega(\frac{k}{2\sqrt{k}})$.

\[
\text{UB} - \sqrt{k}/2 \sum_{i=\text{UB}-2\sqrt{k}}^{\text{UB}} (\frac{k}{i})g(i) \in \Omega(\frac{k}{2\sqrt{k}}).
\] (27)

Further,
\[
\text{UB} - \sqrt{k}/2 \sum_{i=\text{UB}-2\sqrt{k}}^{\text{UB}} (\frac{k}{i})g(i) \in \Omega(\frac{k}{2\sqrt{k}}).
\] (28)

**Combining Inequalities.**

(26), (27), and (28) proves $c_{\text{gap}} \geq \Omega(\frac{\epsilon}{\sqrt{K}})$. **Proving Inequalities.**

Observe that $g(i) \geq g(\text{UB} - \sqrt{k}/2) = \frac{k}{2\sqrt{k}}$ since $\text{UB} \leq k/2$. Since $\text{UB} - \sqrt{k}/2 \geq k - 2\sqrt{k}/k \geq k - 2(k/2 - \sqrt{k}/2) = \frac{1}{\sqrt{k}}$. Combined, we obtain
\[
\left(g(i) - \frac{k}{2\sqrt{k}}\right) \geq \left(\frac{\epsilon}{\sqrt{k} - 2\sqrt{k}/2}\right) \cdot \left(\frac{k}{2\sqrt{k}}\right) \geq \Omega(\frac{k}{2\sqrt{k}}).
\] (29)

**Proving Inequalities.**

At a high level, the claim follows from that the summation consists of $\Omega(\sqrt{K})$ terms, each of size $\Omega(\sqrt{k}/\sqrt{K})$, details in the Appendix. \qed

## 6 RELATED WORKS

**Central Model.**

Data analysis under continual observation has been studied in the central model of differential privacy [3, 6]. Here, a trusted curator, to which the clients report their true data, perturbs and releases the aggregated data. Given a stream of Boolean (0-1) values, differentially private frequency estimation algorithms were independently proposed by both Dwork et al. [6] and Chan et al. [3]: at each time period, they estimate the number of 1s appearing so far. These algorithms guarantee an error of $O(1/\epsilon \cdot \log^{1.5} t)$ (omitting failure probability) at each time $t$.

**Local Model.**

To avoid naively repeating algorithms designed for one-time computation, the memoization technique was proposed and deployed for continual collection of counter data [5, 9], where noisy answers for all elements in the domain are memoized. However, as pointed out by [5], this technique can violate differential

---

*Note that for each $i \in [0 \ldots k], \sum_{j=0}^{k}(\frac{k}{i})g(i) - \sum_{j=k+1}^{k}(\frac{k}{k-j})g(j) = 0$. For each $i \in [\text{LB} \ldots \text{UB}], i \not\in [k - \text{UB} \ldots k - \text{LB}]$, then it holds that $k - i \not\in [\text{LB} \ldots \text{UB}].$ Therefore, $\sum_{i \in [\text{LB} \ldots \text{UB}]} (\frac{k}{i})g(i) - \sum_{i=k-\text{LB}+1}^{k} (\frac{k}{i})g(i) = 0$, and

\[
c_{\text{gap}} = \sum_{i=\text{LB}}^{\text{UB}} (\frac{k}{i})g(i) - \sum_{i=k-\text{LB}+1}^{k} (\frac{k}{i})g(i).
\]
privacy. Recent works [11, 18, 19] also propose to exploit the potential sparsity of users’ data, if it changes infrequently. They differ in the frequency when the algorithm needs to give a prediction.

**Online Setting.** Our algorithmic framework is inspired by the work of Erlingsson et al. [19] for the online setting, where the server is required to report the estimate at each time step. Here we provide a succinct description of their protocol, in the notation and framework of our paper. The protocol by Erlingsson et al. requires an additional sampling step: each user $u$ samples uniformly and keeps only one non-zero coordinate of $X_u$, and sets all other non-zero coordinates to 0. Therefore, there can be at most one non-zero partial sum with sampled order $h_u$. This partial sum is perturbed by the basic randomizer $\mathcal{R}$ (Equation (14)) with $\tilde{\epsilon} = \epsilon/2$, resulting in a $(c_{gap})^{-1} \in O(1/\epsilon)$. However, due to the additional sampling step, the server side estimator of $\hat{S}(I_{h_u})$ (Algorithm 2, line 5) needs to be multiplied by an additional factor of $k$. Via similar analysis to Lemma 4.6, the error guarantee of their protocol involves a factor of $k$, instead of $\sqrt{k}$.

**Offline Setting.** The recent independent parallel work by Zhou et al. [18] considers the problem in the offline setting, where the server is required to report only after it has collected all the data. They describe a protocol that provides error guarantee of $O((1/\epsilon) \cdot \sqrt{k} \cdot (\log n/\beta) \cdot n \cdot \log(d/\beta))$. Their client-side algorithm involves hashing the coordinates of user data into a table, and reporting a perturbed version of the hash table to the server. Since the value of the entries of the hash table depends on all coordinates of the user data, it is unclear how to convert this algorithm into an online one.

**Batch Reporting.** The work by Joseph et al. [11] lies between the online and offline settings: time steps are batched into epochs, and the server is required to report at each epoch. Further, it assumes that users’ data are sampled from some (unknown) distributions, and analyzes its protocol’s performance based on this assumption. Due to this difference in the problem setting, the performance guarantees of the protocol in [11] cannot be compared directly to those provided in [18, 19] and our paper. However, the performance in [11] also degrades only linearly, instead of sub-linearly, with the number of changes in the underlying data distributions.

**Composed Randomizer.** The composed randomizer, $\tilde{\mathcal{R}}$, presented in this paper builds on the composed randomizer proposed by Bun et al. [2]. Their design focused on preserving the statistical distance between the distribution of the output of the composed randomizer, and joint distribution of $k$ independent randomized responses. This difference in the problem setting prompts the non-trivial changes in parameters, assumptions and analysis. The Bun et al. proof [2] relies extensively on concentration and anti-concentration inequalities. Their design can only achieve $c_{gap} \in O(\epsilon/\sqrt{k \ln(k/\epsilon) + (\epsilon/\sqrt{k \ln(k/\epsilon)})^{2/3}})$ (detailed discussion and proof are in Appendix A.2).

When the first term dominates (e.g., when $k \geq 1/\epsilon^2$), this simplifies to $c_{gap} \in O(\epsilon/\sqrt{k \ln(k/\epsilon)})$, which implies that $(c_{gap})^{-1} \in O(\sqrt{k \ln(k/\epsilon)}/\epsilon)$. If we apply this composed randomizer to our framework, according to Lemma 4.6, this leads to an error that scales at least with $\sqrt{k \ln(k/\epsilon)}/\epsilon$. In comparison, our composed randomizer reduces this to at most $\sqrt{\epsilon}/\epsilon$. Further, Bun et al.’s original design [2] applies only to offline inputs. We overcome this limitation by including our pre-computation technique so the algorithm becomes online.

**ACKNOWLEDGMENTS.** We thank the anonymous reviewers for their detailed feedback that helped us improve our work. Olga Ohrimenko is in part supported by a Facebook research award. Hao Wu is supported by an Australian Government Research Training Program (RTP) Scholarship.

**REFERENCES.**

[1] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. 2020. Practical Locally Private Heavy Hitters. J. Mach. Learn. Res. 21 (2020), 16:1–16:42.

[2] Mark Bun, Jelani Nelson, and Uri Stemmer. 2019. Heavy Hitters and the Structure of Local Privacy. ACM Trans. Algorithms 15, 4 (2019), 51:1–51:40.

[3] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private andContinual Release of Statistics. ACM Trans. Inf. Syst. Secur. 14, 3 (2011), 26:1–26:24.

[4] Luc Devroye and Gábor Lugosi. 2001. Combinatorial methods in density estimation. Springer.

[5] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately: In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA, Isabelle Guyon, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). ACM, 3571–3580.

[6] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5–8 June 2010, Leonard J. Schulman (Ed.). ACM, 715–724.

[7] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. Found. Trend. Comput. Sci. 9, 3–4 (2014), 211–407.

[8] Ullar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019; Timothy M. Chan (Ed.). SIAM, 2468–2479.

[9] Ullar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPORT: Randomized Aggregate Privacy-Preserving Ordinal Response. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3–7, 2014. Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 1054–1067.

[10] Giulia C. Fanti, Vasyl Pihur, and Ullar Erlingsson. 2016. Building a RAPPRO with the Unknown Privacy-Preserving Learning of Associations and Data Dictionaries. Proc. Priv. Enhancing Technol. 3, 6 (2016), 41–61.

[11] Matthew Joseph, Aaron Roth, Jonathan R. Ullman, and Bo Waggoner. 2018. Local Differential Privacy for Evolving Data. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3–8, 2018, Montréal, Canada, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). ACM, 2381–2390.

[12] Jiri Matousek and Jaroslav Nesetril. 2009. Invitation to Discrete Mathematics (2. ed.). Oxford University Press.

[13] Thang T. Nguyen, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Huiyun Shin, and Junbum Shin. 2016. Collecting and Analyzing Data from Smart Device Users with Local Differential Privacy. CoRR abs/1606.05053 (2016). arXiv:1606.05053.

[14] Herbert Robbins. 1955. A Remark on Stirling’s Formula. Amer. Math. Monthly 62, 1 (1955), 26–29.

[15] Differential Privacy Team. 2017. Learning with Privacy at Scale. Apple Machine Learning Journal 2017, 1 (2017), 1–25.

[16] Fleming Topsøe. 2001. Bounds for entropy and divergence for distributions over $\mathbb{R}$. J. Math. Anal. Appl. 257, 2 (2001), 820–831.

[17] Stanley L. Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. J. Amer. Statist. Assoc. 60, 309 (1965), 63–69. PMID: 12261830.

[18] Mingxun Zhou, Tianhao Wang, Hubert Chan, Giulia Fanti, and Elaine Shi. 2021. Locally Differentially Private Sparse Vector Aggregation. arXiv:2112.03449 [cs.CR]

[19] Ullar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2020. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. arXiv:1811.12469 [cs.LG]
A APPENDIX

Fact A.1 (Hoeffding’s Inequality [4]). Let $Y_1, \ldots, Y_n$ be independent real-valued random variables such that $|Y_i| \leq a_i$, $\forall i \in [n]$ with probability one. Let $Y = \Sigma_{i \in [n]} Y_i$, then for every $\eta \geq 0$:

$$\Pr[Y - E[Y] \geq \eta] \leq \exp\left(-\frac{2\eta^2}{\Sigma_{i \in [n]}(b_i - a_i)^2}\right),$$

and

$$\Pr[E[Y] - Y \geq \eta] \leq \exp\left(-\frac{2\eta^2}{\Sigma_{i \in [n]}(b_i - a_i)^2}\right).$$

Corollary A.2. Let $Y_1, \ldots, Y_n$ be independent real-valued random variables such that $|Y_i| \leq 1$, $\forall i \in [n]$ with probability one. Let $Y = \Sigma_{i \in [n]} Y_i$. Then, for $\beta \geq 0$, with probability at most $\beta$, it holds that

$$|Y - E[Y]| \geq \sqrt{2n \cdot \ln(2/\beta)}.$$

Fact A.3 (Stirling’s Approximation [12, 14]). For $n = 1, 2, \ldots$

$$\sqrt{2\pi n} \left(\frac{n}{e}\right)^n \exp\left(-\frac{n}{2 + 1}\right) \leq n! \leq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \exp\left(+\frac{1}{12n}\right).$$

Fact A.4 (Entropy Bound [16]). Let $H(x) = -x \log x + (1 - x)(\log(1 - x))$, $\forall x \in [0, 1]$ be the binary entropy function. Then

$$4x(1 - x) \leq H(x).$$

Corollary A.5. For each $x \in [-1/2, 1/2]$,

$$1 - 4x^2 \leq H(1/2 - x).$$

A.1 Proofs for Section 5

A.1.1 Lemma 5.2.

Proof of Lemma 5.2. Recall that $p = 1/(e^{\hat{\epsilon}} + 1)$, $1 - p = e^{\hat{\epsilon}}p$

$$g(i) = p^k(1 - p)^k - p^k \cdot e^{\hat{\epsilon}(k - i)}, \forall i \in [0..k],$$

and

$$p_{avg} = p^k(1 - p)^k \geq g(kp) = p^k \cdot e^{-e^{\hat{\epsilon}(k - kp)}}.$$  

By definitions of $g$ and $p_{avg}$, for each $j \in \mathbb{Z}$, it holds that

$$g(kp + j) = p_{avg} \cdot e^{-e^{\hat{\epsilon}j}}, \text{ if } 0 \leq kp + j \leq k,$n
$$g(kp - j) = p_{avg} \cdot e^{e^{\hat{\epsilon}j}}, \text{ if } 0 \leq kp - j \leq k.$$  

To prove the lemma, we consider $s \in Ann(b)$ and $s \in [-1, 1]^k \setminus Ann(b)$ separately. We show that for each $b \in [-1, 1]^k$, it holds that

$$\Pr[\hat{R}(b) = s] \leq 1/2^k, e^{2\sqrt{2}e}, g(LB) = e^{2\sqrt{2}e}.$$  

We will also prove that

$$g(kp) \geq 2^{-k} \geq g(k/2).$$  

Noting that $\|b - s\|_0 \in [LB..UB]$, we obtain

$$2^{-k} = g(UB) \leq \Pr[\hat{R}(b) = s] = g(\|b - s\|_0) \leq g(LB) = e^{2\sqrt{2}e} \cdot p_{avg},$$

which proves Inequality (34).

Proofing $g(kp) \geq 1/2^k \geq g(k/2)$. Via convexity of the function $y = -\log x$, we have

$$\log(g(kp)) = \frac{-p \log \frac{1}{p} - (1 - p) \log \frac{1}{1 - p}}{k} \geq -\log\left(\frac{p}{1 - p}\right) = -1.$$  

Via concavity of the function $y = \log x$, we have

$$\log\left(\frac{g(k/2)}{k}\right) = \frac{1}{2} \cdot \log p + \frac{1}{2} \cdot \log(1 - p) \leq \log\left(\frac{1}{2} \cdot p + \frac{1}{2} \cdot (1 - p) = -1.$$  

It follows that

$$p_{avg} = p(kp) \geq 1/2^k \geq g(k/2).$$

Proofing $g(UB) = 1/2^k$. Via Equation (32) and the definitions of $p$ and UB,

$$g(UB) = p^k \cdot \exp\left(-k \cdot \frac{1}{e^{\hat{\epsilon}}} \cdot \ln\left(\frac{2e^{\hat{\epsilon}}}{e^{\hat{\epsilon}} + 1}\right)\right) = \left(\frac{1}{e^{\hat{\epsilon}} + 1}\right)^k \cdot \exp\left(-k \cdot \frac{1}{e^{\hat{\epsilon}}} \cdot \ln\left(\frac{2e^{\hat{\epsilon}}}{e^{\hat{\epsilon}} + 1}\right)\right) = \exp\left(k \ln\left(\frac{1}{e^{\hat{\epsilon}} + 1}\right) + \frac{k}{e^{\hat{\epsilon}}} \cdot \ln\left(\frac{2e^{\hat{\epsilon}}}{e^{\hat{\epsilon}} + 1}\right)\right) = \exp\left(k \ln\left(\frac{1}{e^{\hat{\epsilon}} + 1}\right) - \frac{k}{e^{\hat{\epsilon}}} \cdot \ln\left(\frac{2e^{\hat{\epsilon}}}{e^{\hat{\epsilon}} + 1}\right)\right) = 1/2^k.$$  

Therefore, $UB \in [kp, k/2]$.

Proof of Inequality (35). Let $\{LB..UB\} = [0..k] \setminus \{LB..UB\}$. Denote $\mathcal{R}(b) \equiv (\mathcal{R}(b_1), \ldots, \mathcal{R}(b_k))$. First, observe

$$\Pr[\mathcal{R}(b) \notin Ann(b)] = \sum_{i \in [LB..UB]} \binom{k}{i} \cdot g(i).$$

As each $s \notin Ann(b)$ is outputted uniformly at random by $\mathcal{R}$ when $b' = \mathcal{R}(b) \notin Ann(b)$, it holds that

$$\Pr[\mathcal{R}(b) = s] = p_{out}^* = \sum_{i \in [LB..UB]} \binom{k}{i} \cdot g(i).$$

We upper bound and lower bound $p_{out}^*$ separately.

Upper Bound For $p_{out}^*$: As $g$ is decreasing and $g(UB) = 1/2^k$, it holds that

$$\sum_{i \in [LB..UB]} \binom{k}{i} \cdot g(i) \geq \sum_{i \in [LB..UB]} \binom{k}{i} \cdot 1/2^k.$$  

Combining with the fact that

$$\sum_{i \in [0..k]} \binom{k}{i} \cdot g(i) = 1 = \sum_{i \in [0..k]} \binom{k}{i} \cdot 1/2^k.$$  


we obtain
\[
\sum_{i \in [L_B \ldots U_B]} \binom{k}{i} g(i) \leq \sum_{i \in [L_B \ldots U_B]} \binom{k}{i} \cdot 1/2^k.
\]
It concludes that \(P^* \leq 1/2^k\).

**Lower Bound For** \(P_{\text{out}}\). Since \(LB + UB \leq kp - 2\sqrt{k} + k/2 < k\), it holds that \(k - LB > UB\). Hence, we can partition the numerator and denominator of \(P^* \) into three parts:

\[
N_1 = \sum_{i=0}^{LB-1} \binom{k}{i} g(i), \quad N_2 = \sum_{i=UB+1}^{LB-1} \binom{k}{i} g(i), \quad N_3 = \sum_{i=UB+1}^{k} \binom{k}{i} g(i).
\]

We will prove that
\[
\frac{N_2}{\mathcal{D}_2} \geq \frac{1}{2} \cdot \frac{1}{k} \cdot \mathcal{D}_2 \cdot \mathcal{D}_3 \geq \frac{1}{2} \cdot \frac{1}{k} \cdot \mathcal{D}_3
\]
It follows that
\[
P^* = \frac{N_1 + N_2 + N_3}{\mathcal{D}_1 + \mathcal{D}_2 + \mathcal{D}_3} \geq \frac{1}{2} \cdot \frac{1}{k} \cdot \mathcal{D}_3
\]
Similarly, using that \(k - LB = kp + k(1 - 2p) + 2\sqrt{k}\), we get
\[
g(k - (LB - j)) = g(kp + k(1 - 2p) + 2\sqrt{k} + j)
\]
\[
= \frac{1}{2} \cdot \frac{1}{k} \cdot \mathcal{D}_3
\]
Via the inequality that \(x + 1/x \geq 2\), \(\forall x > 0\),
\[
g(LB - j) + g(k - (LB - j)) \geq \frac{1}{2} \cdot \frac{1}{k} \cdot \mathcal{D}_3
\]
\[
= \frac{1}{2} \cdot \frac{1}{k} \cdot \mathcal{D}_3
\]
It concludes that
\[
N_1 + N_3 \geq \sum_{j=1}^{LB-1} \binom{k}{j} \cdot \mathcal{P}_{\text{avg}} \cdot 2 \cdot e^{-\sqrt{k}(1 - 2p)}
\]
Noting that \(\mathcal{D}_1 + \mathcal{D}_3 = 2 \cdot \sum_{j=1}^{LB-1} \binom{k}{j} \) finishes the proof.

\[\square\]

**Lemma 5.3.**

**Proof of Lemma 5.3.** By symmetry of \(\hat{R}\), it suffices to prove that Inequality (5.3) holds for the first coordinate of \(\hat{b}\).

We study the probability of \(Pr[\hat{b}_1 = b_1]\). Recall that for an input \(b \in \{-1, 1\}^k\), \(\hat{R}\) first generates \(b' = R(b) = (R(b_1), \ldots, R(b_k))\). If \(b' \in \text{Ann}(b)\), \(\hat{R}\) outputs it directly. Therefore, for each \(s \in \text{Ann}(b)\), it holds that
\[
Pr[\hat{b} = s] = Pr[\hat{R}(b) = s] = g(||b - s||_0).
\]
For each \(i \in [L_B \ldots U_B]\), consider the set \(s \in \{-1, 1\}^k : ||s - b||_0 = i \subset \text{Ann}(b)\). The set contains \(\binom{k}{i}\) sequences such that each sequence in the set is outputted by \(\hat{R}\) with probability
\[
Pr[\hat{b} = s] = Pr[\hat{R}(b) = s] = g(||b - s||_0) = g(i).
\]
Note that each sequence in the set \(s \in \{-1, 1\}^k : ||s - b||_0 = i \subset \text{Ann}(b)\) differs in \(i\) coordinates from \(b\). If we pick a sequence uniformly at random from this set, with probability \(i/k\), we obtain a sequence \(s\) such that \(s_1 \neq b_1\). It follows that the fraction of sequences in the set whose first coordinate equals \(b_1\) is given by \((k - i)/k\). Therefore,
\[
Pr[\hat{b}_1 = b_1, \|\hat{b} - b\|_1 = i] = \binom{k}{i} g(i) \frac{k - i}{k}.
\]
Similarly, we can prove that for each \(i \in [L_B \ldots U_B]\),
\[
Pr[\hat{b}_1 = b_1, \|\hat{b} - b\|_1 = i] = \binom{k}{i} p_{\text{out}} \frac{k - i}{k}.
\]
Summing over \(i \in [0 \ldots k]\), we get
\[
Pr[\hat{b}_1 = b_1] = \sum_{i=0}^{UB} \binom{k}{i} g(i) \frac{k - i}{k} + P_{\text{out}} \sum_{i=UB}^{LB-1} \binom{k}{i} \frac{k - i}{k},
\]
where \([L_B \ldots U_B] = [0 \ldots LB - 1] \cup [UB + 1 \ldots k]\). Similarly,
\[
Pr[\hat{b}_1 = -b_1] = \sum_{i=0}^{UB} \binom{k}{i} g(i) \frac{i}{k} + P_{\text{out}} \sum_{i=UB}^{LB-1} \binom{k}{i} \frac{i}{k}.
\]
It follows that
\[
gap = \sum_{i \in \overline{\text{LB}} \cup \overline{\text{UB}}} (\binom{k}{i}) \left( \frac{k - 2i}{k} \right) + \sum_{i \in \overline{\text{LB}} \cup \overline{\text{UB}}} p_{\text{out}}(\binom{k}{i}) \left( \frac{k - 2i}{k} \right). \tag{42}
\]
For each \( i \in [0 \ldots k] \), it holds that
\[
\binom{k}{i} \frac{k - 2i}{k} + \binom{k}{i - 1} \frac{k - 2i}{k} = \binom{k}{i} \left( 2 - \binom{k - 1}{i - 1} \right) = 0.
\]
For each \( i \in \overline{\text{LB}} \cup \overline{\text{UB}} \), if \( i \not\in [\text{UB} - \sqrt{k} \ldots \text{UB} - \sqrt{k}/2] \), then \( k - i \not\in \overline{\text{LB}} \cup \overline{\text{UB}} \). Therefore, by pairing up \( \binom{k}{i} \frac{k - 2i}{k} \) with \( \binom{k}{i} \frac{k - 2(k - i)}{k} \) for each \( i \in [\overline{\text{LB}} \cup \overline{\text{UB}}] \setminus [\text{UB} - \sqrt{k} \ldots \text{UB} - \sqrt{k}/2] \), we obtain
\[
\sum_{i \in [\overline{\text{LB}} \cup \overline{\text{UB}}]} \binom{k}{i} \frac{k - 2i}{k} = \sum_{i \in [\text{UB} - \sqrt{k} \ldots \text{UB} - \sqrt{k}/2]} \binom{k}{i} \frac{k - 2i}{k}.
\]
Therefore,
\[
gap = \sum_{i \in \overline{\text{LB}}} \binom{k}{i} g(i) \frac{k - 2i}{k} + \sum_{i \in \overline{\text{LB}}} p_{\text{out}}(\binom{k}{i}) \frac{k - 2i}{k} = \sum_{i \in \overline{\text{LB}}} \binom{k}{i} g(i) - \sum_{i \in \overline{\text{LB}}} p_{\text{out}}(\binom{k}{i}) \frac{k - 2i}{k}.
\]
Since \( \text{UB} > \text{LB} = k - 2\sqrt{k} \), it holds that \([\text{UB} - 2\sqrt{k} \ldots \text{UB} - \sqrt{k}/2] \subset [\overline{\text{LB}} \cup \overline{\text{UB}}] \). According to Inequality (35), we have \( p_{\text{out}}(\binom{k}{i}) \leq 1/2^k \). Via that \( g(i) \) is a decreasing function, Equality (33) and Inequality (37), we get that for each \( i \in [\text{UB} - 2\sqrt{k} \ldots \text{UB} - \sqrt{k}/2] \)
\[
g(i) \geq g(\text{UB} - \sqrt{k}/2) = e^{\sqrt{k}/2} \cdot p_{\text{avg}} \geq e^{\sqrt{k}/2} \cdot 1/2^k.
\]
Further, via that \( \text{UB} \leq k/2 \), for each \( i \in [\text{UB} - 2\sqrt{k} \ldots \text{UB} - \sqrt{k}/2] \),
\[
k - 2i \geq \left( 1 - \frac{1}{2} \cdot \frac{\text{UB} - \sqrt{k}/2}{\text{UB} - \sqrt{k}/2} \right) = \left( \frac{1}{\sqrt{k}} + 1 - \frac{1}{\text{UB} - \sqrt{k}/2} \right) \geq 1/\sqrt{k}.
\]
Now we can lower bound \( \text{gap} \) by
\[
gap \geq \sum_{i = \text{UB} - 2\sqrt{k}} \binom{k}{i} \left( e^{\sqrt{k}/2} \cdot \frac{1}{2} \right) - \left( \frac{1}{2} \right) \frac{1}{\sqrt{k}}
\geq \sum_{i = \text{UB} - 2\sqrt{k}} \binom{k}{i} \left( e^{\sqrt{k}/2} \cdot \frac{1}{2} \right) \frac{1}{\sqrt{k}}
\geq k \left( \frac{\sqrt{k}}{2} \right) \frac{1}{\sqrt{k}} \frac{k}{2} \frac{1}{\sqrt{k}} = \frac{k}{\sqrt{k}} \frac{1}{\sqrt{k}} = \frac{k}{\sqrt{k}} \frac{1}{\sqrt{k}}.
\]
To prove that \( \text{gap} \in \Omega(\sqrt{k}) = \Omega(\epsilon/\sqrt{k}) \), it suffices to prove that
\[
\sum_{i = \text{UB} - 2\sqrt{k}} \binom{k}{i} \left( \frac{1}{2} \right) \frac{k}{\sqrt{k}} \in O(1).
\]
Recall that \( k \leq \text{UB} \leq k/2 \). Further, we have \( k - 2k = k \cdot \frac{\epsilon^2 - 1}{2(\epsilon^2 + 1)} \leq \frac{k}{2} \epsilon \leq \frac{k}{2} \epsilon \), where the first inequality follows from that \( (\epsilon^2 - 1)/(\epsilon^2 + 1) \leq x/2 \) for all \( x \geq 0 \), the second one from that the assumption that \( \epsilon \leq 1/\sqrt{k} \). Hence, \( \text{UB} \geq k/2 - \sqrt{k}/4 \) and
\[
\sum_{i = \text{UB} - 2\sqrt{k}} \binom{k}{i} \left( \frac{1}{2} \right) \frac{k}{\sqrt{k}} \geq \sum_{i = k/2 - 2\sqrt{k}} \binom{k}{i} \left( \frac{1}{2} \right) k^k.
\]
Finally,
\[
\sum_{i = k/2 - 2\sqrt{k}} \binom{k}{i} \left( \frac{1}{2} \right) k^k \geq \sum_{i = k/2 - 2\sqrt{k}} \binom{k}{i} \left( \frac{1}{2} \right) k^k = \binom{k}{1} \left( \frac{1}{2} \right) k.
\]
Via that \( 1 - x \leq e^x \), \( x \in \mathbb{R} \), we get
\[
1 - \frac{1}{4(\sqrt{k} - 1/k + \sqrt{k})} \geq \frac{1}{1 - 1/4(\sqrt{k} - 1/k + \sqrt{k})}.
\]
When \( k \geq 4\sqrt{k} \), we see \( \exp(-4k^2 - 1) = \exp(-4k) \), and \( \sqrt{k} - 1/k + \sqrt{k} \) \( \leq 8/\sqrt{k} \). Hence,
\[
k_2 \geq \sum_{i = k/2 - 2\sqrt{k}} \binom{k}{i} \frac{k}{k - 2\sqrt{k}} \frac{\sqrt{k}}{2} \left( 1 - e^{-1/\sqrt{k}} \right) \geq \left( \frac{k}{k - 2\sqrt{k}} \right) \frac{\sqrt{k}}{9}.
\]
Using the Stirling’s approximation (Fact A.3), we get
\[
\left( \frac{k}{k - 2\sqrt{k}} \right) \frac{\sqrt{k}}{9} \geq e^{-1/6} \cdot \frac{k^k}{2 \cdot \pi \cdot (k/2 - \sqrt{k}) \cdot (k/2 + \sqrt{k})} \cdot \left( k/2 - \sqrt{k} \right) \left( k/2 + \sqrt{k} \right) \geq e^{-1/6} \cdot \frac{1}{\sqrt{k} - 4} \cdot \frac{2}{\pi} \cdot 2^k H(1/2 - 1/\sqrt{k}).
\]
where $H(\cdot)$ is the entropy function. We obtain that
\[
\sum_{i=0}^{k-2\sqrt{k}} \frac{1}{i^2} \geq 9 \sqrt{\frac{k}{k-4}} \cdot e^{-1/6} \cdot \frac{\sqrt{2}}{\pi} \cdot 2^k (H(1/2-1/\sqrt{k})-1)
\]
Via the inequality (Corollary A.5) that $H(1/2-x) \geq 1 - 4x^2$, $\forall x \in [-1/2, 1/2]$,
\[
\sum_{i=0}^{k-2\sqrt{k}} \frac{1}{i^2} \geq 9 \sqrt{\frac{k}{k-4}} \cdot e^{-1/6} \cdot \frac{\sqrt{2}}{\pi} \cdot 2^k (\frac{4}{\sqrt{k}}-1) \in \Omega(1),
\]
which finishes the proof. □

A.2 Composed Randomizer by [2]

The composed randomizer proposed in [2] shares the same pseudo-code as ours, but with different parameter setting and assumptions. For convenience, we repeat the pseudo-code here.

**Algorithm 4 Composed Randomizer by [2]**

1. **Procedure** Basic Randomizer $\mathcal{R}(\xi)$
   - Input: Value $\xi \in \{-1, 1\}$
   - return $-\xi$ w.p. $1/(e^{\xi}+1)$ and $\xi$ w.p. $e^{\xi}/(e^{\xi}+1)$.
2. **Procedure** Composed Randomizer $\tilde{\mathcal{R}}(b)$
   - Input: Vector $b \in \{-1, 1\}^k$
   - Sample $b' \leftarrow (\mathcal{R}(b_1), \ldots, \mathcal{R}(b_k))$.
   - if $b' \notin \text{Ann}(b)$ then
   - $b' \leftarrow (\{-1, 1\}^k \setminus \text{Ann}(b))$
   - return $b'$

Denote $p = 1/(e^{\tilde{\xi}}+1)$. The composed randomizer proposed in [2] sets
\[
\text{LB} \doteq kp - \sqrt{\frac{k}{2} \ln \frac{2}{p}}, \quad \text{UB} \doteq kp + \sqrt{\frac{k}{2} \ln \frac{2}{p}}.
\]
for some additional parameter $\lambda \in (0,1)$. Accordingly, let
\[
\text{Ann}(b) \doteq \{x \in \{-1, 1\}^k : \|b-x\|_p \in [\text{LB}, \text{UB}]\}
\]
The pseudo-code of the composed randomizer [2] is presented in Algorithm 4. The work [2] proves the following fact.

**Fact A.6 (Algorithm 4 [2])**. Suppose that
\[
0 < \lambda < \left(\tilde{\xi} \sqrt{k}/(2(k+1))\right)^{2/3},
\]
and that
\[
\epsilon = 6\tilde{\xi} \sqrt{k} \ln(1/\lambda) \leq 1,
\]
the algorithm $\tilde{\mathcal{R}}$ is $\epsilon$-differentially private, further
\[
\text{Pr}[^{\tilde{\mathcal{R}}(b) \in \text{Ann}(b)}] \geq 1 - \lambda.
\]
We can derived the following constraint between $\lambda, \epsilon$ and $k$.

**Theorem A.7**. If $\lambda$ satisfies Inequality (45) and Equality (46), then
\[
\ln(1/\lambda) \in \Omega(\ln(k/\epsilon))\quad (48)
\]
and
\[
\lambda \in O\left(\frac{\epsilon}{k \cdot \ln(k/\epsilon)}\right)^{2/3}\quad (49)
\]

**Proof**. Substituting $\tilde{\epsilon} = \frac{\epsilon}{6\sqrt{k} \ln(1/\lambda)}$ into Inequality (45), we get
\[
\lambda^{3/2} < \frac{\epsilon^{\sqrt{k}}}{6\sqrt{k} \ln(1/\lambda)} \cdot 2(k+1)
\]
and hence, $\frac{3}{2} \ln \lambda + \frac{1}{2} \ln \frac{1}{\lambda} < \ln 2^{(k+1)}$. This proves that $\ln(1/\lambda) \in \Omega(\ln(k/\epsilon))$. Substituting this back to Inequality (50), we obtain
\[
\lambda \in O\left(\frac{\epsilon^{\sqrt{k}}}{k \ln(k/\epsilon)}\right)^{2/3}.
\]

**Theorem A.8.** For a given input $b$, denote $\tilde{b} = \tilde{\mathcal{R}}(b)$ the output of $\tilde{\mathcal{R}}$. For each input $b \in \{-1, 1\}^k$, and for each $i \in [k]$, it holds that
\[
\text{Pr}[\tilde{b}_i = b_i] - \text{Pr}[\tilde{b}_i = -b_i] = \epsilon_{\text{gap}} \in O\left(\frac{\epsilon}{\sqrt{k \ln(k/\epsilon)}} + \frac{\epsilon}{k \ln(k/\epsilon)}\right)^{2/3}.
\]

**Proof of Theorem A.8.** Let $\tilde{b} = \tilde{\mathcal{R}}(b)$. If $\tilde{b}_i = b_i$, either of the following events happens
(1) $\mathcal{R}(b_i) = b_i$,
(2) $b' = \mathcal{R}(b) \notin \text{Ann}(b)$, and it is replaced with a random sample from $\{-1, 1\}^k \setminus \text{Ann}(b)$.

The former happens with probability $e^{\tilde{\xi}}/(e^{\tilde{\xi}} + 1)$ and the later happens with probability at most $\lambda$ according to Inequality (47). Via union bound, we have
\[
\text{Pr}[\tilde{b}_i = b_i] - \text{Pr}[\tilde{b}_i = -b_i] = \epsilon_{\text{gap}} \in O\left(\frac{\epsilon}{\sqrt{k \ln(k/\epsilon)}} + \frac{\epsilon}{k \ln(k/\epsilon)}\right)^{2/3}.
\]
Combining with Inequality (49), we have
\[
\epsilon_{\text{gap}} \leq \left(\frac{e^{\tilde{\xi}} - 1}{e^{\tilde{\xi}} + 1}\right) + 2 \cdot \lambda.
\]
By Inequality (46) and (48), we have
\[
\tilde{\epsilon} = \frac{\epsilon}{6\sqrt{k} \ln(1/\lambda)} \in O\left(\frac{\epsilon}{\sqrt{k \ln(k/\epsilon)}} + \frac{\epsilon}{k \ln(k/\epsilon)}\right)^{2/3}.
\]

□