Collaborative Search Log Sanitization: Toward Differential Privacy and Boosted Utility

Yuan Hong, *Member, IEEE,* Jaideep Vaidya, *Senior Member, IEEE,* Haibing Lu, *Member, IEEE,* Panagiotis Karras, *Member, IEEE,* and Sanjay Goel, *Member, IEEE*

Abstract—Severe privacy leakage in the AOL search log incident has attracted considerable worldwide attention. However, all the web users' daily search intents and behavior are collected in such data, which can be invaluable for researchers, data analysts and law enforcement personnel to conduct social behavior study [14], criminal investigation [5] and epidemics detection [10]. Thus, an important and challenging research problem is how to sanitize search logs with strong privacy guarantee and sufficiently retained utility. Existing approaches in search log sanitization are capable of only protecting the privacy under a rigorous standard [24] or maintaining good output utility [25]. To the best of our knowledge, there is little work that has perfectly resolved such tradeoff in the context of search logs, meeting a high standard of both requirements. In this paper, we propose a sanitization framework to tackle the above issue in a distributed manner. More specifically, our framework enables different parties to collaboratively generate search logs with boosted utility while satisfying *Differential Privacy*. In this scenario, two privacy-preserving objectives arise: first, the collaborative sanitization should satisfy differential privacy; second, the collaborative parties cannot learn any private information from each other. We present an efficient protocol –**C**ollaborative s**E**arch **L**og **S**anitization (**CELS**) to meet both privacy requirements. Besides security/privacy and cost analysis, we demonstrate the utility and efficiency of our approach with real datasets.

Index Terms—Search Log, Differential Privacy, Sampling, Optimization, Secure Multiparty Computation

1 Introduction

Web search logs contain a large volume of Internet users' posed queries and clicked urls. Such data gives great insight into human behavior via their search intents, then it can be used to examine the observed patterns of human behavior as well as to draw important predictive inferences, e.g., predicting the pattern of flu spread during the flu-season, estimating the customer needs and market trends, and identifying the popularity of electoral candidates and economic confidence. Search engines themselves use it to improve ranking [21], detect common spelling errors [2], and recommend similar queries [7]. Researchers, data analysts and law enforcement personnel also analyze it for deriving the human living habits [14], investigating criminal activities [5] or detecting epidemics [10]. It is also an important tool for the government for shaping public policy based on user concerns and opinions captured through web logs.

There are millions of search queries each day and the information is logged and stored for analysis. However, one problem with the storage and the possible release of search logs is the potential for privacy breach. Although search logs may be stored or published by replacing the sensitive user-IDs with pseudonyms (i.e. the published data in AOL incident [14]), there are hidden clues in the search data that can reveal the identity of the user. Also, the queries may reveal their most private interests and concerns which can be embarrassing (e.g., sexual habits) or lead to discrimination (e.g., health issues). Then, if search log data is released without sanitization or with trivial anonymization like the AOL data, many individuals might be easily re-identified by adversarial data recipients with some prior knowledge, and then web users' entire sensitive search information will be explicitly disclosed to adversaries. In the case of AOL 20 million records were released with only pseudo IDs replacing the names. However, it was possible to identify many users from the dataset which subsequently resulted in a class action lawsuit against AOL. [4], [14] illustrated that it can only take a couple of hours to breach a particular user's privacy in the absence of good anonymization. Thus, it is crucial to sanitize search logs appropriately before storing, analyzing or possibly publishing them.

There has been prior work on anonymization of relational databases (e.g., k-anonymity [42]), yet it is not directly aligned due to significant differences between search logs and relational databases [25]. Some recent research results have been proposed to sanitize search logs along two dimensions. On one hand, each of [5], [17], [22], [25], [27], [30] presented an anonymization method that satisfies its defined privacy notion (mostly the variants of k-anonymity) and attempts to retain good utility. However, the privacy notion in each of the above work entails a high risk of breaching privacy in case of adversaries holding certain prior knowledge. On the other hand, some search log sanitization techniques [13], [24] have been developed based on a more rigorous notion – *Differential Privacy* [8], which provides strong privacy guarantee even if the adversaries hold arbitrary prior knowledge. However, the released dataset in [13], [24] is the statistical information of queries and clicks where all users' search queries and clicks are aggregated without individual attribution. The data

- *Y. Hong and S. Goel are with the Department of Information Technology Management, State University of New York at Albany. E-mail: {hong, goel}@albany.edu*
- *J. Vaidya is with the Department of Management Science and Information Systems, Rutgers University. E-mail: jsvaidya@business.rutgers.edu*
- *H. Lu is with the Department of Operations Management and Information Systems, Santa Clara University. E-mail: hlu@scu.edu*
- *P. Karras is with Skolkovo Institute of Science and Technology. E-mail: P.Karras@skoltech.ru*

Manuscript received 14 Nov. 2013; revised 30 June 2014; accepted 24 Sept. 2014; published online XX.

utility might be greatly damaged since the association between different query-url pairs has been removed. With the output, we can neither develop personalized query recommendation, nor can we carry out human behavior research and criminal investigation since the output data no longer include the individual web user's IDs. Therefore, the output utility of the existing differentially private search log sanitization work [13], [24] is not that satisfactory.

1.1 Contributions

There is little work that has perfectly resolved such tradeoff in the context of search logs – by generating the output which simultaneously achieves a high standard of both privacy protection and output utility. In this paper, to the best of our knowledge, we take a first step towards addressing this deficiency by presenting a novel sanitization framework. The main contributions are summarized as below:

- **Utility.** The existing differentially private algorithms [13], [24] only made the statistical information of queries and clicks publishable, and broke the association of different queries/clicks posed by the same user. This leads to huge utility loss. We propose a novel differentially private mechanism that samples the output with *identical schema as the input*. Thus, the sanitized search log can be analyzed in exactly the same fashion and for the same purposes as the input. E.g., it enables personalized query recommendation, individual human behavior study, criminal investigation, among others.
  Meanwhile, to further boost the utility, we build a collaborative sanitization framework that enables different data owners to produce a collaborative publishable search log, which significantly improves the utility as compared to publishing on their own (this benefits society greatly [1]). We also show the boosted utility in experiments.
- **Privacy and Security.** Practically, involving untrusted participants into the sanitization would pose additional privacy and security concern to every party since each of them holds their own private input data. We present an efficient protocol for collaborative parties, namely **C**ollaborative s**E**arch **L**og **S**anitization (**CELS**), which ensures: 1) the collaborative sanitization over all parties satisfy differential privacy, and 2) the collaborative parties cannot learn any private information from each other.
  Also, we prove differential privacy and protocol security for CELS and experimentally evaluate the performance of our approach with real datasets.

The remainder of this paper is organized as follows. In Sec. 2, some preliminaries for this work are introduced. We present the sampling mechanism and derive the conditions for satisfying differential privacy in Sec. 3. Then, we propose the sanitization framework and the corresponding analysis in Sec. 4 and 5 respectively. Sec. 6 gives experimental results, and Sec. 7 reviews the related literature. Finally, Sec. 8 concludes the paper and discusses the future work.

1. For instance, since the data can be proven to be publishable, various external researchers can now conduct human behavior study on large quantities of search logs collected from various search engines. Neither data owners nor data recipients need to worry about the breach of data privacy.

2 Preliminaries

2.1 Search Log Data

Web users' most sensitive values in their search history are the click-through information. Sometimes queries are more sensitive than the clicked urls (e.g., query "diabetes medicine" and click "www.walmart.com"), or vice versa (e.g., query "medicine" and click "www.cancer.gov"). We thus consider each distinct click-through query-url pair (simply denoted as query-url pair) as a single sensitive item. We let $Q = \{\phi_1, \phi_2, \ldots, \phi_n\}$ be the query-url pair universe where n is the cardinality of Q. Given a search log D including m different users' logged search information, we denote any user u_j (where $1 \leq j \leq m$)'s share in D as:

Definition 1: (User Log U_j) User u_j's all query tuples in D, where every single tuple $[u_j, \phi_i, c_{ij}] \in U_j$ includes a user-ID u_j, a click-through query-url pair ϕ_i, and the count of ϕ_i posed by user u_j ($1 \leq i \leq n$ and $1 \leq j \leq m$).

Search log $D = \{U_1, U_2, \ldots, U_m\}$ consists of all the users $u_1, \ldots, u_m$'s shares. Table 1 gives an example for the search log: non-negative integer at the ith column and the jth row ($1 \leq i \leq n$, $1 \leq j \leq m$) indicates the count of query-url pair ϕ_i posed by user u_j.

TABLE 1
An Example of Search Log D

	Query-url pairs (Sensitive Items)					
users	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6
u_1	3	2	0	1	0	0
u_2	1	1	0	0	4	0
u_3	2	1	0	1	0	2
u_4	0	1	0	0	1	1
u_5	1	0	7	0	0	5
u_6	0	0	1	1	0	0
u_7	0	1	0	0	2	0
u_8	1	0	5	0	0	1
Total (c_i)	8	6	13	3	7	9

2.2 Utility Measure

Search log analysis has been widely used to function many practical applications, such as search results ranking [21], query suggestion and recommendation [7], and spelling correction [2]. In our prior work [26], we have developed differentially private sanitization algorithms with three different measures – total output count, frequent query-url pairs utility and query-url pair diversity. We now define an all-round utility measure for the sanitization instead.

Measuring the count ratio difference of all the query-url pairs in the input and output is a generic way of evaluating utility. However, Deng et al. [6] has noted that, such approach is inherently biased towards heavily-clicked urls. In order to balance this bias, we employ an entropy-biased utility measure to evaluate the information loss in the sanitization. Considering the count ratio of all query-url pairs in the input/output search log as two *probability distribution functions (pdf)*, we define the utility loss function using Kullback-Leibler(KL) divergence between these two *pdf*s:

$$\mathcal{D}_{KL} = \sum_{\forall \phi_i \in Q} \left[\frac{c_i}{|D|} \log \frac{c_i/|D|}{x_i/|O|} \right] \quad (1)$$

where $c_i = \sum_{j=1}^{n} c_{ij}$ (recall that n is the number of all unique query-url pairs), and $|D|$ and $|O|$ represent the total count of all query-url pairs in the input and output respectively. Note that x_i might be enforced to be 0 to satisfy differential privacy in the sanitization. At this time, $\mathcal{D}_{KL}$ may approximate $+\infty$. Thus, we slightly revise Equation 1: $x_i \implies x_i + 1$ and $|O| = \sum_{i=1}^{n} x_i \implies \sum_{i=1}^{n}(x_i + 1) = |O| + n$.

$$\mathcal{D}_{KL}(D, O) = \sum_{i=1}^{n} \left[\frac{c_i}{|D|} \log\left(\frac{c_i}{|D|} \cdot \frac{|O| + n}{x_i + 1}\right) \right] \tag{2}$$

The revised utility loss function $\mathcal{D}_{KL}(D, O)$ lies very close to $\mathcal{D}_{KL}$ and avoids the zero denominator constraint, then we regard it as our utility measure.

2.3 Privacy Notion

We consider two search logs D and D' to be neighbors if they differ in any user log U_j, resulting in the following three *Cases*: (1) $D = D' \bigcup U_j$, (2) $D' = D \bigcup U_j$, and (3) U_j is different in D and D'. However, ensuring ϵ-differential privacy is not always feasible: e.g., if an output O includes an item in D but not in D', such as user-ID u_j in (1), the probability of generating O from D' is zero while the probability of generating O from D is non-zero, hence the ratio between the probabilities cannot be bounded by e^{ϵ} due to a zero denominator. We thus employ the relaxed privacy notion:

Definition 2 ((ϵ, δ)-differential privacy): A randomization algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy if for all neighboring inputs D and D' and any set of possible outputs S, we have $Pr[\mathcal{A}(D) \in S] \leq e^{\epsilon} Pr[\mathcal{A}(D') \in S] + \delta$ and $Pr[\mathcal{A}(D') \in S] \leq e^{\epsilon} Pr[\mathcal{A}(D) \in S] + \delta$.

More specifically, if both $Pr[\mathcal{A}(D) \in S]$ and $Pr[\mathcal{A}(D') \in S]$ are positive, then $e^{-\epsilon} \leq \frac{Pr[\mathcal{A}(D) \in S]}{Pr[\mathcal{A}(D') \in S]} \leq e^{\epsilon}$; if $Pr[\mathcal{A}(D) \in S] = 0$, then $Pr[\mathcal{A}(D') \in S] \leq \delta$ and vice-versa. Our sanitization satisfies the above notion.

3 Differential Privacy Guarantee

In this section, we present our sampling and show that differential privacy for sampling is guaranteed by satisfying some constraints.

3.1 Multinomial Sampling

In fact, the input search log D can be considered as an "Input user-query-url Histogram" $\{\forall c_{ij}\}$, and a coarse-grained "Input query-url Histogram" $c = (c_1, \ldots, c_n)$ can be aggregated by collecting the counts of all n distinct query-url pairs in D. Similarly, we regard those histograms in the output O as "Output user-query-url Histogram" $\{\forall x_{ij}\}$ and "Output query-url Histogram" $x = (x_1, \ldots, x_n)$.

Definition 3 (Multinomial Sampling with D and x): Given input search log D (fine-grained input user-query-url histogram) and the aggregated output counts $x = (x_1, \ldots, x_n)$, sampling a fine-grained output user-query-url histogram $O = \{\forall x_{ij}\}$ with multinomial distribution is the process of sampling specific user-IDs for every query-url pair $\phi_i \in Q$ through x_i multinomial trials, where the probability of every sampled outcome in one trial is given as the ratio $\frac{c_{ij}}{c_i}$, for each query-url pair j (derived from input D).

An example of the multinomial sampling is given in Fig. 1, assuming that $(x_1, \ldots, x_n) = (0, 3, 20, 0, 4)$. We thus have:

1) In every multinomial trial for any query-url pair ϕ_i, the probability that any user-ID u_j is sampled, is $\frac{c_{ij}}{c_i}$. E.g., "car price, kbb.com" in Fig. 1, the probability that user 082 is sampled is $\frac{2}{0+2+5}$. However, the probability that user 081 is sampled for ϕ_i is 0.
2) While sampling user-IDs for query-url pair ϕ_i, since the expected value of every random variable x_{ij} is $E(x_{ij}) = x_i \cdot \frac{c_{ij}}{c_i}$, the shape of sampled user-query-url histograms for ϕ_i is analogous to the input (guaranteed by multinomial distribution). E.g., while sampling user-IDs for "google, google.com", even if the output count $x_i = 20 < c_i = 15 + 7 + 17 = 39$, the shape of a sampled histogram {8, 3, 9} (see Fig. 1(b)) and {15, 7, 17} (in the input) is similar.

Thus, the output user-query-url histogram (see Fig. 1(b)) has the identical schema as the input D, which can be also considered as a user-query-url histogram.

3.2 Differential Privacy for Sampling

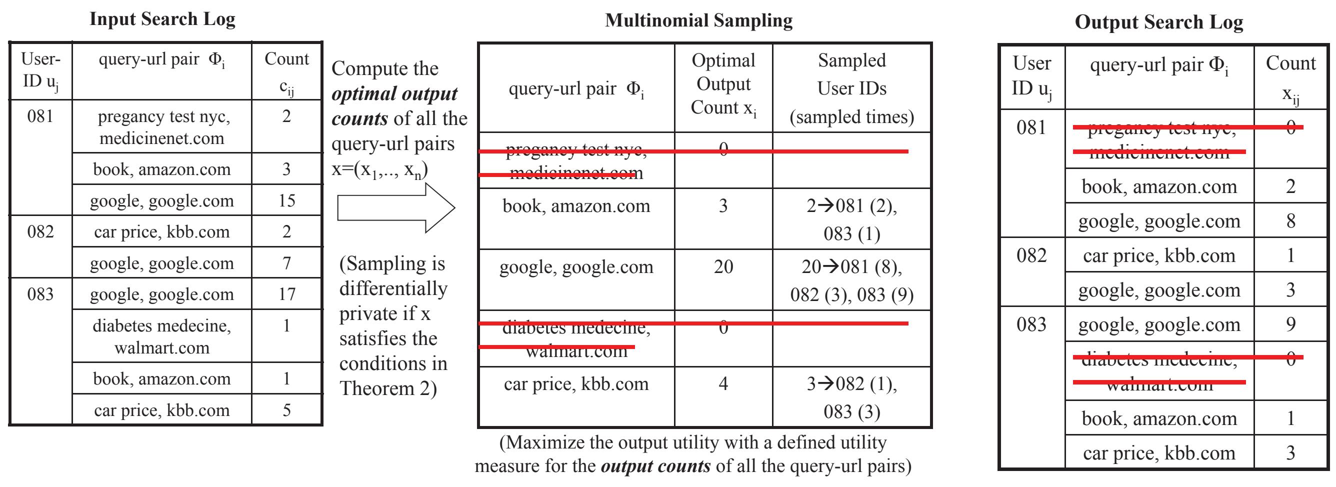
As described above, if $x = (x_1, \ldots, x_n)$ and D are given, multinomial sampling can generate probabilistic outputs (with identical schema as D). We now discuss how (ϵ, δ)-differential privacy can be achieved in the sampling for any two neighboring inputs D and D'. Recall that the relationship between D and D' has three different cases, we first look at Case (1) $D = D' \bigcup U_j$ (where U_j is an arbitrary user u_j's user log) in Sec. 3.2.1 and 3.2.2. Two complementary scenarios of bounding the difference between $Pr[\mathcal{A}(D) \in S]$ and $Pr[\mathcal{A}(D') \in S]$ per Definition 2 will be discussed respectively for Case (1), where S is an arbitrary set of possible outputs. In Sec. 3.2.3, we discuss the extension to Case (2) $D' = D \bigcup U_j$, and Case (3) U_j is different in D and D'.

3.2.1 $Pr[\mathcal{A}(D') \in S] = 0$ in Case (1) $D = D' \bigcup U_j$

While running multinomial sampling with inputs D and D' respectively, if all the outputs in S includes the differential user-ID u_j, then we have $Pr[\mathcal{A}(D') \in S] = 0$, because: D' does not have u_j, and if sampling with the histogram in input D', the output would never include user-ID u_j. Per Definition 2, $Pr[\mathcal{A}(D) \in S] \leq \delta$ should hold. Equivalently, $\max\{Pr[\mathcal{A}(D) \in S]\}$ should be bounded by δ. In fact, the maximum $Pr[\mathcal{A}(D) \in S]$ occurs when S is the set of all the possible outputs including u_j. Thus, we have

$$\begin{aligned} & \max\{Pr[\mathcal{A}(D) \in S]\} \\ =& Pr[\mathcal{A}(D) \in \text{"All outputs including } u_j\text{"}] \\ =& Pr[\mathcal{A}(D) \text{ includes at least one } u_j] \end{aligned} \tag{3}$$

If sampling with the histograms in input D, Equation 3 equals the probability that "*u_j is sampled at least once in the multinomial sampling process of all the distinct query-url pairs in U_j*". For every query-url pair $\phi_i \in U_j$, if its total output count in the sampling is x_i, the probability that



Input Search Log

User-ID u_j	query-url pair Φ_i	Count c_{ij}
081	pregancy test nyc, medicinenet.com	2
	book, amazon.com	3
	google, google.com	15
082	car price, kbb.com	2
	google, google.com	7
083	google, google.com	17
	diabetes medecine, walmart.com	1
	book, amazon.com	1
	car price, kbb.com	5

Compute the *optimal output counts* of all the query-url pairs $x=(x_1,.., x_n)$
(Sampling is differentially private if x satisfies the conditions in Theorem 2)

Multinomial Sampling

query-url pair Φ_i	Optimal Output Count x_i	Sampled User IDs (sampled times)
~~pregancy test nyc, medicinenet.com~~	~~0~~	
book, amazon.com	3	2→081 (2), 083 (1)
google, google.com	20	20→081 (8), 082 (3), 083 (9)
~~diabetes medecine, walmart.com~~	~~0~~	
car price, kbb.com	4	3→082 (1), 083 (3)

(Maximize the output utility with a defined utility measure for the *output counts* of all the query-url pairs)

Output Search Log

User ID u_j	query-url pair Φ_i	Count x_{ij}
081	~~pregancy test nyc, medicinenet.com~~	~~0~~
	book, amazon.com	2
	google, google.com	8
082	car price, kbb.com	1
	google, google.com	3
083	google, google.com	9
	~~diabetes medecine, walmart.com~~	~~0~~
	book, amazon.com	1
	car price, kbb.com	3

(a) Sanitization with Multinomial Sampling (b) A Sample Output

Fig. 1. An Example of the Multinomial Sampling in the Sanitization

u_j is not sampled in any single multinomial trial is $\frac{c_i - c_{ij}}{c_i}$ because user u_j holds ϕ_i with the count c_{ij} and the total count of ϕ_i in D is c_i. Since $\forall \phi_i \in U_j$ may lead to that u_j being sampled, and the multinomial sampling for every query-url pair ϕ_i includes x_i independent trials, we have $Pr[u_j \text{ is not sampled}] = \prod_{i=1}^{n}(\frac{c_i - c_{ij}}{c_i})^{x_i}$. Finally, we can obtain the probability that u_j is sampled at least once: $Pr[u_j \text{ is sampled}] = 1 - \prod_{i=1}^{n}(\frac{c_i - c_{ij}}{c_i})^{x_i}$. Thus, we have:

$$\max\{Pr[\mathcal{A}(D) \in S]\} = 1 - \prod_{i=1}^{n}\left(\frac{c_i - c_{ij}}{c_i}\right)^{x_i} \tag{4}$$

Note that for any query-url pair $\phi_i \in U_j$ where $c_{ij} = c_i$ (ϕ_i is uniquely held by user u_j, e.g., user 083's sensitive query-url pair "diabetes medicine, walmart.com"), if its output count $x_i > 0$, the probability $\max\{Pr[\mathcal{A}(D) \in S]\} = 1$, which cannot be bounded by δ. Therefore, $x_i = 0$ must hold for this scenario and such unique query-url pair should be suppressed.

3.2.2 $Pr[\mathcal{A}(D') \in S] > 0$ *in Case (1)* $D = D' \bigcup U_j$

If existing an output in S without user-ID u_j, both $Pr[\mathcal{A}(D) \in S]$ and $Pr[\mathcal{A}(D') \in S]$ will be positive. At this time, we should bound $\frac{Pr[\mathcal{A}(D') \in S]}{Pr[\mathcal{A}(D) \in S]}$ and $\frac{Pr[\mathcal{A}(D) \in S]}{Pr[\mathcal{A}(D') \in S]}$ with e^ϵ.

Now we divide the arbitrary output set S into S_1 and S_2 where $u_j \in S_1$ and $u_j \notin S_2$, and denote any arbitrary output in S_1 and S_2 as O_1 and O_2 respectively. We can derive a sufficient condition to bound $\frac{Pr[\mathcal{A}(D') \in S]}{Pr[\mathcal{A}(D) \in S]}$ and $\frac{Pr[\mathcal{A}(D) \in S]}{Pr[\mathcal{A}(D') \in S]}$ (Gotz et al. [13] also studied this):

Theorem 1: If $\forall O_2 \in S_2, \frac{Pr[\mathcal{A}(D')=O_2]}{Pr[\mathcal{A}(D)=O_2]} \leq e^\epsilon$ hold, then $\frac{Pr[\mathcal{A}(D') \in S]}{Pr[\mathcal{A}(D) \in S]} \leq e^\epsilon$ also holds.

Proof: Note that $\forall O_1 \in S_1, Pr[\mathcal{A}(D') = O_1] = 0$, then

$$\begin{aligned} Pr[\mathcal{A}(D') \in S] = &\int_{\forall O_1 \in S_1} Pr[\mathcal{A}(D') = O_1] dO_1 \\ + &\int_{\forall O_2 \in S_2} Pr[\mathcal{A}(D') = O_2] dO_2 \\ \leq e^\epsilon &\int_{\forall O_2 \in S_2} Pr[\mathcal{A}(D) = O_2] dO_2 \\ \leq e^\epsilon &Pr[\mathcal{A}(D) \in S_2] \leq e^\epsilon Pr[\mathcal{A}(D) \in S] \end{aligned}$$

This completes the proof. □

Similarly we can prove that $Pr[\mathcal{A}(D) \in S] \leq \delta + e^\epsilon Pr[\mathcal{A}(D') \in S]$. This shows that we can ensuring differential privacy by letting $\forall O_2 \in S_2, e^{-\epsilon} \leq \frac{Pr[\mathcal{A}(D)=O_2]}{Pr[\mathcal{A}(D')=O_2]} \leq e^\epsilon$ in multinomial sampling, detailed as below.

For all query-url pairs in both U_j and D', sampling user-IDs from D involves an additional user-ID u_j (but $u_j \notin O_2$) compared to sampling user-IDs from D'. We then have $\frac{Pr[\mathcal{A}(D)=O_2]}{Pr[\mathcal{A}(D')=O_2]} \leq 1 \leq \frac{Pr[\mathcal{A}(D')=O_2]}{Pr[\mathcal{A}(D)=O_2]}$. Since the ratio $\frac{Pr[\mathcal{A}(D)=O_2]}{Pr[\mathcal{A}(D')=O_2]}$ is bounded by 1 (and e^ϵ), we only need to drive a bound for the ratio $\frac{Pr[\mathcal{A}(D')=O_2]}{Pr[\mathcal{A}(D)=O_2]}$. Since the conducted sampling for all query-url pairs $\phi_i \in Q$ is independent, we denote ϕ_i's share in ratio $\frac{Pr[\mathcal{A}(D')=O_2]}{Pr[\mathcal{A}(D)=O_2]}$ as r_i (note that $\frac{Pr[\mathcal{A}(D')=O_2]}{Pr[\mathcal{A}(D)=O_2]} = \prod_{\forall \phi_i \in Q} r_i$).

As mentioned in Sec. 3.2.1, all the query-url pairs in D but not in D' (unique query-url pairs in U_j) should be removed in O_2, where $x_i = 0$ makes $r_i = 1$. Thus, to sample O_2 from D, we only sample user-IDs for the common query-url pairs in D and D'. Two categories of them can be identified:

- $\forall \phi_i$ in $D' \setminus U_j$ (e.g., D' is 082 and 083's user logs, $u_j = 081$, ϕ_i="car price, kbb.com"), the probabilities of sampling user-IDs for ϕ_i from D and D' are equivalent because these query-url pairs' query-url-user histograms in D and D' are identical. Now we have $r_i = 1$.
- $\forall \phi_i$ in $D' \bigcap U_j$ (e.g., D' is 082 and 083's user logs, $u_j = 081$, ϕ_i="book, amazon.com"), we can consider every sampled user-ID in the process of $\mathcal{A}(D)$ into two occasions: (a) "*u_j is sampled*" and (b) "*u_j is not sampled*". In every multinomial trial for ϕ_i, the probability of sampling u_j is $\frac{c_{ij}}{c_i}$ while the probability of sampling another user-ID in D (any user-ID in D') is $1 - \frac{c_{ij}}{c_i}$. Since we run x_i times independent multinomial trials for ϕ_i, we have r_i= $1/(1 - \frac{c_{ij}}{c_i})^{x_i} = (\frac{c_i}{c_i - c_{ij}})^{x_i}$ (since O_2 does not contain user-ID u_j, u_j should not be sampled in all x_i independent trials while generating O_2 from D).

To generate any output $O_2 \in S_2$ from D and D' respectively, it is independent to sample user-IDs for the above two categories of query-url pairs. Thus, $\forall O_2 \in S_2$, $\frac{Pr[\mathcal{A}(D')=O_2]}{Pr[\mathcal{A}(D)=O_2]} = \prod_{\forall \phi_i \in Q} r_i$. Note that $\forall \phi_i \in D' \setminus U_j, r_i = 1$, then we have for all $O \in \Omega_2$:

$$\frac{Pr[\mathcal{A}(D') = O_2]}{Pr[\mathcal{A}(D) = O_2]} = \prod_{\forall \phi_i \in Q} r_i =$$

$$\left(\prod_{\forall \phi_i \in U_j \setminus D'} r_i\right) * \left(\prod_{\forall \phi_i \in D' \cap U_j} r_i\right) * \left(\prod_{\forall \phi_i \in D' \setminus U_j} r_i\right)$$

$$= \prod_{\forall \phi_i \in D' \cap U_j} \left(\frac{c_i}{c_i - c_{ij}}\right)^{x_i} = \prod_{i=1}^{n} \left(\frac{c_i}{c_i - c_{ij}}\right)^{x_i} \tag{5}$$

Thus, bounding Equation 5 by e^ϵ is effective to guarantee differential privacy for this scenario.

3.2.3 Case (2) and (3) of D and D'

We now discuss the extension from Case (1) $D = D' \bigcup U_j$ to Case (2) and (3) for D and D'.

First, in Case (2) $D' = D \bigcup U_j$, similar to Sec. 3.2.1 and 3.2.2, we need to examine two scenarios $Pr[\mathcal{A}(D) \in S] = 0$ and $Pr[\mathcal{A}(D) \in S] > 0$ since D does not include user-ID u_j. Conducting similar analysis as well as Case (1) $D = D' \bigcup U_j$ (by swapping D and D'), we can discover that bounding the probability differences in Equation 4 and 5 (derived from D') with δ and e^ϵ respectively could make the multinomial sampling based on this pair of D and D' differentially private.

Second, in Case (3), user u_j has different user logs U_j in D and D'. We now show that multinomial sampling based on this pair of D and D' could also satisfy differential privacy. Specifically, in this case, both D and D' include u_j, thus we have $Pr[\mathcal{A}(D) \in S] > 0$ and $Pr[\mathcal{A}(D') \in S] > 0$, and we should derive the upper bound for the multiplicative differences $\frac{Pr[\mathcal{A}(D)\in S]}{Pr[\mathcal{A}(D')\in S]}$ and $\frac{Pr[\mathcal{A}(D')\in S]}{Pr[\mathcal{A}(D)\in S]}$. To do so, we can conduct similar analysis as Sec. 3.2.2. Letting U_j and U'_j denote user u_j's user log in D and D' respectively, the count of query-url pair ϕ_i in U_j and U'_j is defined as c_{ij} and c'_{ij} respectively.

- If $c_{ij} = c'_{ij}$, the probabilities of sampling any user-ID from D and D' in any multinomial trial are *identical*, thus ϕ_i's share in ratio $\frac{Pr[\mathcal{A}(D')\in S]}{Pr[\mathcal{A}(D)\in S]}$ always equals 1.
- If $c_{ij} > c'_{ij} > 0$, the probabilities of sampling any user-ID from D and D' in any multinomial trial are *closer than that of Case (1) $D = D' \bigcup U_j$* since $c'_{ij} = 0$ in Case (1) and counts histograms of D and D' are closer in Case (3) than Case (1).
- Similarly, if $0 < c_{ij} < c'_{ij}$, the probabilities of sampling any user-ID from D and D' in any multinomial trial are *closer than that of Case (2) $D' = D \bigcup U_j$* since $c_{ij} = 0$ in Case (2) and counts histograms of D and D' are closer in Case (3) than Case (2).
- If $c'_{ij} = 0$, the probabilities of sampling any user-ID from D and D' in any multinomial trial are the *same as Case (1) $D = D' \bigcup U_j$* since $c'_{ij} = 0$ in both Case (1) and (3) and the counts.
- If $c_{ij} = 0$, the probabilities of sampling any user-ID from D and D' in any multinomial trial are the *same as Case (2) $D' = D \bigcup U_j$* since $c_{ij} = 0$ in both Case (2) and (3).

For simplicity of notation, we denote $\frac{Pr[\mathcal{A}(D)\in S]}{Pr[\mathcal{A}(D')\in S]}$ in Case (1), (2) and (3) as R_1, R_2 and R_3 respectively, thus we have $R_1, R_2 \in [e^{-\epsilon}, e^\epsilon]$. Since sampling user-IDs for any query-url pair is independent, the overall multiplicative difference of the probabilities is the *product of all multiplicative differences* for all query-url pairs (which may fall into any of the above five cases). Therefore, by converting the multiplicative probability difference representation using Theorem 1, we have $R_3 \leq R_1 * R_2 \leq e^{2\epsilon}$ and similarly we can also derive $\frac{1}{R_3} \leq e^{2\epsilon}$.

In summary, the multinomial sampling satisfies $(2\epsilon, \delta)$-differential privacy in Case (3) if we bound Equation 4 and 5 (for both D and D') with δ and e^ϵ respectively.

3.2.4 Differentially Private Sampling

Following the above analysis, we can derive a set of sufficient conditions that the output counts x satisfy in order for (ϵ, δ)-differential privacy in the multinomial sampling.

Theorem 2: Multinomial sampling $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy if for any input search log D, the output counts of query-url pairs $x = (x_1, \dots, x_n)$ meet the following conditions:

1) for any ϕ_i posed by user u_j only, let $x_i = 0$;
2) $\forall j \in [1, m], \prod_{i=1}^{n} (\frac{c_i}{c_i - c_{ij}})^{x_i} \leq e^{\epsilon/2}$;
3) $\forall j \in [1, m], 1 - \prod_{i=1}^{n} (\frac{c_i - c_{ij}}{c_i})^{x_i} \leq \delta$.

Proof: With the analysis in Sec. 3.2.1, 3.2.2 and 3.2.3, it is straightforward to prove this theorem. □

Note that if $\exists c_{ij} = c_i$, unique query-url pair ϕ_i will be suppressed in the output (Condition 1). If $\exists c_{ij}$ which is close to c_i, the output count x_i will be also enforced to 0 while satisfying Condition 2 and 3 in the optimization of output utility, thus we can still use effective ϵ and δ (sufficiently small) to ensure differential privacy for this special case.

Since Condition 2 and 3 have given two sets of constraints for the output counts of all query-url pairs $x = (x_1, \dots, x_n)$, we can compute the optimal x for differentially private multinomial sampling by solving the following problem (minimizing the utility loss in Equation 2):

$$\min: \quad \sum_{i=1}^{n} \left[\frac{c_i}{|D|} \log(\frac{c_i}{|D|} \cdot \frac{|O| + n}{x_i + 1})\right]$$
$$s.t. \begin{cases} \forall j \in [1, m], \prod_{i=1}^{n} (\frac{c_i}{c_i - c_{ij}})^{x_i} \leq e^{\epsilon/2} \\ \forall j \in [1, m], 1 - \prod_{i=1}^{n} (\frac{c_i - c_{ij}}{c_i})^{x_i} \leq \delta \\ \forall x_i \geq 0 \text{ and } x_i \text{ is an integer} \end{cases}$$

This is a nonlinear programming (NLP) problem with *linear constraints*: for simplicity, we let constant $t_{ij} = \frac{c_i}{c_i - c_{ij}}$, and combine the righthand-side constants of each user log U_j's two constraints as $b = \min\{\epsilon/2, \log \frac{1}{1-\delta}\}$. Then, we have

$$\min: \quad \sum_{i=1}^{n} \left[\frac{c_i}{|D|} \log(\frac{c_i}{|D|} \cdot \frac{|O| + n}{x_i + 1})\right]$$
$$s.t. \begin{cases} \forall j \in [1, m], \sum_{i=1}^{n} (x_i \cdot \log t_{ij}) \leq b \\ \forall x_i \geq 0 \text{ and } x_i \text{ is an integer} \end{cases} \tag{6}$$

where $|D|$ and $|O|$ represent the total count of all query-url pairs in D and O respectively, and x_i represents the overall count of query-url pair ϕ_i in any sampled output. We can solve this NLP problem by linear approximation [45]: approximating the nonlinear objective function (proved as convex in the Appendix) by piecewise linear functions, then the optimal solution of this NLP problem can be derived after solving an LP problem. The details are presented in Appendix B.

3.3 Differential Privacy Guarantee before Sampling

Besides the differential privacy guarantee for multinomial sampling shown above, we also ensure that the process of computing the optimal counts $x = (x_1, \dots, x_n)$ satisfies differential privacy (such process occurs before sampling). One simple way to do this is to use the generic way of adding Laplacian noise to the counts derived from the optimization $(x_1, \dots, x_n)$.

Similar to Korolova et al. [24], if the count differences of every query-url pair $\phi_i \in Q$ in the optimal solutions derived from two neighboring inputs (D, D') are bounded by a constant d, computing optimal counts can be guaranteed to be ϵ'-differentially private [24] (ϵ' is the parameter of ensuring differential privacy for such step) by adding Laplacian noise $Lap(d/\epsilon')$ to the optimal count of every query-url pair: $\forall i \in [1, n], x_i \leftarrow x_i + Lap(d/\epsilon')$. Indeed, given d, we can simply bound the difference of every query-url pair's optimal count computed from any two neighboring inputs with a preprocessing procedure by examining every user log U_j in the input D (for details, please refer Appendix C). Note that $Lap(d/\epsilon')$ has mean 0 and insignificant standard deviation $\frac{\sqrt{2}d}{\epsilon'}$ with sensitivity d, and this is the minor price of guaranteeing complete differential privacy. Since adding Laplacian noise is a well-studied generic approach, we do not discuss this privacy guarantee here, and the sanitization mechanism refers to the sampling in this paper.

4 Sanitization Model

In this section, we present our sampling based sanitization model that boosts the output utility. Specifically, the model enables r different parties $P_1, \dots, P_r$ to jointly generate an output with their own private search logs. We consider a common scenario in practice that search logs are *horizontally partitioned* to r different shares, assuming that every user's all search tuples are completely held by one party (viz. sets of users held by different parties are disjoint). In this case, if different parties have different sets of unique query-url pairs, the overall query-url pair universe can be securely obtained by any secure union protocol (e.g., Algo. 3). Table 2 gives an example: P_1, P_2 and P_3 hold their own private user logs.

4.1 Collaborative Search Log Sanitization (CELS)

Jointly generating the output with private search logs by r different parties needs more privacy/security consideration:

Definition 4 (CELS): Given search log D horizontally partitioned to r shares $D_1, \dots, D_r$ (held by r parties $P_1, \dots, P_r$), **C**ollaborative s**E**arch **L**og **S**anitization (CELS) is to efficiently generate a probabilistic output O such that (1) the output production satisfies differential privacy, (2) the utility of the output O is maximized under the sampling mechanism, and (3) each party cannot learn any private information from each other in the whole process.

TABLE 2
Horizontally Partitioned Search Logs

		Query-url Pairs (Sensitive Items)					
parties	users	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6
P_1	u_1^1	3	2	0	1	0	0
	u_2^1	1	1	0	0	4	0
P_2	u_1^2	2	1	0	1	0	2
	u_2^2	0	1	0	0	1	1
	u_3^2	1	0	7	0	0	5
P_3	u_1^3	0	0	1	1	0	0
	u_2^3	0	1	0	0	2	0
	u_3^3	1	0	5	0	0	1
	Total (c_i)	8	6	13	3	7	9

We develop a secure communication protocol under Secure Multiparty Computation [11], [47] for sanitization (denoted as *CELS protocol*), ensuring that CELS protocol satisfies differential privacy, and every party cannot learn any private information from each other in the protocol. We assume *semi-honest model* where all parties are honest to follow the protocol but curious to infer information from other parties.

Note that the output utility of CELS protocol can be considerably boosted, compared to the union of all the local output utility derived from each party's own sampling-based sanitization (while ensuring the same level of differential privacy). This is justified in our experiments (Sec. 6).

4.2 Protocol

While sanitizing input D, an NLP problem should be solved to maximize the output utility and the sampling is based on the optimal solution. Therefore, in CELS, a global NLP problem should be securely formulated and solved among them. With the optimal solution of the global NLP problem, all parties should jointly generate the probabilistic output with limited disclosure. We then illustrate the CELS protocol as below.

4.2.1 Secure NLP

In CELS, inputs $D_1, \dots, D_r$ is privately held by parties $P_1, \dots, P_r$ respectively. Letting m_k be the number of users in D_k, if we mathematically formulate the NLP problem Equation 6 among r parties, its constraints are also *horizontally partitioned*: every party $P_k (1 \le k \le r)$ holds its own set of linear inequality constraints – each of them is derived from a user log in its input D_k. In addition, all parties know the set of variables x (the output counts of all query-url pairs) and objective function (the minimum utility loss). Table 3 gives the partitioned shares of the NLP problem, where constant $b = \min\{\epsilon/2, \log \frac{1}{1-\delta}\}$, $t_{ij}^k = \frac{c_i}{c_i - c_{ij}^k}$ and $1 \le i \le n, 1 \le j \le m_k, 1 \le k \le r$.

(1) Secure Counts Sum. since formulating the privately held constraints in the NLP problem requires the total count of every query-url pair in D, we first *securely sum* the global count of every query-url pair over r parties using Homomorphic Encryption [35], [37]. We assume that $P_1, \dots, P_r$ can learn the total count of every query-url pair $(c_1, \dots, c_n)$ in our sanitization (but the individual counts are unknown to each other). Then they can remove the unique query-url pairs after

TABLE 3
Partitioned NLP Problem

All r Parties Know	min : $\sum_{i=1}^{n} \boxed{\frac{c_i}{\lvert D\rvert}\log(\frac{c_i}{\lvert D\rvert}\cdot\frac{\lvert O\rvert+n}{x_i+1})}$
P_1 Holds	$\forall j \in [1, m_1], \sum_{i=1}^{n}(x_i \log t_{ij}^1) \le b$
P_2 Holds	$\forall j \in [1, m_2], \sum_{i=1}^{n}(x_i \log t_{ij}^2) \le b$
$\vdots$	$\vdots$
P_r Holds	$\forall j \in [1, m_r], \sum_{i=1}^{n}(x_i \log t_{ij}^r) \le b$

obtaining the total count of every query-url pair (for satisfying Condition 1 in Theorem 2), and preprocess its own input for ensuring differential privacy for the step of computing the optimal output counts x if necessary.

More specifically, every party P_k ($1 \le k \le r$) holds a query-url pair count vector $\overrightarrow{v_k} = (c_1^k, \dots, c_n^k)$. An arbitrary party is chosen to generate a pair of public-private key (pk, sk) (w.l.o.g., P_1 will do so). P_1 then sends the public key pk to all the remaining parties $P_2, \dots, P_r$. Every party P_k encrypts their count vector $\overrightarrow{v_k}$ using pk: $\overrightarrow{v_k}' = Enc_{pk}(\overrightarrow{v_k})$. The sum of $\overrightarrow{v} = \sum_{k=1}^{r}\overrightarrow{v_k}$ can be guaranteed by the homomorphic property of the cipher-texts (as shown in Algo. 1).

Algorithm 1 Secure Counts Sum

```
Input: $\overrightarrow{v_k}$ held by $P_k$ $(1 \le k \le r)$
Output: $\overrightarrow{v} = \sum_{k=1}^{r}\overrightarrow{v_k}$
1: $P_1$ generates a pair of public-private key $(pk, sk)$
2: $P_1$ sends the public key $pk$ to $P_2, \dots, P_r$
3: for $k = 1, \dots, r$ do
4:     $P_k$ encrypts $\prod_{s=1}^{k}\overrightarrow{v_s}'$ as $\prod_{s=1}^{k}\overrightarrow{v_s}' = \prod_{s=1}^{k-1}\overrightarrow{v_s}' * Enc_{pk}(\overrightarrow{v_k})$
       using $pk$ (note that $*$ and $\prod_{s=1}^{k-1}$ stands for the product of the
       $ith$ entry in the encrypted vectors where $1 \le i \le n$)
5:     $P_k$ sends $\prod_{s=1}^{k}\overrightarrow{v_s}'$ to the next party $P_{k+1}$ (if $k = r$, the next
       party is $P_1$)
6: $P_1$ decrypts $\prod_{k=1}^{r}\overrightarrow{v_k}'$ with the private key $sk$ to obtain $\sum_{k=1}^{r}\overrightarrow{v_k}$
   and distributes the sum to $P_2, \dots, P_r$
```

(2) Secure Linear Constraints Transformation and Variables Permutation. We propose a transformation approach for the horizontally partitioned linear constraints with strong security guarantee by extending the work of Mangasarian [31] and Li et al. [29] (they have not implemented cryptography-based strong security and variables permutation).

First, we let each party $P_k (1 \le k \le r)$ convert its private linear *inequality constraints* into linear *equality constraints* (the standard form): for party P_k's inequality constraint derived from its jth user log U_j^k ($1 \le j \le m_k$), $\sum_{i=1}^{n}(x_i \log t_{ij}^k) \le b$, we add a slack variable y_j^k with a random positive coefficient [2] f_j^k:

$$s.t.\begin{cases}\forall j \in [1, m_1], \sum_{i=1}^{n}(x_i \log t_{ij}^1) + f_j^1 y_j^1 = b\\ \forall j \in [1, m_2], \sum_{i=1}^{n}(x_i \log t_{ij}^2) + f_j^2 y_j^2 = b\\ \vdots \quad \vdots\\ \forall j \in [1, m_r], \sum_{i=1}^{n}(x_i \log t_{ij}^r) + f_j^r y_j^r = b\end{cases} \tag{7}$$

2. Using coefficient f_j^k for slack variable y_j^k is equivalent as simply adding the slack variable with coefficient 1, and f_j^k can prevent learning the transformation matrix for horizontally partitioned linear constraints [29].

The above standard form of all the linear constraints can be written as $Tx + Fy = B$ where constraint matrix $T = \begin{bmatrix} T_1 \\ \vdots \\ T_r \end{bmatrix}$ (note that $T, T_1, \dots, T_r$ are $m \times n$, $m_1 \times n$, $\dots$, $m_r \times n$ matrices with constant entries computed by $\log t_{ij}^k$) and diagonal matrix $F = diag[F_1, F_2, \dots, F_r]$, where $F_1, \dots, F_r$ are diagonal matrices with positive entries $\{f_1^1, \dots, f_{m_1}^1\}, \dots, \{f_1^r, \dots, f_{m_r}^r\}$ in the diagonals respectively.

Second, each party securely pre-multiplies a random matrix to its matrices/vectors in the constraints and implements an index permutation to all the variables $x = (x_1, \dots, x_n)$.

More specifically, the pre-multiplication is $Tx + Fy = B \iff ATx + AFy = AB$, where A is vertically partitioned as $A = [A_1\ A_2\ \dots\ A_r]$ and $A_1, \dots, A_r$ are random matrices generated by parties $P_1, \dots, P_r$ respectively (B is horizontally partitioned as $B = \begin{bmatrix} B_1 \\ \vdots \\ B_r \end{bmatrix}$, and $1 \le k \le r, B_k$ is a vector with m_k entries with the value b).

Note that Homomorphic Cryptosystem [37] facilitates each party to securely pre-multiply its random matrix to its share of the constraint matrix. See the pre-multiplication as below:

$$\begin{array}{ll} P_1: & A_1T_1x + A_1F_1y = A_1B_1 \\ \vdots & \vdots \\ P_r: & A_rT_rx + A_rF_ry = A_rB_r \\ \Longrightarrow & \sum_{k=1}^{r} A_k(T_kx + F_ky) = \sum_{k=1}^{r} A_kB_k \end{array} \tag{8}$$

where the optimal solution remains original [31].

Then, we can let every party $P_k (1 \le k \le r)$ permute the columns of the transformed constraint matrix AT (via permuting the cipher-texts) with its own permutation (or considered as permuting the variables of the NLP problem). With this, no party can learn the true matrix column/variables index and the transformation for other parties' share.

(3) Algorithm. we present the detailed steps of securely solving the NLP problem for CELS protocol in Algo. 2. We can employ an external party, the output data recipient or the cloud P_0 to solve the transformed NLP problem, and let it generate the public-privacy key pair (pk, sk) and send the public key pk to the data holders $P_1, \dots, P_r$ for encryption.

First, every party encrypts all the required scalar products in the matrix multiplication A_kT_k, A_kF_k, A_kB_k using pk (Line 3-7). Second, all parties jointly computes the cipher-texts of $\sum_{k=1}^{r} A_kT_k$, $\sum_{k=1}^{r} A_kF_k$ and $\sum_{k=1}^{r} A_kB_k$ with homomorphic property (Line 8). Note that the query-url count vector $\overrightarrow{v}$ should be applied with the same permutation as matrix AT since the coefficients for variables $(x_1, \dots, x_n)$ in the objective function are derived from $\overrightarrow{v} = (c_1, \dots, c_n)$, where $\overrightarrow{v}$ should be encrypted before permutation. Third, every party $P_k (k = 1, \dots, r)$ applies its column/index permutation π_k to the cipher-texts $Enc_{pk}(\overrightarrow{v})$ and $Enc_{pk}[AT]$ respectively (Line 10-11), and then they send all the cipher-texts to P_0. After receiving the cipher-texts from $P_1, \dots, P_r$, P_0 decrypts the ciphertexts with its private key sk, and solves the transformed NLP (Line 12-13) with the approach shown in Appendix B.

Algorithm 2 Secure NLP

```
Input: Horizontally partitioned constraint matrix $T_k$ held by $P_k$ $(1 \leq k \leq r)$, $P_0$ is an external party, data recipient or cloud
Output: Optimal Solution $x = (x_1, \dots, x_n)$
    {All parties agree on a large integer value $\ell \geq m$}
 1: every party $P_k$ $(1 \leq k \leq r)$ generates an $\ell \times m_k$ random matrix $A_k$
 2: $P_0$ generates a pair of public-private key $(pk, sk)$ and sends $pk$ to $P_1, \dots, P_r$
    {Homomorphic Encryption: a random nonce is chosen for each encryption}
 3: for $k = 1, \dots, r$ do
 4:     $P_k$ encrypts all the entries in $A_k, T_k, F_k$ with $pk$: $Enc_{pk}(A_k) = A'_k$, $Enc_{pk}(T_k) = T'_k$, and $Enc_{pk}(F_k) = F'_k$
 5:     for each row $s \in [1, \ell]$ of $A'_k$ and each column $i \in [1, n]$ of $T'_k$ do
 6:         $P_k$ computes $Enc_{pk}[A_k T_k]_{si} = \prod_{j=1}^{m_k} [A'_k]_{sj}^{(T_k)'_{ji}}$
 7:     $P_k$ computes $Enc_{pk}[A_k F_k]$ and $Enc_{pk}[A_k B_k]$ like Line 5-6
 8: $P_1, \dots, P_r$ jointly computes $Enc_{pk}[AT] = \prod_{k=1}^{r} Enc_{pk}[A_k T_k]$, $Enc_{pk}[AF] = \prod_{k=1}^{r} Enc_{pk}[A_k F_k]$, and $Enc_{pk}[AB] = \prod_{k=1}^{r} Enc_{pk}[A_k B_k]$
 9: an arbitrary party encrypts the query-url pair count vector $\overrightarrow{v}$ with $pk$: $Enc_{pk}[\overrightarrow{v}]$
    {Line 10-11: Variables Permutation}
10: for $k = 1, \dots, r$ do
11:     $P_k$ applies index permutation $\pi_k$ to $Enc_{pk}[AT]$ and the encrypted count vector $Enc_{pk}[\overrightarrow{v}]$, and sends them to the next party.
    {the last party sends $Enc_{pk}[\pi_r(\dots\pi_1(AT)\dots)]$, $Enc_{pk}[AB]$, $Enc_{pk}[AF]$ and $Enc_{pk}[\pi_r(\dots\pi_1(\overrightarrow{v})\dots)]$ to $P_0$}
12: $P_0$ decrypts $Enc_{pk}[\pi_r(\dots\pi_1(AT)\dots)]$, $Enc_{pk}[AF]$, $Enc_{pk}[AB]$ and $Enc_{pk}[\pi_r(\dots\pi_1(\overrightarrow{v})\dots)]$ with $sk$ to obtain $\pi_r(\dots\pi_1(AT)\dots)$, $AF$, $AB$ and $\pi_r(\dots\pi_1(\overrightarrow{v})\dots)$
13: $P_0$ solves the NLP problem with linear constraints $\pi_r(\dots\pi_1(AT)\dots)x + AFy = AB$, global constraint $\sum_{i=1}^{n} x_i = |O|$ and the query-url pair count vector $\pi_r(\dots\pi_1(\overrightarrow{v})\dots)$ in the permuted objective function (Solving process is given in Appendix B): the permuted optimal solution $\pi_r(\dots\pi_1(x)\dots)$ is obtained
    {All parties get the optimal solution in true index by applying the inverse permutations $\pi_k^{-1}(k = r, \dots, 1)$ to $\pi_r(\dots\pi_1(x)\dots)$ in order}
```

Finally, every party applies their inverse permutations $k = r, \dots, 1$, π_k^{-1} to the permuted optimal solution $\pi_r($ ... $\pi_1(x)\dots)$ in order, and acquire $x = (x_1, \dots, x_n)$ with the true index.

4.2.2 Secure Sampling

We now discuss how to securely sample the output by all parties with the optimal solution of the NLP problem.

(1) Global and Local Sampling. In the sampling based sanitization, we denote "global sampling" as sampling the output with the global input database $D = \bigcup_{k=1}^{r} D_k$ and the optimal output count $x = (x_1, \dots, x_n)$. Moreover, we denote "local sampling" as – every party P_k locally samples an output O_k with its privately held input D_k and a share of the global optimal output counts $x = (x_1, \dots, x_n)$: i.e. P_k locally runs $x_i \cdot \frac{c_i^k}{c_i}$ times multinomial trials for query-url pair ϕ_i.

In essence, since the expectation of sampling every user-ID for every query-url pair is identical for both global and local sampling, the output of the global sampling is equivalent to the union of all the outputs of the local sampling.

(2) Secure Union. After obtaining the outputs with local sampling, we can utilize a commutative encryption based protocol to securely union all the local outputs $O_1, \dots, O_r$. Since the commutative encryption-based protocol (e.g., Pohlig-Hellman's encryption scheme [41]) generates the same cipher-text by encrypting the plain-text with multiple encryption keys in any arbitrary order, we can let every party first encrypt its local data to cipher-text and then encrypt all the cipher-texts by all parties (note that an arbitrary party should be picked as the coordinator of this protocol. W.l.o.g., we let P_1 be the coordinator which cannot learn more information than other parties). Finally, all parties decrypt all the cipher-texts to get the union of the plain-texts. Algo. 3 describes the details.

Algorithm 3 Secure Union

```
Input: Sampled outputs $O_1, \dots, O_r$ held by $r$ parties $P_1, \dots, P_r$, Commutative encryption and decryption keys of $P_k$: $(e_k, s_k)$
Output: the union $O = \bigcup_{k=1}^{r} O_k$
 1: every party $P_k (1 \leq k \leq r)$ encrypts $O_k$: $O'_k = Enc_{e_k}(O_k)$ and sends $O'_k$ to $P_1$
    {At Party $P_1$ (Coordinator)}
 2: $O \leftarrow \{\}$, $O' \leftarrow \{\}$
 3: for each $O'_k (1 \leq k \leq r)$ do
 4:     $temp \leftarrow O'_k$
 5:     for $i = 1, \dots, r, i \neq k$ (parties besides $P_k$) do
 6:         send $temp$ to $P_i$ for commutative encryption with key $e_i$: $temp \leftarrow Enc_{e_i}(temp)$ ($P_i$ sends $temp$ back to $P_1$)
 7:     At $P_1$: $O' \leftarrow O' \bigcup temp$
    {Decryption of $O'$ by all parties, at Party $P_1$ (Coordinator)}
 8: for each $O'_k \in O' (1 \leq k \leq r)$ do
 9:     $temp \leftarrow O'_k$
10:     for $i = 1, \dots, r$ do
11:         send $temp$ to $P_i$ for decryption with $D_i$ ($P_i$ decrypts $temp$: $temp \leftarrow Dec_{s_i}(temp)$, and sends $temp$ back to $P_1$)
12:         $P_1$ receives $temp$ from $P_i$
13:     $O \leftarrow O \bigcup temp$
14: output $O = \bigcup_{k=1}^{r} O_k$
```

4.2.3 CELS Protocol

CELS protocol can be obtained by composing secure sum (Algo. 1), secure NLP (Algo. 2), local sampling, and secure union (Algo. 3).

5 Analysis

In this section, we analyze the CELS protocol security in semi-honest model by quantifying the privacy leakage under SMC [11], [47]. SMC states that a computation is secure if the view of each party during the execution of the protocol can be effectively simulated knowing only its input and output. This is more about protocol security but not quite the same as saying that all private information is protected against leakage. Indeed, differential privacy can complement the protocol security by bounding the privacy risk of inferring information in the sanitized result. We also analyze the complexity of computation and communication of the CELS protocol.

5.1 CELS Protocol Security

5.1.1 Secure Counts Sum

Theorem 3: Algo. 1 privately computes the sum of r query-url pairs count vectors $\overrightarrow{v_k}(1 \leq k \leq r)$, where each party P_k only knows the sum $\overrightarrow{v} = \sum_{k=1}^{r} \overrightarrow{v_k}$.

Proof: For all $k \in [1, r]$, P_k receives the encrypted vector sum from the previous party $Enc_{pk}(\sum_{s=1}^{k} \overrightarrow{v_s}) = \prod_{s=1}^{k} Enc_{pk}(\overrightarrow{v_s})$ (note that $\prod_{s=1}^{k-1}$ stands for the products of all the ith entries in the encrypted vectors where $1 \leq i \leq n$). It is straightforward to simulate the views of all parties by repeating the encryptions in the algorithm step by step. The simulator runs in linear time w.r.t. the size of the input vectors, and thus satisfies the security proof requirement. □

5.1.2 Secure NLP

Most part of Algo. 2 are locally executed by each party.

Theorem 4: In Algo. 2, $P_1, \dots, P_r$ learns only P_0's public key pk and the permuted optimal solution $\pi_r(...\pi_1(x)...)$ while P_0 learns only the transformed matrices/vector $\pi_r(...\pi_1(AT)...)$, AF, AB and $\pi_r(...\pi_1(\overrightarrow{v})...)$.

Proof: $P_1, \dots, P_r$'s view: every party $P_k(1 \leq k \leq r)$ first encrypts the scalar products in its transformed matrix/vector A_kT_k, A_kF_k, and A_kB_k with the public key pk (Line 3-7), and there is no communication occurring in this stage. This can be simulated by running these steps by each party where random nonce are chosen. In addition, Line 8 calls a secure sum subprotocol with the inputs of the distributed matrices/vectors (the messages can be simulated per the proof of Theorem 3). In the stage of permutation (Line 10-11), $P_1, \dots, P_r$ can run an inverse permutation algorithm to the cipher-texts with permutation $\pi_r(...\pi_1(\cdot)...)$ in linear time. Thus, $P_1, \dots, P_r$'s view can be simulated in polynomial time, and they learn only P_0's public key pk and the final output $\pi_r(...\pi_1(x)...)$.

P_0's view: P_0 receives following messages from $P_1, \dots, P_r$: $Enc_{pk}[\pi_r(...\pi_1(AT)...)]$, Enc_{pk} $[AF]$, $Enc_{pk}[AB]$ and $Enc_{pk}[\pi_r(...\pi_1(\overrightarrow{v})...)]$ in only one round communication. P_0 learns $\pi_r(...\pi_1(AT)...)$, AF and vectors Ab and $\pi_r(...\pi_1(\overrightarrow{v})...)$ to solve the transformed NLP problem. Thus, the messages can be simulated by encrypting $\pi_r(...\pi_1(AT)...)$, AF and vectors Ab and $\pi_r(...\pi_1(\overrightarrow{v})...)$ with its own public key pk. This simulator is clearly constructed in linear time. □

Mangasarian [31] has shown that it is nearly impossible to learn matrices A, F and vector B with the known transformed matrices AT, AF and vector AB. Note that even if all the entries in B equal $\min\{\epsilon/2, \log\frac{1}{1-\delta}\}$ which is known to all parties, it is still impossible to reconstruct $A = [A_1\ A_2\ \dots\ A_r]$ with only known Ab and B ($\forall k \in [1, r]$, the entries in A_k and even the sizes of b_k and A_k are unknown to other parties). Besides the matrix multiplication based transformation, we let all parties jointly permute the variables in the optimization problem. Thus, P_0 can only formulate a random NLP problem which reveals nothing about the original problem.

5.1.3 Secure Union

All parties only send and receive cipher-texts in Algo. 3.

Theorem 5: Algo. 3 privately generates the union of the local sampling outputs $O = \bigcup_{k=1}^{r} O_k$.

Proof: The exchanged cipher-texts received by all parties can be simulated by an inverse algorithm of the secure union protocol with the same commutative cryptosystem. Thus, Algo. 3 privately generates the union of the local sampling outputs $O = \bigcup_{k=1}^{r} O_k$ where only the length of $\forall k \in [1, r]$, O_k can be obtained. □

5.1.4 Overall CELS Protocol

Theorem 6: CELS protocol reveals at most $\overrightarrow{v}$ and $\pi_r(...\pi_1(x)...)$ to $P_1, \dots, P_r$, and the transformed matrices/vectors $\pi_r(...\pi_1(AT)...)$, AF, Ab and permuted counts vector $\pi_r(...\pi_1(\overrightarrow{v})...)$ to P_0.

Proof: All the communication in CELS protocol occurs in the calls to one-time secure counts sum (Algo. 1), secure NLP (Algo. 2) and secure union (Algo. 3). Applying the composition theorem [11] can complete the proof. □

Note that $P_1, \dots, P_r$ learns the permuted output counts vector $\pi_r(...\pi_1(x)...)$ in the CELS protocol, and they have to recover the true index for sampling (thus know $x = (x_1, \dots, x_n)$ since then). Hence, some of those parties might guess the overall permutation index from this. However, every party's individual permutation cannot be inferred by those parties in this case, and P_0 does not know the overall permutation. Theorem 6 still holds under the SMC definition, and this minor information disclosure does not hurt any party.

5.2 Differential Privacy

Theorem 7: CELS protocol is (ϵ, δ)-differentially private.

Proof: All the conditions in Theorem 2 are satisfied in CELS from a global point of view (Condition 1 is satisfied after every party knows the total count of every query-url pair and suppresses the unique queries; Condition 2 and 3 are satisfied by subjecting to the linear constraints in the NLP problem), hence CELS protocol satisfies (ϵ, δ)-differential privacy. Note that if differential privacy for computing the optimal output counts with the NLP problem is desirable, every party can locally preprocess its input with known total count and jointly add Laplacian noise to the output count to ensure differential privacy for the step before sampling. □

5.3 Cost Analysis

We now analyze the computation and communication complexity of the CELS protocol.

5.3.1 Computation Cost

NLP Computation Cost. In CELS protocol, the NLP problem with linear constraints is securely transformed by $P_1, \dots, P_r$ and solved by P_0. Such NLP problem is formulated with n variables and m private linear constraints where m linear constraints are securely transformed into ℓ new linear constraints ($\ell \geq m$). As described in Appendix B, P_0 can solve it by linear approximation with K intervals in $[0, |O|]$, and the final LP problem consists of Km variables and $(\ell + 1)$ linear constraints. Standard solvers like Simplex method can find the optimal solution within an ignorable time in the protocol.

TABLE 4
Characteristics of the Data Sets

Datasets	$\|D\|$	m	n	$max\|U_j\|$	$avg\|U_j\|$	$max\|\|U_j\|\|$	$avg\|\|U_j\|\|$
AOL	1,864,860	51,922	1,190,491	6,925	54.7	5609	38.14
MSNBC	4,698,794	989,818	17	14,795	4.75	17	1.72

Encryption and Decryption Cost. In CELS protocol, we analyze the cost in three major subprotocols. Specifically,

- Secure counts sum: every party has to perform a homomorphic encryption on a length-n vector, and $P_2, \dots, P_r$ additionally compute the homomorphic products of the cypher-texts with minor computation cost. Finally, P_1 runs only one time decryption for length-n vector.
- Secure NLP: every party P_k first encrypts the scalar products of $\ell(n+m+1)$ pairs of length-m_k vectors (one time length-n objective vector encryption is also required). The secure sum then executes with $\ell(n+m+1)$ entries amongst r parties. Note that the computation cost of permutation can be ignored compared with encryption and decryption. Finally, P_0 runs one time decryption for $\ell(n+m+1)+n$ entries.
- Secure union: r^2 times commutative encryption and decryption on each party's local sampling output $\forall k \in [1,r]$, O_k (which is a counts matrix with size $\approx m_k \times n$).

Thus, the computational complexity w.r.t. encryption and decryption is $nr + \ell(n+m+1)r * (2\sum_{k=1}^{r} m_k) + n + r^2 * (\sum_{k=1}^{r} m_k)n \approx O(\ell mnr + mnr^2)$ and $n + \ell(n+m+1) + n + r^2 * (\sum_{k=1}^{r} m_k) * n \approx O(r^2 mn)$ respectively.

5.3.2 Communication Cost

- Secure counts sum: r times communication are required among r parties to deliver the cipher-texts of a length-n vector to the next party (P_1 also sends its public key to $P_2, \dots, P_r$). Thus, the total bit cost for this step is $r * n + (r-1) * pk\_size$ bits.
- Secure NLP: The external party (or cloud) P_0 first sends its public key to $P_1, \dots, P_k$. Consequently, the secure sum and variables permutation call $(r-1)$ rounds of communication among $P_1, \dots, P_r$ in total while the outsourcing calls one-time communication between P_r and P_0. Thus, the total bit cost for this step is $r * pk\_size + 2(r-1)(\ell * n + \ell * m + \ell) + (\ell * n + \ell * m + \ell + n) \approx O(r\ell(m+n))$ bits. Notice that the communication cost of inverse permutation on the length-n optimal solution can be neglected due to tiny overheads.
- Secure union: $(r-1)r$ rounds of communication (from the coordinator P_1 to $P_2, \dots, P_r$) are required in the commutative encryption while the decryption also requires $(r-1)r$ rounds of communication. Thus, the total bit cost is $2(r-1)r * \sum_{k=1}^{r} m_k * n \approx O(mnr^2)$ bits.

The overall communication complexity is $O(mnr^2)$.

6 Experimental Results

In this section, we examine the performance of our CELS protocol using real datasets.

6.1 Experiment Setup

Datasets. We conduct experiments on two real datasets – AOL search logs [14] and MSNBC (available at UCI ML Repository). The AOL dataset was originally logged by the AOL search engine in ten partitions. The 2 Gigabytes dataset consists of 20 million web queries collected from about $650k$ users over three months in 2006. The original MSNBC dataset describes the URL categories visited by users in sequential order. Each tuple includes a set of distinct URL categories visited by a user and the visited counts (every URL category can be considered as a query-url pair ϕ_i).

The statistics of two datasets are summarized in Table 4 (one AOL dataset partition and the whole MSNBC dataset). $|D|$ is the total count of query-url pairs in D; m and n denote the number of distinct users and query-url pairs respectively; $max|U_j|$ and $avg|U_j|$ represent the maximum and average value of every user's total count of query-url pairs; $max||U_j||$ and $avg||U_j||$ are the maximum and average value of every user's total number of distinct query-url pairs. Obviously, AOL dataset is extremely highly-dimensional and sparse while MSNBC dataset is also sparse but contains a small universe size. Therefore, we conduct the experiments with randomly picked subsets of these two datasets. Note that we clean the AOL data by removing all the query-url pairs with total count less than 20; for MSNBC data, we keep the query-url universe since all of the query-url pairs are frequent.

Parameters Setup. We first examine the utility loss in the preprocess on varying $d \in [10, 80]$ for $b = 10^{-4}$ and 10^{-2} (where b is the combination of ϵ and δ: $\min\{\epsilon/2, \log\frac{1}{1-\delta}\}$).

In the remaining experiments, we let $d = 30$ and $\epsilon' = 1$, and inject a set of Laplacian noise $Lap(30)$ to the optimal solutions for end-to-end differential privacy. To observe the utility of CELS on different $b \in [10^{-5}, 10^{-1}]$, we demonstrate the KL-divergence of input and output as well as the percent of retained distinct query-url pairs (Recall), where the input dataset D is equally partitioned to $r = 2, 4, 6, 8, 10$ shares held by different parties. The utility is the average of the results obtained when $r = 2, 4, 6, 8, 10$.

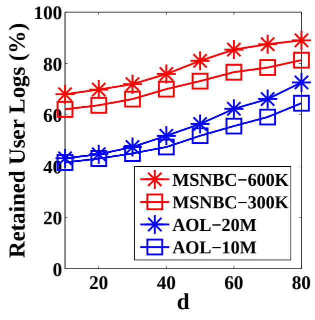
To depict the boosted utility with CELS protocol, we compare the output utility of CELS and local sanitization (all parties locally sanitize their own input with the same differential privacy requirement and integrate all the outputs together). In such experimental group, we let $r = 2, 4, 8$. The utility results of CELS aver averaged.

We also evaluate the efficiency of CELS protocol by examining the computation and communication overheads on varying number of parties $r = 2, 4, 6, 8, 10$ and input size (AOL: $100K - 20M$; MSNBC: $25K - 1.6M$).

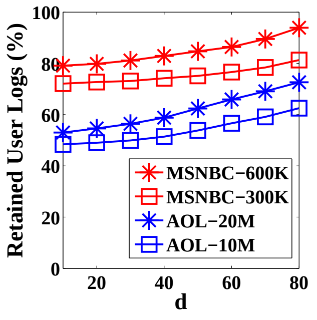
Platform. All the experiments are performed on an HP PC with Intel Core 2 Quad 3GHz 64-bit CPU and 6G RAM.

6.2 Prerequisite

Preprocess. For differential privacy of computing optimal output counts, we can preprocess the inputs to make the neighboring optimal solutions bounded. However, such process trades off the output utility for stronger privacy protection: some exceptional user logs might be removed. Fig. 2(a) and 2(b) present such utility loss (percent of retained user logs). For both AOL and MSNBC datasets, applying CELS with larger d or larger input size can clearly retain more percent of the user logs in the output. This is quite reasonable – larger d provides greater difference tolerance, and neighboring inputs with larger size are even more similar (thus the optimal solutions become closer).



(a) $b = 10^{-4}$



(b) $b = 10^{-2}$

Fig. 2. Retained User Logs

Note that AOL users posed most of their query-url pairs for less than 10 times [14], thus the difference between two neighboring inputs could be bounded with a relatively small d. It would be preferable to make d less than 100 (otherwise, the noise for the output counts might be too large). We can try different values of $d \leq 100$ and specify an appropriate value according to the utility requirements on the preprocessed search logs (Korolova et al. [24] chose $d \in [1, 80]$ in their experiments as well).

Maximum Output Count $|\mathbb{O}|$**.** While satisfying all the privacy conditions, the output count is indeed bounded by a constant [26] (also discussed in solving the NLP problem in Appendix B). Since the maximum output count $|\mathbb{O}|$ is important in solving such NLP problem, we regard it as a prerequisite of the experiments and present the result for different inputs and parameters.

Specifically, to better compare $|\mathbb{O}|$ with the input query-url pairs count $|D|$, we plot the maximum percent of them $\frac{|\mathbb{O}|}{|D|}$ in Fig. 3. Since search logs are highly-dimensional, the shown maximum percent $\frac{|\mathbb{O}|}{|D|}$ is sufficiently good for differential privacy guaranteed algorithms.

6.3 Utility of CELS

While solving the NLP problem in CELS protocol, the utility loss (KL-Divergence) can be minimized for any specified output size $|O| \leq \mathbb{O}$. With an appropriate output size $|O|$, the minimum KL-Divergence represent the best output utility (for details, please refer Appendix B).

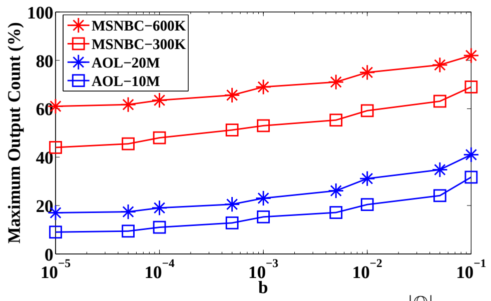


Fig. 3. Maximum Output Count (Percent $\frac{|\mathbb{O}|}{|D|}$)

In our experiments, we measure the minimum utility loss with two datasets (each has two sizes) in Fig. 4(a)-4(d). Note that for each input, we choose three different ratio $\frac{|O|}{|D|}$ (all no greater than $\frac{|\mathbb{O}|}{|D|}$ that are derived from the prerequisite), ensuring the feasibility of the NLP problems.

Fig. 4(a)-4(d) show that the utility loss decreases rapidly if we lower the requirement of differential privacy (with increased b) for all the inputs.

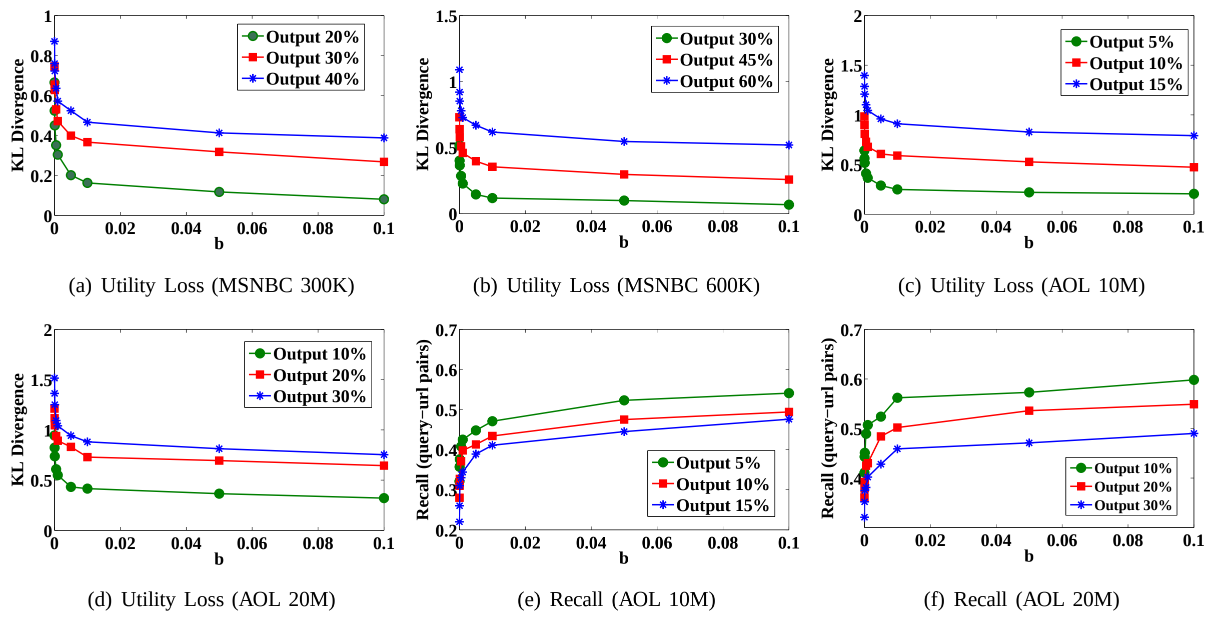
In addition, for every input sample dataset (e.g., AOL $20M$), *smaller specified total output count* $|O|$ can produce the sanitized output with *less KL-Divergence*. This is also reasonable – since KL-divergence only measures the difference between two probability distributions ($\{\frac{c_1}{|D|}, \dots, \frac{c_n}{|D|}\}$ and $\{\frac{x_1}{|O|}, \dots, \frac{x_n}{|O|}\}$), it is easier for every query-url pair (e.g., ϕ_i) to achieve its input count proportion $\frac{c_i}{|D|}$ in the output O with smaller $|O|$ while satisfying all the privacy conditions (constraints). In other words, given $|O|$, the ideal output count of ϕ_i is $x_i = |O| \cdot \frac{c_i}{|D|}$. Recall that satisfying the privacy conditions may reduce x_i and then deviate x_i from $|O| \cdot \frac{c_i}{|D|}$. With small $|O|$, $\forall i \in [1, n], x_i = |O| \cdot \frac{c_i}{|D|}$ are also small. Therefore, $\forall i \in [1, n], x_i = |O| \cdot \frac{c_i}{|D|}$ (or slightly reduced $i \in [1, n], x_i$) may easily satisfy all the privacy conditions, and then the minimum ratio difference (KL-Divergence) between $\{\frac{c_1}{|D|}, \dots, \frac{c_n}{|D|}\}$ and $\{\frac{x_1}{|O|}, \dots, \frac{x_n}{|O|}\}$ can be very small.

Finally, we present the retained query-url pair diversity in Fig. 4(e)-4(f) for AOL datasets (since MSNBC datasets have smaller universe size, the diversity is always maintained). With the same setup, the retained diversity (exhibited by the *recall of the number of distinct query-url pairs in the output*) increases for weakened privacy guarantee. Meanwhile, the preserved diversity in the output also shows the effectiveness of the entropy-biased utility measure in the sanitization.

6.4 Utility of CELS vs. Local Sanitization (LS)

When different parties need to securely sanitize their search logs and integrate the outputs with limited information disclosure, two possible sanitization methods can be utilized with identical privacy guarantee (same parameter b for differential privacy): 1) Local Sanitization (LS) – all parties locally impose the privacy conditions with its data with parameter b and a share of $|O|$, sample the local output and then securely integrate the outputs, and 2) CELS protocol.

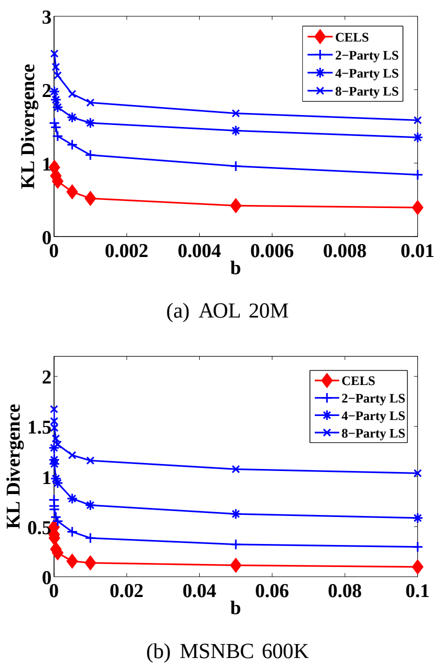
We first compare the utility of CELS and LS using two input datasets AOL $20M$ and MSNBC $600K$, where each of them is equally partitioned to $r = 2, 4, 8$ shares in three groups of experiments. For $r = 2$, two party each holds half of the input,



(a) Utility Loss (MSNBC 300K) (b) Utility Loss (MSNBC 600K) (c) Utility Loss (AOL 10M)

(d) Utility Loss (AOL 20M) (e) Recall (AOL 10M) (f) Recall (AOL 20M)

Fig. 4. KL-Divergence based Utility Loss and Retained Query-url Diversity (Recall)



(a) AOL 20M

(b) MSNBC 600K

Fig. 5. Boosted Utility by CELS Protocol

and they can either execute CELS protocol or LS (so do $r = 4$ and 8). In Fig. 5, the result of CELS protocol is compared to "2-Party LS", "4-Party LS" and "8-Party LS". Clearly, CELS protocol provides remarkably boosted output utility than local sanitization while ensuring the same level of privacy.

6.5 Efficiency

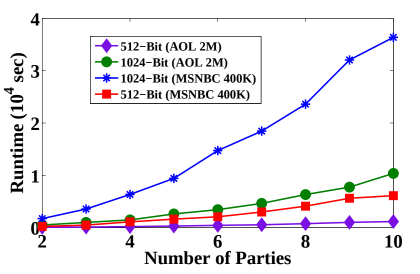
We have implemented the CELS protocol based on Paillier's homomorphic cryptosystem [37] (512-bit and 1024-bit key length resp.) and Pohlig-Hellman's commutative encryption scheme [41] (1024-bit key length), which is provably secure. Thus, we conduct two groups of experiments to validate the computation costs:

1) given fixed dataset (e.g., AOL $2M$ or MSNBC $400K$), testing the overall computation cost of CELS protocol by increasing the number of parties, where every party holds an equal share of the input. The experimental result is given in Fig. 6(a), which shows a linear increasing trend.
2) given fixed number of parties (e.g., 6 parties), testing the overall computation cost of CELS protocol by increasing the input size, where every party also holds equally partitioned input. Fig. 6(b) and 6(c) also demonstrate very good computational scalability on varying inputs for both datasets and different key length.

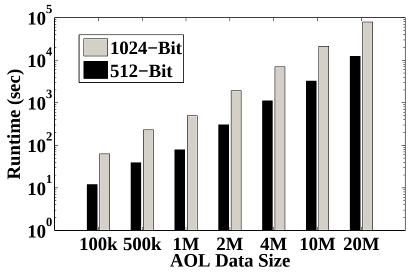
In Table 5, we present the communication overheads required in our CELS protocol (the overall bits/Bytes transferred among all the parties) where the key length for homomorphic cryptosystem and commutative encryption is given as 512-bit and 1024-bit respectively. The overall bandwidth consumption of all parties is very affordable for large inputs in CELS protocol, and the bandwidth grows slowly as the number of parties increases. Moreover, in the protocol, the relatively small amount of overall sent/received data (based on large inputs and up to 10 parties) is well balanced for all the participants – each consumes almost identical bandwidth. Therefore, such low bandwidth requirement enables our CELS protocol to be readily implemented and scaled to large inputs in most networking environments.

TABLE 5
All Parties' Total Bandwidth Consumption (Megabytes)

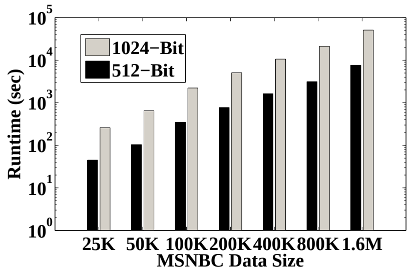
# of Parties (r)	2	4	6	8	10
AOL 1M	3.7	14.3	30.3	56.2	85.2
AOL 2M	9.8	28.2	65.3	121.2	183.8
AOL 4M	19.3	66.0	152.6	257.4	405.3
MSNBC 200K	2.2	8.8	19.1	32.6	46.2
MSNBC 400K	4.6	18.3	40.9	62.1	87.3
MSNBC 800K	8.9	36.5	89.3	132.7	194.6



(a) Varying Number of Parties (Two Datasets)



(b) Varying Input Size (AOL)



(c) Varying Input Size (MSNBC)

Fig. 6. Scalability of CELS Protocol

7 Related Work

Search Log Anonymization. Following the AOL search log incident, there has been some work on privately publishing search logs. Adar [1] proposes a secret sharing scheme where a query must appear at least t times before it can be decoded. Kumar et al. [27] showed that token based hashing is an anonymization that does not work. More recently, some privacy models [17], [24], [25], [30] have been proposed to make search log release possible. He et al. [17], Hong et al. [25] and Liu et al. [30] anonymized search logs based on k-anonymity. Korolova et al. [24] first applied differential privacy to search log release by adding Laplacian noise. Götz et al. [13] analyzed algorithms of publishing frequent keywords, queries and clicks in search logs and conducted a comparison for two relaxations of ϵ-differential privacy. Feild et al. [9] presented a framework for collecting, storing and mining search logs in a distributed scenario.

Differential Privacy. Dwork et al. [8] first proposed differential privacy that provides sufficient privacy protection regardless of adversaries' prior knowledge. It has been extended to data release in various different contexts. Xiao et al. [46] introduced a data publishing technique which ensures ϵ-differential privacy while providing accurate answers for range-count queries. Hay et al. [16] presented a differentially private algorithm to release a provably private estimate of the degree distribution of a network. McSherry et al. [32] solved the problem of producing recommendations from collective user behavior while providing differential privacy. Two other recent work [15], [23] showed how to sanitize the matrix under differential privacy. Nissim et al. [36] addressed smooth sensitivity and sampling in differentially private data analysis. Li et al. [28] discussed sampling based differential privacy.

Secure Distributed Data Anonymization. Zhong et al. [48] presented two formulations for securely building global k-anonymous tabular data held by distributed sites. Jiang and Clifton [20] addressed the distributed k-anonymity problem for vertically partitioned data between two parties. Mohammed et al. [33] extended the above work to securely anonymizing distributed data with k-anonymity to multiple parties and malicious model. Mohammed et al. [34] also tackled the anonymization problem for centralized and distributed healthcare data with the privacy model LKC-privacy. Goryczka et al. [12] raised a privacy notion m-privacy to bound the number of colluding parties in distributed anonymization. Alhadidi et al. [3] presented a two-party protocol for publishing horizontally partitioned data with differential privacy and SMC. To the best of our knowledge, we take the first step towards securely sanitizing highly-dimensional data from multiple parties.

Finally, since our sanitization model securely solves a collaborative NLP problem and sample the output based on the optimal solution, some privacy-preserving collaborative optimization models [18], [19], [29], [31], [39], [40], [43], [44] in literature are also very relevant to our algorithm.

8 Conclusion

We have addressed the important practical problem of sanitizing search logs for potential storage and publish. Specifically, we presented a differentially private mechanism that can generate the output with identical schema with the input rather than statistical information. This significantly preserves the utility of the sanitized search logs. To better improve the output utility, we built a framework (CELS protocol) that involve distributed parties to securely generate the output while satisfying differential privacy. We proved the security of the protocol and differential privacy guarantee. Finally, the performance of CELS protocol has been experimentally validated with real datasets.

We can extend our work in several directions. First, in most of the literature on search log release, the database schema of the input and output does not include query time and the rank of the clicked url, thus it is an open problem to probe effective approaches for publishing search logs with more complex schema (that may cause additional privacy concern with the property of time series). Second, it is unclear that whether the adversaries can breach the privacy by inferring the correlations between users' query-url pairs or not, and whether the differential privacy guaranteed sanitization algorithms can handle such potential privacy breach or not is worth investigating. Finally, we also intend to develop incentive compatible sanitization protocols that are secure and honesty-reinforced against malicious adversaries.

References

[1] E. Adar. User 4xxxxx9: Anonymizing query logs. In *Workshop at the WWW '07*, 2007.
[2] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *HLT/EMNLP*, 2005.
[3] D. Alhadidi, N. Mohammed, B. C. M. Fung, and M. Debbabi. Secure distributed framework for achieving ϵ-differential privacy. In *PETS*, pages 120–139, 2012.
[4] M. Barbaro and T. Zeller Jr. A face is exposed for aol searcher no. 4417749, Augest 9, 2006. (New York Times).

[5] A. Cooper. A survey of query log privacy-enhancing techniques from a policy perspective. *TWEB*, 2(4), 2008.
[6] H. Deng, I. King, and M. R. Lyu. Entropy-biased models for query representation on the click graph. In *SIGIR*, pages 339–346, 2009.
[7] G. Dupret and M. Mendoza. Automatic query recommendation using click-through data. In *IFIP PPAI*, pages 303–312, 2006.
[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
[9] H. A. Feild, J. Allan, and J. Glatt. Crowdlogging: distributed, private, and anonymous search logging. In *SIGIR*, pages 375–384, 2011.
[10] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457:1012 – 1014, February 2009.
[11] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter Encryption Schemes. Cambridge University Press, 2004.
[12] S. Goryczka, L. Xiong, and B. C. M. Fung. m-privacy for collaborative data publishing. In *CollaborateCom*, 2011.
[13] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Publishing search logs - a comparative study of privacy guarantees. *IEEE TKDE*, 24(3):520–532, 2012.
[14] K. Hafner. Researchers yearn to use aol logs, but they hesitate, Augest 23, 2006. (New York Times).
[15] M. Hardt and A. Roth. Beating randomized response on incoherent matrices. In *STOC*, pages 1255–1268, 2012.
[16] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.
[17] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1):934–945, 2009.
[18] Y. Hong, J. Vaidya, and H. Lu. Efficient Distributed Linear Programming with Limited Disclosure. In *DBSEC*, pages 170–185, 2011.
[19] Y. Hong, J. Vaidya, and H. Lu. Secure and Efficient Distributed Linear Programming. *Journal of Computer Security*, 20(5):583–634, 2012.
[20] W. Jiang and C. Clifton. A secure distributed framework for achieving k-anonymity. *VLDB J.*, 15(4):316–333, 2006.
[21] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142. ACM, 2002.
[22] R. Jones, R. Kumar, B. Pang, and A. Tomkins. “I know what you did last summer”: query logs and user privacy. In *CIKM ’07*, pages 909–914, 2007.
[23] K. Kenthapadi, A. Korolova, I. Mironov, and N. Mishra. Privacy via the johnson-lindenstrauss transform. *CoRR*, abs/1204.2606, 2012.
[24] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180, 2009.
[25] Y. Hong, X. He, J. Vaidya, N. R. Adam, and V. Atluri. Effective anonymization of query logs. In *CIKM*, pages 1465–1468, 2009.
[26] Y. Hong, J. Vaidya, H. Lu, and M. Wu. Differentially private search log sanitization with optimal output utility. In *EDBT*, pages 50–61, 2012.
[27] R. Kumar, J. Novak, B. Pang, and A. Tomkins. On anonymizing query logs via token-based hashing. In *WWW*, pages 629–638, 2007.
[28] N. Li, W. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *ASIACCS*, pages 32–33, 2012.
[29] W. Li, H. Li, and C. Deng. Privacy-preserving horizontally partitioned linear programs with inequality constraints. *Optimization Letters*, 7(1):137–144, 2013.
[30] J. Liu and K. Wang. Enforcing vocabulary k-anonymity by semantic similarity based clustering. In *ICDM*, pages 899–904, 2010.
[31] O. L. Mangasarian. Privacy-preserving horizontally partitioned linear programs. *Optimization Letters*, 6(3):431–436, 2012.
[32] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *KDD*, pages 627–636, 2009.
[33] N. Mohammed, B. C. M. Fung, and M. Debbabi. Anonymity meets game theory: secure data integration with malicious participants. *VLDB J.*, 20(4):567–588, 2011.
[34] N. Mohammed, B. C. M. Fung, P. C. K. Hung, and C. K. Lee. Centralized and distributed anonymization for high-dimensional healthcare data. *TKDD*, 4(4):18, 2010.
[35] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *CCS*, pages 59–66, 1998.
[36] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.
[37] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Eurocrypt ’99*, pages 223–238, 1999.
[38] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, 1982.
[39] Y. Hong. Privacy-preserving Collaborative Optimization. *Ph.D. Dissertation*, Rutgers University, 2013.
[40] Y. Hong, J. Vaidya, H. Lu, and L. Wang. Collaboratively Solving the Traveling Salesman Problem with Limited Disclosure. In *DBSEC*, pages 179–194, 2014.
[41] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, IT-24:106–110, 1978.
[42] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.
[43] Y. Hong, J. Vaidya, and S. Wang. A Survey of Privacy-aware Supply Chain Collaboration: From Theory to Applications. *Journal of Information Systems*, 28(1):243–268, AAA, 2014.
[44] Y. Hong, J. Vaidya, H. Lu, and B. Shafiq. Privacy-preserving Tabu Search for Distributed Graph Coloring. In *PASSAT*, pages 951–958, 2011.
[45] H.A. Taha. *Operations Research: An Introduction*. Prentice Hall, 2010.
[46] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.
[47] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE FOCS*, pages 162–167, 1986.
[48] S. Zhong, Z. Yang, and R. N. Wright. Privacy-enhancing -anonymization of customer data. In *PODS*, pages 139–147, 2005.

Yuan Hong is an Assistant Professor in the Department of Information Technology Management at University at Albany, SUNY. He received the Ph.D. degree in Information Technology from Rutgers University. His research interests primarily lie at the intersection of Privacy, Security, Digital Forensics, Optimization and Data Mining.

Jaideep Vaidya is an Associate Professor in the MSIS department at Rutgers University. He received the Ph.D. degree in Computer Science from Purdue University. His general area of research is in data mining, data management, security, and privacy. He has published over 100 technical papers in peer-reviewed journals and conference proceedings, and has received three best paper awards. He is also the recipient of an NSF Career Award and is a senior member of the IEEE.

Haibing Lu is an Assistant Professor in the Department of Operations Management and Information Systems at Santa Clara University. He received the Ph.D. degree in Information Technology from Rutgers University. His general area of research is in information security and privacy, and data mining.

Panagiotis Karras is an Assistant Professor at Skoltech. He earned a Ph.D. in Computer Science from the University of Hong Kong. His interests are in the confluence of the management, mining, and security of data. His research has been awarded by the Hong Kong Institution of Science and funded by the Lee Kuan Yew Endowment and the Skolkovo Foundation. He regularly publishes in and serves as PC member and referee for major conferences and journals in the above areas.

Sanjay Goel is an Associate Professor in the Department of Information Technology Management at University at Albany, SUNY. He received the Ph.D. degree from Rensselaer Polytechnic Institute. His general area of research is in self-organized systems for modeling of autonomous computer security systems, distributed service-based computing, network security and active networks.

APPENDIX A FREQUENTLY USED NOTATIONS

TABLE 6
Frequently Used Notations for Centralized Input

ϕ_i	a query-url pair
(ϕ_i, u_j)	user u_j's query-url pair ϕ_i
D and D'	neighboring inputs
U_j	the differential user log between D and D'
O	output search log
S	an arbitrary output set
S_1, S_2	S is divided into S_1 (with u_j) and S_2 (without u_j)
$\|D\|$	total count of all query-url pairs in D
$\|O\|$	total count of all query-url pairs in O
$\|\mathbb{O}\|$	the maximum value of $\|O\|$ (given D)
c_i and x_i	the count of ϕ_i in D and O
x	$x = (x_1, \dots, x_n)$
c_{ij} and x_{ij}	the count of u_j's ϕ_i in D and O
Constants	$b = \min\{\epsilon, \log \frac{1}{1-\delta}\}, t_{ij} = \frac{c_i}{c_i - c_{ij}}$

TABLE 7
Frequently Used Notations for Distributed Inputs

$P_1, \dots, P_r$	r distributed parties
P_0	external party (or data recipient, cloud)
D_k	P_k's input search logs
D	the global input $D = \bigcup_{k=1}^{r} D_k$
U_j^k	the jth user log in D_k
m_k	number of user logs in D_k
c_i^k	query-url pair ϕ_i's count in D_k
c_{ij}^k	query-url pair ϕ_i's count in U_j^k
$\overrightarrow{v_k} = (c_1^k, \dots, c_n^k)$	P_k's query-url pair count vector
$\overrightarrow{v} = \sum_{k=1}^{r} \overrightarrow{v_k}$	query-url pair count vector in D
π_k	P_k's Permutation
π_k^{-1}	P_k's inverse permutation
(pk, sk)	public-private key pair
constants	$t_{ij}^k = \frac{c_i}{c_i - c_{ij}^k}$ $c_i = \sum_{k=1}^{r} c_i^k$, $m = \sum_{k=1}^{r} m_k$

APPENDIX B THE NLP IN SANITIZATION

B.1 Refining the NLP Problem

Since the entropy-biased utility objective function seeks the minimized difference between two probability distribution functions (two fraction vectors), only its high value cannot always represent good utility. For example, if the total number of all query-url pairs in the output is small, and ratios are very close to the original ones, KL divergence would be very small (this cannot reflect good utility simply because the output only includes very few query-url pairs). Alternatively, we can specify a fixed total output count $|O|$ (of all the query-url pairs) and seek the minimum KL divergence.

1) The Upper Bound of $|O|$. Essentially, $|O|$ is bounded by all the privacy conditions derived from the input D. If we regard the output count $|O| = \sum_{i=1}^{n} x_i$ as the utility objective function, we can simply find the upper bound of $|O|$ by solving the following integer linear programming (LP) problem:

$$\max : \quad \sum_{i=1}^{n} x_i$$
$$s.t. \begin{cases} \forall j \in [1, m], \sum_{i=1}^{n} (x_i \cdot \log t_{ij}) \le b \\ \forall x_i \ge 0 \text{ and } x_i \text{ is an integer} \end{cases}$$

Note that the above problem can be solved using linear relaxation and a standard solver such as simplex method [38]. We denote the computed upper bound of $|O|$ as $|\mathbb{O}|$.

2) Simplified Objective Function. If we specify a fixed output count $|O| \le |\mathbb{O}|$, Equation 6 can be considered as an NLP problem where the non-linear objective function can be expressed as a sum of some single variable functions (namely *separable* functions [45]), simplified as below:

$$\min : \sum_{i=1}^{n} \left[\frac{c_i}{|D|} \log\left(\frac{c_i}{|D|} \cdot \frac{|O| + n}{x_i + 1}\right) \right]$$
$$\iff \min : \sum_{i=1}^{n} \left[\frac{c_i}{|D|} \log\left(\frac{c_i \cdot (|O| + n)}{|D|}\right) \right] - \sum_{i=1}^{n} \left[\frac{c_i}{|D|} \log(x_i + 1) \right]$$
$$\iff \max : \sum_{i=1}^{n} [c_i \log(x_i + 1)]$$

3) Fixed Output Count. An extra linear constraint should be satisfied $\sum_{i=1}^{n} x_i = |O|$ since the total count of all the query-url pairs in the output is fixed as $|O|$.

B.2 Solving the NLP Problem

As a result, the formulated optimization problem is equivalent to the following NLP problem with linear constraints:

$$\max : \quad \sum_{i=1}^{n} [c_i \log(x_i + 1)]$$
$$s.t. \begin{cases} \forall j \in [1, m], \sum_{i=1}^{n} (x_i \cdot \log t_{ij}) \le b \\ \sum_{i=1}^{n} x_i = |O| \\ \forall x_i \ge 0 \text{ and } x_i \text{ is an integer} \end{cases} \quad (9)$$

Any separable convex problem can be solved by linear approximation [45] with the idea of convexity combination. In this problem, all n non-linear functions $1 \le x_i \le n, \log(x_i + 1)$ are strictly convex since the derivative $\frac{\partial \log(x_i+1)}{\partial x_i} = \frac{1}{x_i} > 0$. Hence, $1 \le x_i \le n, \log(x_i + 1)$ can be approximated by a piecewise linear function [45]. Specifically, we can approximate $\forall i \in [1, n], \log(x_i + 1)$ over the interval $[0, |O|]$ ($\forall i \in [1, n], 0 \le x_i \le |O|$) as follows:

- define $a_s, s = 1, 2, \dots, K$ as the sth breakpoint on interval $[0, |O|]$ where $a_1 < a_2 < \dots < a_K$.
- $\log(x_i + 1) \approx \sum_{s=1}^{K} \log(a_s + 1) w_s^i$ where w_s^i is a nonnegative weight associated with the sth breakpoint and variable x_i. Note that $\sum_{s=1}^{K} w_s^i = 1$ (the non-linear function $\log(x_i + 1)$ is approximated as a convex combination of a set of logarithmic values $1 \le s \le K, \log(a_s + 1)$ based on a_s).

- since $a_1, \dots, a_K$ and $\log(a_1+1), \dots, \log(a_K+1)$ are constants in the approximated LP problem, we have $x_i = \sum_{s=1}^{K} a_s w_s^i$ and $\log(x_i+1) = \sum_{s=1}^{K} \log(a_s+1) w_s^i$ where $1 \le s \le K$, w_s^i are new variables replacing x_i.
- similarly, we can approximate all linear functions (w.r.t. $1 \le i \le n, \log(x_i+1)$) with the same set of breakpoints on $[0, |O|]$.

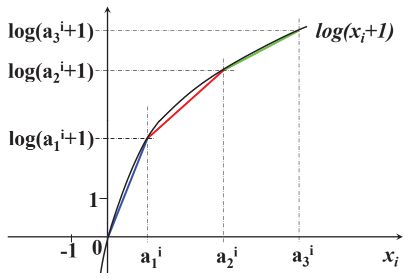


Fig. 7. Piecewise Linear Approximation of $\log(x_i+1)$

Fig. 7 shows an example of approximating convex function $\log(x_i+1)$, three breakpoints excluding the endpoints in x_i's value range $[0, |O|]$ are specified as a_1, a_2, a_3. Therefore, $\log(x_i+1)$ can be approximately expressed as $\log(a_1+1)w_1^i + \cdots + \log(a_3+1)w_3^i$ (the convex combination of the values $\log(a_1+1)$,...,$\log(a_3+1)$) where $w_1^i + w_2^i + w_3^i = 1$ and $x_i = a_1 w_1^i + a_2 w_2^i + a_3 w_3^i$.

Then, we present the approximated LP problem for Equation 9:

$$\max: \quad \sum_{i=1}^{n} \left[c_i \sum_{s=1}^{K} w_s^i \cdot \log(a_s+1) \right]$$
$$s.t. \begin{cases} \forall j \in [1,m], \sum_{i=1}^{n} (\sum_{s=1}^{K} a_s w_s^i \cdot \log t_{ij}) \le b \\ \sum_{i=1}^{n} \sum_{s=1}^{K} a_s w_s^i = |O| \\ \forall i \in [1,n], \sum_{s=1}^{K} w_s^i = 1 \\ \forall i \in [1,n], s \in [1,K], 0 \le w_s^i \le 1 \end{cases} \tag{10}$$

Note that the above LP problem is always feasible and bounded if $|O|$ is specified as a value in $[0, |\mathbb{O}|]$, since all the m differential privacy constraints are feasible and bounded.

Finally, Equation 10 can be efficiently solved with Simplex or Revised Simplex method [45], and the optimal solution of the original optimization problem (Equation 6) can be straightforwardly derived with expressions: $\forall i \in [1,n], x_i = \lfloor \sum_{s=1}^{K} a_s w_s^i \rfloor$, where linear relaxation returns the integer part of the decimal numbers.

Appendix C
Preprocess in Sanitization

We first look at Case (1) $D = D' \bigcup U_j$:

1) use Equation 6 to formulate two optimization problems with the neighboring inputs D and $D' = D \setminus U_j$ respectively, and solve them.
2) if the count difference of any query-url pair in both optimal solutions is greater than d, remove U_j from D and restart the preprocessing procedure with input $D \setminus U_j$ (D is updated as $D \setminus U_j$ at this time) [3].

The above preprocessing procedure iteratively examines every user log U_j in the input search log D. Thereby, if U_j causes an unbounded optimal count difference, U_j will be removed and the preprocessing procedure restarts. Thus, the optimal counts computed from any two neighboring inputs D and $D' = D \setminus U_j$ should be bounded: first, in case of any removal of U_j from D, restarting the procedure with the updated input $D \setminus U_j$ is identical to starting the procedure with input D', thus the optimal counts derived from inputs D and D' are identical and definitely bounded by d; second, if there is no user log removed from D in the preprocessing procedure (the best case), then the optimal counts derived from D and $D' = D \setminus U_j$ are bounded by d.

For Case (2) of D and D', if $D' = D \bigcup U_j$, preprocessing D' could also bound the optimal count difference for D and D'. For Case (3) of D and D', if U_j is different in D and D', preprocessing both D and D' could also bound the optimal count difference for D and D'.

3. The optimization problems result from any two neighboring inputs (especially the large neighboring inputs) generate similar optimal solutions. Thus, if d is not too small, the output count difference can be bounded by d. Otherwise, if d is required to be sufficiently small (for reducing sensitivity/noise), we remove some user logs (that cause large differences in two optimal solutions). This allows us to trade off utility for end-to-end differential privacy.