Publishing Search Logs—A Comparative Study of Privacy Guarantees

Michaela Götz, Ashwin Machanavajjhala, Guozhang Wang, Xiaokui Xiao, and Johannes Gehrke

Abstract—Search engine companies collect the "database of intentions," the histories of their users' search queries. These search logs are a gold mine for researchers. Search engine companies, however, are wary of publishing search logs in order not to disclose sensitive information. In this paper, we analyze algorithms for publishing frequent keywords, queries, and clicks of a search log. We first show how methods that achieve variants of k-anonymity are vulnerable to active attacks. We then demonstrate that the stronger guarantee ensured by ϵ-differential privacy unfortunately does not provide any utility for this problem. We then propose an algorithm ZEALOUS and show how to set its parameters to achieve (ϵ, δ)-probabilistic privacy. We also contrast our analysis of ZEALOUS with an analysis by Korolova et al. [17] that achieves (ϵ', δ')-indistinguishability. Our paper concludes with a large experimental study using real applications where we compare ZEALOUS and previous work that achieves k-anonymity in search log publishing. Our results show that ZEALOUS yields comparable utility to k-anonymity while at the same time achieving much stronger privacy guarantees.

Index Terms—Security, integrity, and protection, general, database management, information technology and systems, web search, general, information storage and retrieval, information technology and systems.

1 Introduction

Civilization is the progress toward a society of privacy. The savage's whole existence is public, ruled by the laws of his tribe. Civilization is the process of setting man free from men. —Ayn Rand.

My favorite thing about the Internet is that you get to go into the private world of real creeps without having to smell them. —Penn Jillette.

Search engines play a crucial role in the navigation through the vastness of the web. Today's search engines do not just collect and index webpages, they also collect and mine information about their users. They store the queries, clicks, IP-addresses, and other information about the interactions with users in what is called a *search log*. Search logs contain valuable information that search engines use to tailor their services better to their users' needs. They enable the discovery of trends, patterns, and anomalies in the search behavior of users, and they can be used in the development and testing of new algorithms to improve search performance and quality. Scientists all around the world would like to tap this gold mine for their own research; search engine companies, however, do not release them because they contain sensitive information about their users, for example searches for diseases, lifestyle choices, personal tastes, and political affiliations.

- *M. Götz, G. Wang, and J. Gehrke are with the Department of Computer Science, Cornell University, Upson Hall, Ithaca, NY 14853. E-mail: {goetz, guoz, johannes}@cs.cornell.edu.*
- *A. Machanavajjhala is with Yahoo! Research in Silicon Valley, 4301 Great America Pkwy, Santa Clara, CA 95054. E-mail: mvnak@yahoo-inc.com.*
- *X. Xiao is with the School of Computer Engineering, Nanyang Technological University, Block N4, Room #02c-115, 50 Nanyang Avenue, Singapore 639798. E-mail: xkxiao@ntu.edu.sg.*

Manuscript received 15 Dec. 2009; revised 1 July 2010; accepted 17 Aug. 2010; published online 31 Jan. 2011.
Recommended for acceptance by D. Srivastava.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-12-0843.
Digital Object Identifier no. 10.1109/TKDE.2011.26.

The only release of a search log happened in 2006 by AOL, and it went into the annals of tech history as one of the great debacles in the search industry.[1] AOL published three months of search logs of 650,000 users. The only measure to protect user privacy was the replacement of user-ids with random numbers—utterly insufficient protection as the New York Times showed by identifying a user from Lilburn, Georgia [4], whose search queries not only contained identifying information but also sensitive information about her friends' ailments.

The AOL search log release shows that simply replacing user-ids with random numbers does not prevent information disclosure. Other ad hoc methods have been studied and found to be similarly insufficient, such as the removal of names, age, zip codes, and other identifiers [14] and the replacement of keywords in search queries by random numbers [18].

In this paper, we compare formal methods of limiting disclosure when publishing frequent keywords, queries, and clicks of a search log. The methods vary in the guarantee of disclosure limitations they provide and in the amount of useful information they retain. We first describe two negative results. We show that existing proposals to achieve *k-anonymity* [23] in search logs [1], [21], [12], [13] are insufficient in the light of attackers who can actively influence the search log. We then turn to *differential privacy* [9], a much stronger privacy guarantee; however, we show that it is impossible to achieve good utility with differential privacy.

We then describe Algorithm ZEALOUS,[2] developed independently by Korolova et al. [17] and us [10] with the goal to achieve relaxations of differential privacy. Korolova et al. showed how to set the parameters of ZEALOUS to

1. http://en.wikipedia.org/wiki/AOL_search_data_scandal describes the incident, which resulted in the resignation of AOL's CTO and an ongoing class action lawsuit against AOL resulting from the data release.
2. **Zearch Log publising**.

guarantee (ϵ, δ)-indistinguishability [8], and we here offer a new analysis that shows how to set the parameters of ZEALOUS to guarantee (ϵ, δ)-*probabilistic differential privacy* [20] (Section 4.2), a much stronger privacy guarantee as our analytical comparison shows.

Our paper concludes with an extensive experimental evaluation, where we compare the utility of various algorithms that guarantee anonymity or privacy in search log publishing. Our evaluation includes applications that use search logs for improving both search experience and search performance, and our results show that ZEALOUS' output is sufficient for these applications while achieving strong formal privacy guarantees.

We believe that the results of this research enable search engine companies to make their search log available to researchers without disclosing their users' sensitive information: Search engine companies can apply our algorithm to generate statistics that are (ϵ, δ)-probabilistic differentially private while retaining good utility for the two applications we have tested. Beyond publishing search logs we believe that our findings are of interest when publishing frequent item sets, as ZEALOUS protects privacy against much stronger attackers than those considered in existing work on privacy-preserving publishing of frequent items/item sets [19].

The remainder of this paper is organized as follows. We start with some background in Section 2. Our negative results are presented in Section 3. We then describe Algorithm ZEALOUS and its analysis in Section 4. We compare indistinguishability with probabilistic differential privacy in Section 5. Section 6 shows the results of an extensive study of how to set parameters in ZEALOUS, and Section 7 contains a thorough evaluation of ZEALOUS in comparison with previous work. We conclude with a discussion of related work and other applications.

2 Preliminaries

In this section, we introduce the problem of publishing frequent keywords, queries, clicks, and other items of a search log.

2.1 Search Logs

Search engines such as Bing, Google, or Yahoo log interactions with their users. When a user submits a query and clicks on one or more results, a new entry is added to the search log. Without loss of generality, we assume that a search log has the following schema:

$$\langle \text{USER-ID, QUERY, TIME, CLICKS} \rangle,$$

where a USER-ID identifies a user, a QUERY is a set of keywords, and CLICKS is a list of *urls* that the user clicked on. The user-id can be determined in various ways; for example, through cookies, IP addresses, or user accounts. A *user history* or *search history* consists of all search entries from a single user. Such a history is usually partitioned into *sessions* containing similar queries; how this partitioning is done is orthogonal to the techniques in this paper. A *query pair* consists of two subsequent queries from the same user within the same session.

We say that a user history *contains* a keyword k if there exists a search log entry such that k is a keyword in the query of the search log. A *keyword histogram* of a search log S records for each keyword k the number of users c_k whose search history in S contains k. A keyword histogram is thus a set of pairs (k, c_k). We define the *query histogram*, the *query pair histogram*, and the *click histogram* analogously. We classify a keyword, query, consecutive query, click in a histogram to be *frequent* if its count exceeds some predefined threshold τ; when we do not want to specify whether we count keywords, queries, etc., we also refer to these objects as *items*.

With this terminology, we can define our goal as publishing frequent items (utility) without disclosing sensitive information about the users (privacy). We will make both the notion of utility and privacy more formal in the next sections.

2.2 Disclosure Limitations for Publishing Search Logs

A simple type of disclosure is the identification of a particular user's search history (or parts of the history) in the published search log. The concept of k-anonymity has been introduced to avoid such identifications.

Definition 1 (k-anonymity [23]). *A search log is k-anonymous if the search history of every individual is indistinguishable from the history of at least $k-1$ other individuals in the published search log.*

There are several proposals in the literature to achieve different variants of k-anonymity for search logs. Adar proposes to partition the search log into sessions and then to discard queries that are associated with fewer than k different user-ids. In each session, the user-id is then replaced by a random number [1]. We call the output of Adar's Algorithm a *k-query anonymous search log*. Motwani and Nabar add or delete keywords from sessions until each session contains the same keywords as at least $k-1$ other sessions in the search log [21], following by a replacement of the user-id by a random number. We call the output of this algorithm a *k-session anonymous search log*. He and Naughton generalize keywords by taking their prefix until each keyword is part of at least k search histories and publish a histogram of the partially generalized keywords [12]. We call the output a *k-keyword anonymous search log*. Efficient ways to anonymize a search log are also discussed by Yuan et al. [13].

Stronger disclosure limitations try to limit what an attacker can learn about a user. Differential privacy guarantees that an attacker learns roughly the same information about a user whether or not the search history of that user was included in the search log [9]. Differential privacy has previously been applied to contingency tables [3], learning problems [5], [16], synthetic data generation [20] and more.

Definition 2 (ϵ-differential privacy [9]). *An algorithm $\mathcal{A}$ is ϵ-differentially private if for all search logs S and S' differing in the search history of a single user and for all output search logs O:*

$$Pr[\mathcal{A}(S) = O] \leq e^{\epsilon} Pr[\mathcal{A}(S') = O].$$

This definition ensures that the output of the algorithm is insensitive to changing/omitting the complete search

history of a single user. We will refer to search logs that only differ in the search history of a single user as *neighboring search logs*. Similar to the variants of k-anonymity, we could also define variants of differential privacy by looking at neighboring search logs that differ only in the content of one session, one query or one keyword. However, we chose to focus on the strongest definition in which an attacker learns roughly the same about a user even if that user's whole search history was omitted.

Differential privacy is a very strong guarantee and in some cases it can be too strong to be practically achievable. We will review two relaxations that have been proposed in the literature. Machanavajjhala et al. proposed the following probabilistic version of differential privacy.

Definition 3 (probabilistic differential privacy [20]). *An algorithm $\mathcal{A}$ satisfies (ϵ, δ)-probabilistic differential privacy if for all search logs S we can divide the output space Ω into two sets Ω_1, Ω_2 such that*

$$(1)\ \Pr[\mathcal{A}(S) \in \Omega_2] \leq \delta, \text{ and}$$

for all neighboring search logs S' and for all $O \in \Omega_1$:

$$(2) e^{-\epsilon} \Pr[\mathcal{A}(S') = O] \leq \Pr[\mathcal{A}(S) = O] \leq e^{\epsilon} \Pr[\mathcal{A}(S') = O].$$

This definition guarantees that algorithm $\mathcal{A}$ achieves ϵ-differential privacy with high probability ($\geq 1 - \delta$). The set Ω_2 contains all outputs that are considered privacy breaches according to ϵ-differential privacy; the probability of such an output is bounded by δ.

The following relaxation has been proposed by Dwork et al. [8].

Definition 4 (indistinguishability [8]). *An algorithm $\mathcal{A}$ is (ϵ, δ)-indistinguishable if for all search logs S, S' differing in one user history and for all subsets $\mathcal{O}$ of the output space Ω:*

$$\Pr[\mathcal{A}(S) \in \mathcal{O}] \leq e^{\epsilon} \Pr[\mathcal{A}(S') \in \mathcal{O}] + \delta.$$

We will compare these two definitions in Section 5. In particular, we will show that probabilistic differential privacy implies indistinguishability, but the converse does not hold: We show that there exists an algorithm that is (ϵ', δ')-indistinguishable yet not (ϵ, δ)-probabilistic differentially private for any ϵ and any $\delta < 1$, thus showing that (ϵ, δ)-probabilistic differential privacy is clearly stronger than (ϵ', δ')-indistinguishability.

2.3 Utility Measures

We will compare the utility of algorithms producing sanitized search logs both theoretically and experimentally.

2.3.1 Theoretical Utility Measures

For simplicity, suppose we want to publish all items (such as keywords, queries, etc.) with frequency at least τ in a search log; we call such items *frequent items*; we call all other items *infrequent items*. Consider a discrete domain of items $\mathcal{D}$. Each user contributes a set of these items to a search log S. We denote by $f_d(S)$ the frequency of item $d \in \mathcal{D}$ in search log S. We drop the dependency from S when it is clear from the context.

We define the inaccuracy of a (randomized) algorithm as the expected number of items it gets wrong, i.e., the number of frequent items that are not included in the output, plus the number of infrequent items that are included in the output. We do not expect an algorithm to be perfect. It may make mistakes for items with frequency very close to τ, and thus we do not take these items in our notion of accuracy into account. We formalize this "slack" by a parameter ξ, and given ξ, we introduce the following new notions. We call an item d with frequency $f_d \geq \tau + \xi$ a *very frequent item* and an item d with frequency $f_d \leq \tau - \xi$ a *very infrequent item*. We will measure the inaccuracy of an algorithm then only using its inability to retain the very frequent items and its inability to filter out the very infrequent items.

Definition 5 ($(\mathcal{A}, S)$-inaccuracy). *Given an algorithm $\mathcal{A}$ and an input search log S, the $(\mathcal{A}, S)$-inaccuracy with slack ξ is defined as*

$$E[|\{d \in \mathcal{A}(S) | f_d(S) > \tau - \xi\} \cup \{d \notin \mathcal{A}(S) | f_d(S) > \tau + \xi\}|].$$

The expectation is taken over the randomness of the algorithm. As an example, consider the simple algorithm that always outputs the empty set; we call this algorithm the *baseline algorithm*. On input S, the Baseline Algorithm has an inaccuracy equal to the number of items with frequency at least $\tau + \xi$.

For the results in the next sections, it will be useful to distinguish the error of an algorithm on the very frequent items and its error on the very infrequent items. We can rewrite the inaccuracy as:

$$\sum_{d: f_d(S) > \tau + \xi} 1 - \Pr[d \in \mathcal{A}(S)] + \sum_{d \in D: f_d(S) < \tau - \xi} \Pr[d \in \mathcal{A}(S)].$$

Thus, the $(\mathcal{A}, S)$-inaccuracy with slack ξ can be rewritten as the inability to retain the very frequent items plus the inability to filter out the very infrequent items. For example, the baseline algorithm has an inaccuracy to filter of 0 inaccuracy to retain equal to the number of very frequent items.

Definition 6 (c-accuracy). *An algorithm $\mathcal{A}$ is c-accurate if for any input search log S and any very frequent item d in S, the probability that $\mathcal{A}$ outputs d is at least c.*

2.3.2 Experimental Utility Measures

Traditionally, the utility of a privacy-preserving algorithm has been evaluated by comparing some statistics of the input with the output to see "how much information is lost." The choice of suitable statistics is a difficult problem as these statistics need to mirror the sufficient statistics of applications that will use the sanitized search log, and for some applications the sufficient statistics are hard to characterize. To avoid this drawback, Brickell and Shmatikov [6] measure the utility with respect to data mining tasks and they take the actual classification error of an induced classifier as their utility metric.

In this paper, we take a similar approach. We use two real applications from the information retrieval community: Index caching, as a representative application for search performance, and query substitution, as a representative application for search quality. For both application, the sufficient statistics are histograms of keywords, queries, or query pairs.

Index caching. Search engines maintain an inverted index which, in its simplest instantiation, contains for each

keyword a posting list of identifiers of the documents in which the keyword appears. This index can be used to answer search queries, but also to classify queries for choosing sponsored search results. The index is usually too large to fit in memory, but maintaining a part of it in memory reduces response time for all these applications. We use the formulation of the *index caching problem* from Baeza-Yates [2]. We are given a keyword search workload, a distribution over keywords indicating the likelihood of a keyword appearing in a search query. It is our goal to cache in memory a set of posting lists that for a given workload maximizes the cache-hit-probability while not exceeding the storage capacity. Here, the hit probability is the probability that the posting list of a keyword can be found in memory given the keyword search workload.

Query substitution. Query substitutions are suggestions to rephrase a user query to match it to documents or advertisements that do not contain the actual keywords of the query. Query substitutions can be applied in query refinement, sponsored search, and spelling error correction [15]. Algorithms for query substitution examine query pairs to learn how users re-phrase queries. We use an algorithm developed by Jones et al. [15].

3 Negative Results

In this section, we discuss the deficiency of two existing models of disclosure limitations for search log publication. Section 3.1 focuses on k-anonymity, and Section 3.2 investigates differential privacy.

3.1 Insufficiency of Anonymity

k-anonymity and its variants prevent an attacker from uniquely identifying the user that corresponds to a search history in the sanitized search log. Nevertheless, even without unique identification of a user, an attacker can infer the keywords or queries used by the user. k-anonymity does not protect against this severe information disclosure.

There is another issue largely overlooked with the current implementations of anonymity. That is instead of guaranteeing that the keywords/queries/sessions of k *individuals* are indistinguishable in a search log they only assure that the keywords/queries/sessions associated with k different *user-IDs* are indistinguishable. These two guarantees are not the same since individuals can have multiple accounts or share accounts. An attacker can exploit this by creating multiple accounts and submitting the same fake queries from these accounts. It can happen that in a k-keyword/query/session-anonymous search log the keywords/queries/sessions of a user are only indistinguishable from $k-1$ fake keywords/queries/sessions submitted by an attacker. It is doubtful that this type of indistinguishability at the level of user-IDs is satisfactory.

3.2 Impossibility of Differential Privacy

In the following, we illustrate the infeasibility of differential privacy in search log publication. In particular, we show that, under realistic settings, no differentially private algorithm can produce a sanitized search log with reasonable utility (utility is measured as defined in Section 2.3.1 using our notion of accuracy). Our analysis is based on the following lemma.

Lemma 1. *For a set of U users, let S and S' be two search logs each containing at most m items from some domain D per user. Let $\mathcal{A}$ be an ϵ-differentially private algorithm that, given S, retains a very frequent item d in S with probability p. Then, given S', $\mathcal{A}$ retains d with probability at least $p/(e^{L_1(S,S')\cdot\epsilon/m})$, where $L_1(S,S') = \sum_{d\in D}|f_d(S) - f_d(S')|$ denotes the L_1 distance between S and S'.*

Lemma 1 follows directly from the definition of ϵ-differential privacy. Based on Lemma 1, we have the following theorem, which shows that any ϵ-differentially private algorithm that is accurate for very frequent items must be inaccurate for very infrequent items. The rationale is that, if given a search log S, an algorithm outputs one very frequent item d in S, then even if the input to the algorithm is a search log where d is very infrequent, the algorithm should still output d with a certain probability; otherwise, the algorithm cannot be differentially private.

Theorem 1. *Consider an accuracy constant c, a threshold τ, a slack ξ, and a very large domain $\mathcal{D}$ of size $\geq Um(\frac{2e^{2\epsilon(\tau+\xi)/m}}{c(\tau+\xi)} + \frac{1}{\tau-\xi+1})$, where m denotes the maximum number of items that a user may have in a search log. Let $\mathcal{A}$ be an ϵ-differentially private algorithm that is c-accurate (according to Definition 6) for the very frequent items. Then, for any input search log, the inaccuracy of $\mathcal{A}$ is greater than the inaccuracy of an algorithm that always outputs an empty set.*

Proof. Consider an ϵ-differentially private algorithm $\mathcal{A}'$ that is c-accurate for the very frequent items. Fix some input S. We are going to show that for each very infrequent item d in S the probability of outputting d is at least $c/(e^{\epsilon(\tau+\xi)/m})$. For each item, $d \in \mathcal{D}$ construct S'_d from S by changing $\tau+\xi$ of the items to d. That way d is very frequent (with frequency at least $\tau+\xi$) and $L_1(S,S'_d) \leq 2(\tau+\xi)$. By Definition 6, we have that

$$\Pr[d \in \mathcal{A}'(S'_d)] \geq c.$$

By Lemma 1, it follows that the probability of outputting d is at least $c/(e^{2\epsilon(\tau+\xi)/m})$ for any input database. This means that we can compute a lower bound on the inability to filter out the very infrequent items in S by summing up this probability over all possible values $d \in \mathcal{D}$ that are very infrequent in S. Note that there are at least $\mathcal{D} - \frac{Um}{\tau-\xi+1}$ many very infrequent items in S. Therefore, the inability to filter out the very infrequent items is at least $(|\mathcal{D}| - \frac{Um}{\tau-\xi+1})c/(e^{2\epsilon(\tau+\xi)/m})$. For large domains of size at least $Um(\frac{2e^{2\epsilon(\tau+\xi)/m}}{c(\tau+\xi)} + \frac{1}{\tau-\xi+1})$, the inaccuracy is at least $\frac{2Um}{\tau+\xi}$ which is greater than the inaccuracy of the baseline. □

To illustrate Theorem 1, let us consider a search log S where each query contains at most three keywords selected from a limited vocabulary of 900,000 words. Let D be the domain of the consecutive query pairs in S. We have $|D| = 5.3 \times 10^{35}$. Consider the following setting of the parameters $\tau+\xi = 50, m = 10, U = 1{,}000{,}000, \epsilon = 1$ that is typical practice. By Theorem 1, if an ϵ-differentially private algorithm $\mathcal{A}$ is 0.01-accurate for very frequent query pairs, then, in terms of overall inaccuracy (for both very frequent and very infrequent query pairs), $\mathcal{A}$ must be inferior to an algorithm that always outputs an empty set. In other words,

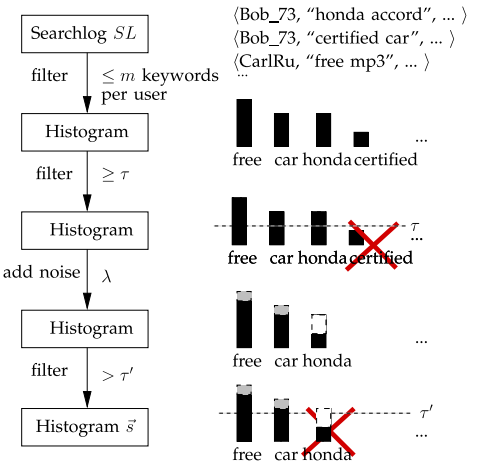


Fig. 1. Privacy-preserving algorithm.

no differentially private algorithm can be accurate for both very frequent and very infrequent query pairs.

4 Achieving Privacy

In this section, we introduce a search log publishing algorithm called ZEALOUS that has been independently developed by Korolova et al. [17] and us [10]. ZEALOUS ensures probabilistic differential privacy, and it follows a simple two-phase framework. In the first phase, ZEALOUS generates a histogram of items in the input search log, and then removes from the histogram the items with frequencies below a threshold. In the second phase, ZEALOUS adds noise to the histogram counts, and eliminates the items whose noisy frequencies are smaller than another threshold. The resulting histogram (referred to as the *sanitized* histogram) is then returned as the output. Fig. 1 depicts the steps of ZEALOUS.

Algorithm ZEALOUS for Publishing Frequent Items of a Search Log

Input: Search log S, positive numbers m, λ, τ, τ'

1. For each user u select a set s_u of up to m distinct items from u's search history in S.[3]
2. Based on the selected items, create a histogram consisting of pairs (k, c_k), where k denotes an item and c_k denotes the number of users u that have k in their search history s_u. We call this histogram the *original* histogram.
3. Delete from the histogram the pairs (k, c_k) with count c_k smaller than τ.
4. For each pair (k, c_k) in the histogram, sample a random number η_k from the Laplace distribution $\text{Lap}(\lambda)$,[4] and add η_k to the count c_k, resulting in a noisy count: $\tilde{c}_k \leftarrow c_k + \eta_k$.
5. Delete from the histogram the pairs $(k, \tilde{c}_k)$ with noisy counts $\tilde{c}_k \leq \tau'$.
6. Publish the remaining items and their noisy counts.

3. These items can be selected in various ways as long as the selection criteria are not based on the data. Random selection is one candidate.

4. The Laplace distribution with scale parameter λ has the probability density function $\frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}}$.

To understand the purpose of the various steps one has to keep in mind the privacy guarantee we would like to achieve. Steps 1, 2, and 4 of the algorithm are fairly standard. It is known that adding Laplacian noise to histogram counts achieves ϵ-differential privacy [9]. However, the previous section explained that these steps alone result in poor utility because for large domains many infrequent items will have high noisy counts. To deal better with large domains, we restrict the histogram to items with counts at least τ in Step 2. This restriction leaks information and thus the output after Step 4 is not ϵ-differentially private. One can show that it is not even (ϵ, δ)-probabilistic differentially private (for $\delta < 1/2$). Step 5 disguises the information leaked in Step 3 in order to achieve probabilistic differential privacy.

In what follows, we will investigate the theoretical performance of ZEALOUS in terms of both privacy and utility. Sections 4.1 and 4.2 discuss the privacy guarantees of ZEALOUS with respect to (ϵ, δ)-indistinguishability and (ϵ, δ)-probabilistic differential privacy, respectively. Section 4.3 presents a quantitative analysis of the privacy protection offered by ZEALOUS. Sections 4.4 and 4.5 analyze the utility guarantees of ZEALOUS.

4.1 Indistinguishability Analysis

Theorem 2 states how the parameters of ZEALOUS can be set to obtain a sanitized histogram that provides (ϵ', δ')-indistinguishability.

Theorem 2 ([17]). *Given a search log S and positive numbers m, τ, τ', and λ, ZEALOUS achieves (ϵ', δ')-indistinguishability, if*

$$\lambda \geq 2m/\epsilon', \text{ and} \tag{1}$$

$$\tau = 1, \text{ and} \tag{2}$$

$$\tau' \geq m\left(1 - \frac{\log(\frac{2\delta'}{m})}{\epsilon'}\right). \tag{3}$$

To publish not only frequent queries but also their clicks, Korolova et al. [17] suggest to first determine the frequent queries and then publish noisy counts of the clicks to their top-100 ranked documents. In particular, if we use ZEALOUS to publish frequent queries in a manner that achieves (ϵ', δ')-indistinguishability, we can also publish the noisy click distributions of the top-100 ranked documents for each of the frequent queries, by simply adding Laplacian noise to the click counts with scale $2m/\epsilon'$. Together the sanitized query and click histogram achieves $(2\epsilon', \delta')$-indistinguishability.

4.2 Probabilistic Differential Privacy Analysis

Given values for ϵ, δ, τ, and m, the following theorem tells us how to set the parameters λ and τ' to ensure that ZEALOUS achieves (ϵ, δ)-probabilistic differential privacy.

Theorem 3. *Given a search log S and positive numbers m, τ, τ', and λ, ZEALOUS achieves (ϵ, δ)-probabilistic differential privacy, if*

$$\lambda \geq 2m/\epsilon, \text{ and} \tag{4}$$

TABLE 1
(ϵ', δ')-Indistinguishability versus (ϵ, δ)-Probabilistic Differential Privacy

Privacy Guarantee	$\tau' = 100$	$\tau' = 200$
$\lambda = 1$ $(\epsilon, \epsilon' = 10)$	$\delta = 1.3 \times 10^{-37}$	$\delta = 4.7 \times 10^{-81}$
	$\delta' = 1.4 \times 10^{-41}$	$\delta' = 5.2 \times 10^{-85}$
$\lambda = 5$ $(\epsilon, \epsilon' = 2)$	$\delta = 3.2 \times 10^{-3}$	$\delta = 6.5 \times 10^{-12}$
	$\delta' = 1.4 \times 10^{-8}$	$\delta' = 2.9 \times 10^{-17}$

$U = 500k$, $m = 5$.

$$\tau' - \tau \geq \max\left(-\lambda \ln\left(2 - 2e^{-\frac{1}{\lambda}}\right), -\lambda \ln\left(\frac{2\delta}{U \cdot m/\tau}\right)\right), \quad (5)$$

where U denotes the number of users in S.

The proof of Theorem 3 can be found in the Appendix A.1, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.26.

4.3 Quantitative Comparison of Probabilistic Differential Privacy and Indistinguishability for ZEALOUS

In Table 1, we illustrate the levels of (ϵ', δ')-indistinguishability and (ϵ, δ)-probabilistic differential privacy achieved by ZEALOUS for various noise and threshold parameters. We fix the number of users to $U = 500k$, and the maximum number of items from a user to $m = 5$, which is a typical setting that will be explored in our experiments. Table 1 shows the tradeoff between utility and privacy: A larger λ results in a greater amount of noise in the sanitized search log (i.e., decreased data utility), but it also leads to smaller ϵ and ϵ' (i.e., stronger privacy guarantee). Similarly, when τ' increases, the sanitized search log provides less utility (since fewer items are published) but a higher level of privacy protection (as δ and δ' decreases).

Interestingly, given λ and τ', we always have $\delta > \delta'$. This is due to the fact that (ϵ, δ)-probabilistic differential privacy is a stronger privacy guarantee than (ϵ', δ')-indistinguishability, as will be discussed in Section 5.

4.4 Utility Analysis

Next, we analyze the utility guarantee of ZEALOUS in terms of its accuracy (as defined in Section 2.3.1).

Theorem 4. *Given parameters $\tau = \tau^* - \xi, \tau' = \tau^* + \xi$, noise scale λ, and a search log S, the inaccuracy of ZEALOUS with slack ξ equals*

$$\sum_{d: f_d(S) > \tau + \xi} 1/2e^{-2\xi/\lambda} + \sum_{d \in D: f_d(S) < \tau - \xi} 0.$$

In particular, this means that ZEALOUS is $(1 - 1/2e^{-\frac{\xi}{\lambda}})$-accurate for the very frequent items (of frequency $\geq \tau^* + \xi$) and it provides perfect accuracy for the very infrequent items (of frequency $< \tau^* - \xi$).

Proof. It is easy to see that ZEALOUS provides perfect accuracy of filtering out infrequent items. Moreover, the probability of outputting a very frequent item is at least

$$1 - 1/2e^{-\frac{\xi}{\lambda}},$$

which is the probability that the $\mathrm{Lap}(\lambda)$-distributed noise that is added to the count is at least $-\xi$ so that a very frequent item with count at least $\tau + \xi$ remains in the output of the algorithm. This probability is at least $1/2$. All in all, it has higher accuracy than the baseline algorithm on all inputs with at least one very frequent item. □

4.5 Separation Result

Combining the analysis in Sections 3.2 and 4.4, we obtain the following separation result between ϵ-differential privacy and (ϵ, δ)- probabilistic differential privacy.

Theorem 5 (Separation result). *Our (ϵ, δ)-probabilistic differentially private algorithm ZEALOUS is able to retain frequent items with probability at least $1/2$ while filtering out all infrequent items. On the other hand, for any ϵ-differentially private algorithm that can retain frequent items with nonzero probability (independent of the input database), its inaccuracy for large item domains is larger than an algorithm that always outputs an empty set.*

5 Comparing Indistinguishability with Probabilistic Differential Privacy

In this section, we study the relationship between (ϵ, δ)-probabilistic differential privacy and (ϵ', δ')-indistinguishability. First, we will prove that probabilistic differential privacy implies indistinguishability. Then, we will show that the converse is not true. We show that there exists an algorithm that is (ϵ', δ')-indistinguishable yet blatantly non-ϵ-differentially private (and also not (ϵ, δ)-probabilistic differentially private for any value of ϵ and $\delta < 1$). This fact might convince a data publisher to strongly prefer an algorithm that achieves (ϵ, δ)-probabilistic differential privacy over one that is only known to achieve (ϵ', δ')-indistinguishability. It also might convince researchers to analyze the probabilistic privacy guarantee of algorithms that are only known to be indistinguishable as in [8] or [22].

First, we show that our definition implies (ϵ, δ)-indistinguishability.

Proposition 1. *If an algorithm $\mathcal{A}$ is (ϵ, δ)-probabilistic differentially private then it is also (ϵ, δ)-indistinguishable.*

The proof of Proposition 1 can be found in the Appendix A.2, available in the online supplemental material. The converse of Proposition 1 does not hold. In particular, there exists an algorithm that is (ϵ', δ')-indistinguishable but not (ϵ, δ)-probabilistic differentially private for any choice of ϵ and $\delta < 1$, as illustrated in the following example.

Example 1. Consider the following algorithm that takes as input a search log S with search histories of U users.

Algorithm $\hat{\mathcal{A}}$
Input: Search log $S \in \mathcal{D}^U$
1. Sample uniformly at random a single search history from the set of all histories excluding the first user's search history.
2. Return this search history.

The following proposition analyzes the privacy of Algorithm $\hat{\mathcal{A}}$.

Proposition 2. *For any finite domain of search histories $\mathcal{D}$, Algorithm $\hat{\mathcal{A}}$ is $(\epsilon', 1/(|\mathcal{D}| - 1))$-indistinguishable for all $\epsilon' > 0$ on inputs from $\mathcal{D}^U$.*

TABLE 2
τ' as a Function of τ for $m = 2$, $\epsilon = 1$, $\delta = 0.01$

τ	1	3	4	5	7	9
τ'	81.1205	78.7260	78.5753	78.6827	79.3368	80.3316

The proof can be found in the Appendix A.3, available in the online supplemental material. The next proposition shows that every single output of the algorithm constitutes a privacy breach.

Proposition 3. *For any search log S, the output of Algorithm $\hat{\mathcal{A}}$ constitutes a privacy breach according to ϵ-differentially privacy for any value of ϵ.*

Proof. Fix an input S and an output O that is different from the search history of the first user. Consider the input S' differing from S only in the first user history, where $S'_1 = O$. Here,

$$1/(|\mathcal{D}| - 1) = \Pr[\mathcal{A}(S) = O] \not\leq e^{\epsilon} \Pr[\mathcal{A}(S') = O] = 0.$$

Thus, the output S breaches the privacy of the first user according to ϵ-differentially privacy. □

Corollary 1. *Algorithm $\hat{\mathcal{A}}$ is $(\epsilon', 1/(|\mathcal{D}| - 1))$-indistinguishable for all $\epsilon' > 0$. But it is not (ϵ, δ)-probabilistic differentially private for any ϵ and any $\delta < 1$.*

By Corollary 1, an algorithm that is (ϵ', δ')-indistinguishable may not achieve any form of (ϵ, δ)-probabilistic differential privacy, even if δ' is set to an extremely small value of $1/(|\mathcal{D}| - 1)$. This illustrates the significant gap between (ϵ', δ')-indistinguishable and (ϵ, δ)-probabilistic differential privacy.

6 Choosing Parameters

Apart from the privacy parameters ϵ and δ, ZEALOUS requires the data publisher to specify two more parameters: τ, the first threshold used to eliminate keywords with low counts (Step 3), and m, the number of contributions per user. These parameters affect both the noise added to each count as well as the second threshold τ'. Before we discuss the choice of these parameters, we explain the general setup of our experiments.

Data. In our experiments, we work with a search log of user queries from a major search engine collected from 500,000 users over a period of one month. This search log contains about one million distinct keywords, three million distinct queries, three million distinct query pairs, and 4.5 million distinct clicks.

Privacy parameters. In all experiments, we set $\delta = 0.001$. Thus, the probability that the output of ZEALOUS could breach the privacy of any user is very small. We explore different levels of (ϵ, δ)-probabilistic differential privacy by varying ϵ.

6.1 Choosing Threshold τ

We would like to retain as much information as possible in the published search log. A smaller value for τ' immediately leads to a histogram with higher utility because fewer items and their noisy counts are filtered out in the last step of ZEALOUS. Thus, if we choose τ in a way that minimizes τ' we maximize the utility of the resulting histogram. Interestingly, choosing $\tau = 1$ does not necessarily minimize the value of τ'. Table 2 presents the value of τ' for different values of τ for $m = 2$ and $\epsilon = 1$. Table 2 shows that for our parameter settings τ' is minimized when $\tau = 4$. We can show the following optimality result which tells us how to choose τ optimally in order to maximize utility.

Proposition 4. *For a fixed ϵ, δ and m choosing $\tau = \lceil 2m/\epsilon \rceil$ minimizes the value of τ'.*

The proof follows from taking the derivative of τ' as a function of τ (based on (5)) to determine its minimum.

TABLE 3
Coverage

(A) Distinct item counts with different m.

m	1	4	8	20	40
keywords	6667	6043	5372	4062	2964
queries	3334	2087	1440	751	408
clicks	2813	1576	1001	486	246
query pairs	331	169	100	40	13

(B) Total item counts $\times 10^3$ with different m.

m	1	4	8	20	40
keywords	329	1157	1894	3106	3871
queries	147	314	402	464	439
clicks	118	234	286	317	290
query pairs	8	14	15	12	7

6.2 Choosing the Number of Contributions m

Proposition 4 tells us how to set τ in order to maximize utility. Next, we will discuss how to set m optimally. We will do so by studying the effect of varying m on the coverage and the precision of the top-j most frequent items in the sanitized histogram. The *top-j coverage of a sanitized search log* is defined as the fraction of distinct items among the top-j most frequent items in the original search log that also appear in the sanitized search log. The *top-j precision of a sanitized search log* is defined as the distance between the relative frequencies in the original search log versus the sanitized search log for the top-j most frequent items. In particular, we study two distance metrics between the relative frequencies: the average L-1 distance and the KL-divergence.

As a first study of coverage, Table 3 shows the number of distinct items (recall that items can be keywords, queries, query pairs, or clicks) in the sanitized search log as m increases. We observe that coverage decreases as we increase m. Moreover, the decrease in the number of published items is more dramatic for larger domains than for smaller domains. The number of distinct keywords decreases by 55 percent while at the same time the number of distinct query pairs decreases by 96 percent as we increase m from 1 to 40. This trend has two reasons. First, from Theorem 3 and Proposition 4, we see that threshold τ' increases superlinearly in m. Second, as m increases the number of keywords contributed by the users increases only sublinearly in m; fewer users are able to supply m items for increasing values of m. Hence, fewer items pass the threshold τ' as m increases. The reduction is larger for query pairs than for keywords,

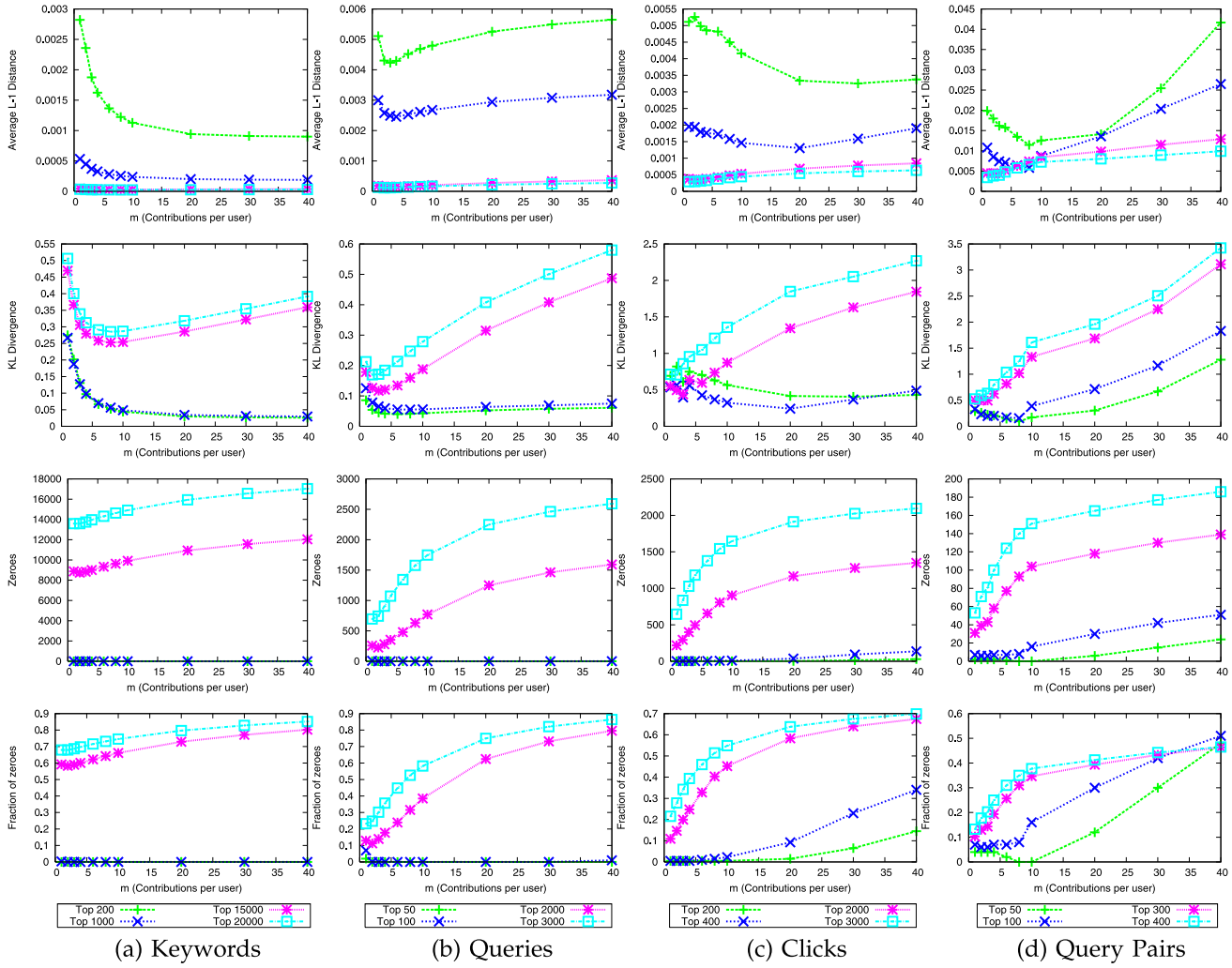
TABLE 4
Average Number of Items per User in the Original Search Log

	keywords	queries	click	query pairs
avg items/user	56	20	14	7

because the average number of query pairs per user is smaller than the average number of keywords per user in the original search log (shown in Table 4).

To understand how m affects precision, we measure the total sum of the counts in the sanitized histogram as we increase m in Table 3. Higher total counts offer the possibility to match the original distribution at a finer granularity. We observe that as we increase m, the total counts increase until a tipping point is reached after which they start decreasing again. This effect is as expected for the following reason: As m increases, each user contributes more items, which leads to higher counts in the sanitized histogram. However, the total count increases only sub-linearly with m (and even decreases) due to the reduction in coverage we discussed above. We found that the tipping point where the total count starts to decrease corresponds approximately to the average number of items contributed by each user in the original search log (shown in Table 4). This suggests that we should choose m to be smaller than the average number of items, because it offers better coverage, higher total counts, and reduces the noise compared to higher values of m.

Let us take a closer look at the precision and coverage of the histograms of the various domains in Figs. 2 and 3. In Fig. 2, we vary m between 1 and 40. Each curve plots the precision or coverage of the sanitized search log at various values of the top-j parameter in comparison to the original search log. We vary the top-j parameter but never choose it higher than the number of distinct items in the original search log for the various domains. The first two rows plot precision curves for the average L-1 distance (first row) and the KL-divergence (second row) of the relative frequencies. The lower two rows plot the coverage curves, i.e., the total number of top-j items (third row) and the relative number of top-j items (fourth row) in the original search log that do not appear in sanitized search log. First, observe that the coverage decreases as m increases, which confirms our discussion about the number of distinct items. Moreover, we see that the coverage gets worse for increasing values of the top-j parameter. This illustrates that ZEALOUS gives better utility for the more frequent items. Second, note that for small



(a) Keywords (b) Queries (c) Clicks (d) Query Pairs

Fig. 2. Effect on statistics by varying m for several values of top-j. (a) Keywords. (b) Queries. (c) Clicks. (d) Query pairs.

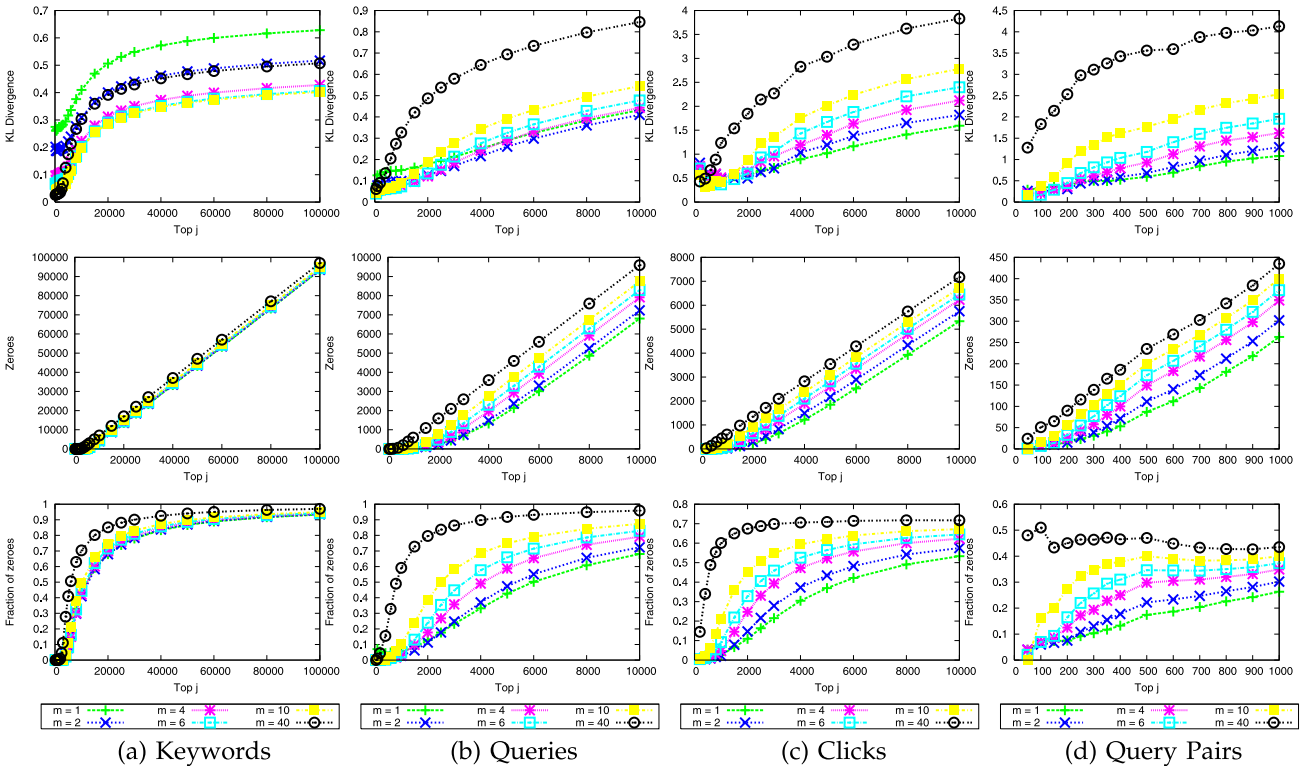


Fig. 3. Effect on statistics of varying j in top-j for different values of m. (a) Keywords. (b) Queries. (c) Clicks. (d) Query pairs.

values of the top-j parameter, values of $m > 1$ give better precision. However, when the top-j parameter is increased, $m = 1$ gives better precision because the precision of the top-j values degrades due to items no longer appearing in the sanitized search log due to the increased cutoffs.

Fig. 3 shows the same statistics varying the top-j parameter on the x-axis. Each curve plots the precision for $m = 1, 2, 4, 8, 10, 40$, respectively. Note that $m = 1$ does not always give the best precision; for keywords, $m = 8$ has the lowest KL-divergence, and for queries, $m = 2$ has the lowest KL-divergence. As we can see from these results, there are two "regimes" for setting the value of m. If we are mainly interested in coverage, then m should be set to 1. However, if we are only interested in a few top-j items then we can increase precision by choosing a larger value for m, and in this case we recommend the average number of items per user.

We will see this dichotomy again in our real applications of search log analysis: The index caching application does not require high coverage because of its storage restriction. However, high precision of the top-j most frequent items is necessary to determine which of them to keep in memory. On the other hand, in order to generate many query substitutions, a larger number of distinct queries and query pairs is required. Thus, m should be set to a large value for index caching and to a small value for query substitution.

7 Application-Oriented Evaluation

In this section, we show the results of an application-oriented evaluation of privacy-preserving search logs generated by ZEALOUS in comparison to a k-anonymous search log and the original search log as points of comparison. Note that our utility evaluation does not determine the "better" algorithm since when choosing an algorithm in practice one has to consider both the utility and disclosure limitation guarantees of an algorithm. Our results show the "price" that we have to pay (in terms of decreased utility) when we give the stronger guarantees of (probabilistic versions of) differential privacy as opposed to k-anonymity.

Algorithms. We experimentally compare the utility of ZEALOUS against a representative k-anonymity algorithm by Adar for publishing search logs [1]. Recall that Adar's Algorithm creates a k-query anonymous search log as follows: First all queries that are posed by fewer than k distinct users are eliminated. Then, histograms of keywords, queries, and query pairs from the k-query anonymous search log are computed. ZEALOUS can be used to achieve (ϵ', δ')-indistinguishability as well as (ϵ, δ)-probabilistic differential privacy. For the ease of presentation, we only show results with probabilistic differential privacy; using Theorems 2 and 3, it is straightforward to compute the corresponding indistinguishability guarantee. For brevity, we refer to the (ϵ, δ)-probabilistic differentially private algorithm as ϵ-Differential in the figures.

Evaluation metrics. We evaluate the performance of the algorithms in two ways. First, we measure how well the output of the algorithms preserves selected statistics of the original search log. Second, we pick two real applications from the information retrieval community to evaluate the utility of ZEALOUS: Index caching as a representative application for search performance, and query substitution as a representative application for search quality. Evaluating the output of ZEALOUS with these two applications will help us to fully understand the performance of ZEALOUS in an application context. We first describe our

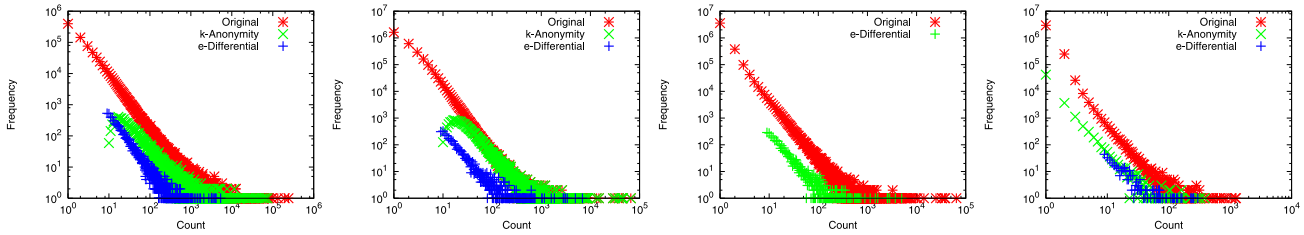


Fig. 4. Distributions of counts in the histograms. (a) Keyword counts. (b) Query Counts. (c) Click Counts. (d) Query pair counts.

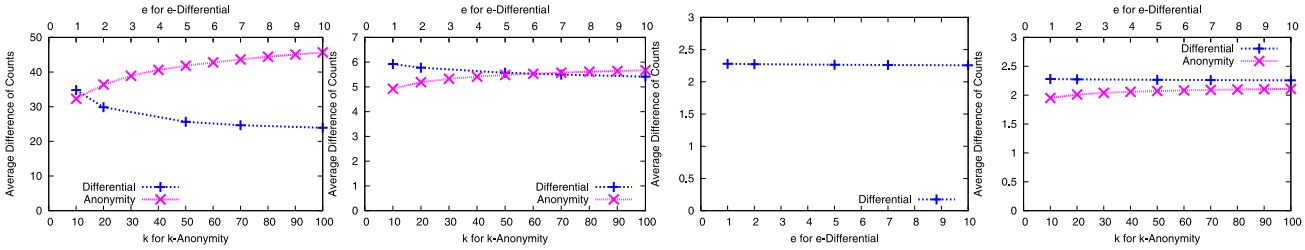


Fig. 5. Average difference between counts in the original histogram and the probabilistic differential privacy-preserving histogram, and the anonymous histogram for varying privacy/anonymity parameters ϵ and k. Parameter m is fixed to 1. (a) Keywords. (b) Queries. (c) Clicks. (d) Query pairs.

utility evaluation with statistics in Section 7.1 and then our evaluation with real applications in Sections 7.2 and 7.3.

7.1 General Statistics

We explore different statistics that measure the difference of sanitized histograms to the histograms computed using the original search log. We analyze the histograms of keywords, queries, and query pairs for both sanitization methods. For clicks, we only consider ZEALOUS histograms since a k-query anonymous search log is not designed to publish click data.

In our first experiment, we compare the distribution of the counts in the histograms. Note that a k-query anonymous search log will never have query and keyword counts below k, and similarly a ZEALOUS histogram will never have counts below τ'. We choose $\epsilon = 5, m = 1$ for which threshold $\tau' \approx 10$. Therefore, we deliberately set $k = 10$ such that $k \approx \tau'$.

Fig. 4 shows the distribution of the counts in the histograms on a log-log scale. Recall that the k-query anonymous search log does not contain any click data, and thus it does not appear in Fig. 4c. We see that the power-law shape of the distribution is well preserved. However, the total frequencies are lower for the sanitized search logs than the frequencies in the original histogram because the algorithms filter out a large number of items. We also see the cutoffs created by k and τ'. We observe that as the domain increases from keywords to clicks and query pairs, the number of items that are not frequent in the original search log increases. For example, the number of clicks with count equal to one is an order of magnitude larger than the number of keywords with count equal to one.

While the shape of the count distribution is well preserved, we would also like to know whether the counts of frequent keywords, queries, query pairs, and clicks are also preserved and what impact the privacy parameters ϵ and the anonymity parameter k have. Fig. 5 shows the average differences to the counts in the original histogram. We scaled up the counts in sanitized histograms by a common factor so that the total counts were equal to the total counts of the original histogram, then we calculated the average difference between the counts. The average is taken over all keywords that have nonzero count in the original search log. As such, this metric takes both coverage and precision into account.

As expected, with increasing ϵ the average difference decreases, since the noise added to each count decreases. Similarly, by decreasing k the accuracy increases because more queries will pass the threshold. Fig. 5 shows that the average difference is comparable for the k-anonymous histogram and the output of ZEALOUS. Note that the output of ZEALOUS for keywords is more accurate than a k-anonymous histogram for all values of $\epsilon > 2$. For queries, we obtain roughly the same average difference for $k = 60$ and $\epsilon = 6$. For query pairs, the k-query anonymous histogram provides better utility.

We also computed other metrics such as the root-mean-square value of the differences and the total variation difference; they all reveal similar qualitative trends. Despite the fact that ZEALOUS disregards many search log records (by throwing out all but m contributions per user and by throwing out low-frequent counts), ZEALOUS is able to preserve the overall distribution well.

7.2 Index Caching

In the index caching problem, we aim to cache in-memory a set of posting lists that maximizes the hit probability over all keywords (see Section 2.3.2). In our experiments, we use an improved version of the algorithm developed by Baeza-Yates to decide which posting lists should be kept in memory [2]. Our algorithm first assigns each keyword a score, which equals its frequency in the search log divided by the number of documents that contain the keyword. Keywords are chosen using a greedy bin-packing strategy

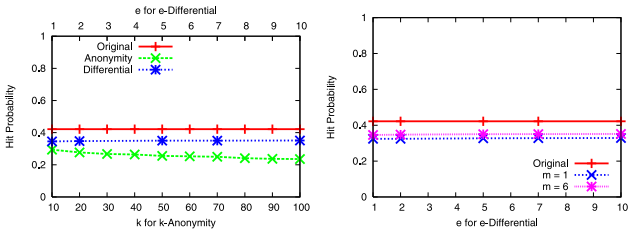


Fig. 6. Hit probabilities.

where we sequentially add posting lists from the keywords with the highest score until the memory is filled. In our experiments, we fixed the memory size to be 1 GB, and each document posting to be 8 Bytes (other parameters give comparable results). Our inverted index stores the document posting list for each keyword sorted according to their relevance which allows to retrieve the documents in the order of their relevance. We truncate this list in memory to contain at most 200,000 documents. Hence, for an incoming query the search engine retrieves the posting list for each keyword in the query either from memory or from disk. If the intersection of the posting lists happens to be empty, then less relevant documents are retrieved from disk for those keywords for which only the truncated posting list is kept on memory.

Fig. 6a shows the hit probabilities of the inverted index constructed using the original search log, the k-anonymous search log, and the ZEALOUS histogram (for $m = 6$) with our greedy approximation algorithm. We observe that our ZEALOUS histogram achieves better utility than the k-query anonymous search log for a range of parameters. We note that the utility suffers only marginally when increasing the privacy parameter or the anonymity parameter (at least in the range that we have considered). This can be explained by the fact that it requires only a few very frequent keywords to achieve a high hit probability. Keywords with a big positive impact on the hit probability are less likely to be filtered out by ZEALOUS than keywords with a small positive impact. This explains the marginal decrease in utility for increased privacy.

As a last experiment, we study the effect of varying m on the hit probability in Fig. 6b. We observe that the hit probability for $m = 6$ is above 0.36 whereas the hit probability for $m = 1$ is less than 0.33. As discussed a higher value for m increases the accuracy, but reduces the coverage. Index caching really requires roughly the top 85 most frequent keywords that are still covered when setting $m = 6$. We also experimented with higher values of m and observed that the hit probability decreases at some point.

7.3 Query Substitution

Algorithms for query substitution examine query pairs to learn how users rephrase queries. We use an algorithm developed by Jones et al. in which related queries for a query are identified in two steps [15]. First, the query is partitioned into subsets of keywords, called *phrases*, based on their mutual information. Next, for each phrase, candidate query substitutions are determined based on the distribution of queries.

We run this algorithm to generate ranked substitution on the sanitized search logs. We then compare these rankings with the rankings produced by the original search log which serve as ground truth. To measure the quality of the query substitutions, we compute the precision/recall, mean average precision (MAP) and normalized discounted cumulative gain (NDG) of the top-j suggestions for each query; let us define these metrics next.

Consider a query q and its list of top-j ranked substitutions $q'_0, \ldots, q'_{j-1}$ computed based on a sanitized search log. We compare this ranking against the top-j ranked substitutions $q_0, \ldots, q_{j-1}$ computed based on the original search log as follows. The *precision* of a query q is the fraction of substitutions from the sanitized search log that are also contained in our ground truth ranking:

$$\text{Precision } (q) = \frac{|\{q_0, \ldots, q_{j-1}\} \cap \{q'_0, \ldots, q'_{j-1}\}|}{|\{q'_0, \ldots, q'_{j-1}\}|}.$$

Note that the number of items in the ranking for a query q can be less than j. The *recall* of a query q is the fraction of substitutions in our ground truth that are contained in the substitutions from the sanitized search log:

$$\text{Recall } (q) = \frac{|\{q_0, \ldots, q_{j-1}\} \cap \{q'_0, \ldots, q'_{j-1}\}|}{|\{q_0, \ldots, q_{j-1}\}|}.$$

MAP measures the precision of the ranked items for a query as the ratio of true rank and assigned rank:

$$\text{MAP } (q) = \sum_{i=0}^{j-1} \frac{i+1}{\text{rank of } q_i \text{ in } [q'_0, \ldots, q'_{j-1}] + 1},$$

where the rank of q_i is zero in case it is not contained in the list $[q'_0, \ldots, q'_{j-1}]$; otherwise, it is i', s.t. $q_i = q'_{i'}$.

Our last metric called NDCG measures how the relevant substitutions are placed in the ranking list. It not only compare the ranks of a substitution in the two rankings, but

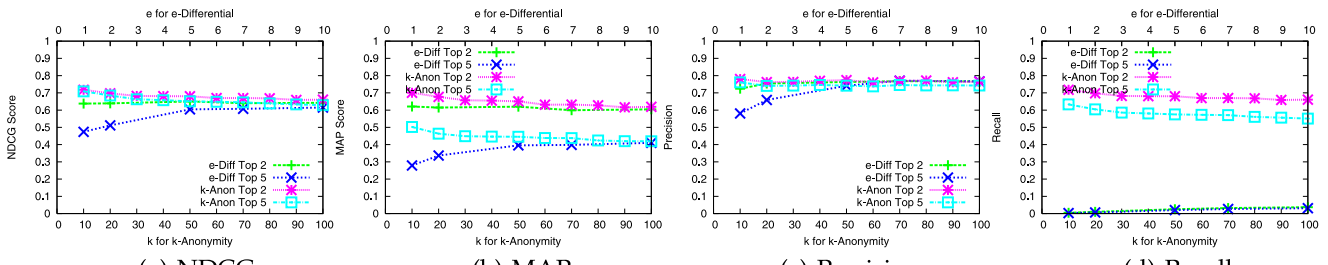


Fig. 7. Quality of the query substitutions of the privacy-preserving histogram, and the anonymous search log. (a) NDCG. (b) MAP. (c) Precision. (d) Recall.

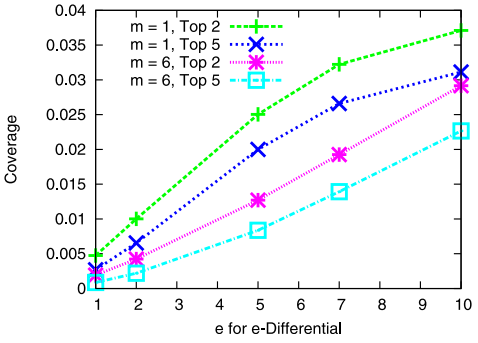


Fig. 8. Coverage of the privacy-preserving histograms for $m = 1$ and $m = 6$.

also penalizes highly relevant substitutions according to $[q_0, \ldots, q_{j-1}]$ that have a very low rank in $[q'_0, \ldots, q'_{j-1}]$. Moreover, it takes the length of the actual lists into consideration. We refer the reader to the paper by Chakrabarti et al. [7] for details on NDCG.

The discussed metrics compare rankings for one query. To compare the utility of our algorithms, we average over all queries. For coverage, we average over all queries for which the original search log produces substitutions. For all other metrics that try to capture the precision of a ranking, we average only over the queries for which the sanitized search logs produce substitutions. We generated query substitution only for the 100,000 most frequent queries of the original search log since the substitution algorithm only works well given enough information about a query.

In Fig. 7, we vary k and ϵ for $m = 1$ and we draw the utility curves for top-j for $j = 2$ and $j = 5$. We observe that varying ϵ and k has hardly any influence on performance. On all precision measures, ZEALOUS provides utility comparable to k-query-anonymity. However, the coverage provided by ZEALOUS is not good. This is because the computation of query substitutions relies not only on the frequent query pairs but also on the count of phrase pairs which record for two sets of keywords how often a query containing the first set was followed by another query containing the second set. Thus, a phrase pair can have a high frequency even though all query pairs it is contained in have very low frequency. ZEALOUS filters out these low-frequency query pairs and thus loses many frequent phrase pairs.

As a last experiment, we study the effect of increasing m for query substitutions. Fig. 8 plots the average coverage of the top-2 and top-5 substitutions produced by ZEALOUS for $m = 1$ and $m = 6$ for various values of ϵ. It is clear that across the board larger values of m lead to smaller coverage, thus confirming our intuition outlined the previous section.

8 Related Work

Related work on anonymity in search logs [1], [12], [21], [13] is discussed in Section 3.1.

More recently, there has been work on privacy in search logs. Korolova et al. [17] propose the same basic algorithm that we propose in [10] and review in Section 4.[5] They show (ϵ', δ')-indistinguishability of the algorithm whereas we show (ϵ, δ)-probabilistic differential privacy of the algorithm which is a strictly stronger guarantee (see Section 5). One difference is that our algorithm has two thresholds τ, τ' as opposed to one and we explain how to set threshold τ optimally. Korolova et al. [17] set $\tau = 1$ (which is not the optimal choice in many cases). Our experiments augment and extend the experiments of Korolova et al. [17]. We illustrate the tradeoff of setting the number of contributions m for various domains and statistics including L1-distance and KL divergence which extends [17] greatly. Our application-oriented evaluation considers different applications. We compare the performance of ZEALOUS to that of k-query anonymity and observe that the loss in utility is comparable for anonymity and privacy while anonymity offers a much weaker guarantee.

5. In order to improve utility of the algorithm as stated in [17], we suggest to first filter out infrequent keywords using the 2-threshold approach of ZEALOUS and then publish noise counts of queries consisting of up to three frequent keywords and the clicks of their top ranked documents.

9 Beyond Search Logs

While the main focus of this paper are search logs, our results apply to other scenarios as well. For example, consider a retailer who collects customer transactions. Each transaction consists of a basket of products together with their prices, and a time stamp. In this case, ZEALOUS can be applied to publish frequently purchased products or sets of products. This information can also be used in a recommender system or in a market basket analysis to decide on the goods and promotions in a store [11]. Another example concerns monitoring the health of patients. Each time a patient sees a doctor, the doctor records the diseases of the patient and the suggested treatment. It would be interesting to publish frequent combinations of diseases.

All of our results apply to the more general problem of publishing frequent items/item sets/consecutive item sets. Existing work on publishing frequent item sets often only tries to achieve anonymity or makes strong assumptions about the background knowledge of an attacker, see, for example, some of the references in the survey by Luo et al. [19].

10 Conclusions

This paper contains a comparative study about publishing frequent keywords, queries, and clicks in search logs. We compare the disclosure limitation guarantees and the theoretical and practical utility of various approaches. Our comparison includes earlier work on anonymity and (ϵ', δ')-indistinguishability and our proposed solution to achieve (ϵ, δ)-probabilistic differential privacy in search logs. In our comparison, we revealed interesting relationships between indistinguishability and probabilistic differential privacy which might be of independent interest. Our results (positive as well as negative) can be applied more generally to the problem of publishing frequent items or item sets.

A topic of future work is the development of algorithms to release useful information about *infrequent* keywords, queries, and clicks in a search log while preserving user privacy.

Acknowledgments

The authors would like to thank their colleagues Filip Radlinski and Yisong Yue for helpful discussions about the usage of search logs.

REFERENCES

[1] E. Adar, "User 4xxxxx9: Anonymizing Query Logs," *Proc. World Wide Web (WWW) Workshop Query Log Analysis,* 2007.

[2] R. Baeza-Yates, "Web Usage Mining in Search Engines," *Web Mining: Applications and Techniques,* Idea Group, 2004.

[3] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar, "Privacy Accuracy and Consistency Too: A Holistic Solution to Contingency Table Release," *Proc. ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS),* 2007.

[4] M. Barbaro and T. Zeller, "A Face is Exposed for AOL Searcher No. 4417749," New York Times, http://www.nytimes.com/2006/08/09/technology/09aol.html?ex=1312776000en=f6f61949c6da4d38ei=5090, 2006.

[5] A. Blum, K. Ligett, and A. Roth, "A Learning Theory Approach to Non-Interactive Database Privacy," *Proc. 40th Ann. ACM Symp. Theory of Computing (STOC),* pp. 609-618, 2008.

[6] J. Brickell and V. Shmatikov, "The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing," *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD),* 2008.

[7] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya, "Structured Learning for Non-Smooth Ranking Losses," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD),* pp. 88-96, 2008.

[8] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our Data Ourselves: Privacy via Distributed Noise Generation," *Proc. Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT),* 2006.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," *Proc. Theory of Cryptography Conf. (TCC),* 2006.

[10] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, "Privacy in Search Logs," *CoRR,* abs/0904.0682v2, 2009.

[11] J. Han and M. Kamber, *Data Mining: Concepts and Techniques,* first ed. Morgan Kaufmann, Sept. 2000.

[12] Y. He and J.F. Naughton, "Anonymization of Set-Valued Data via Top-Down, Local Generalization," *Proc. VLDB Endowment,* vol. 2, no. 1, pp. 934-945, 2009.

[13] Y. Hong, X. He, J. Vaidya, N. Adam, and V. Atluri, "Effective Anonymization of Query Logs," *Proc. ACM Conf. Information and Knowledge Management (CIKM),* 2009.

[14] R. Jones, R. Kumar, B. Pang, and A. Tomkins, "I Know What You Did Last Summer: Query Logs and User Privacy," *Proc. ACM Conf. Information and Knowledge Management (CIKM),* 2007.

[15] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating Query Substitutions," *Proc. 15th Int'l Conf. World Wide Web (WWW),* 2006.

[16] S. Prasad Kasiviswanathan, H.K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What Can We Learn Privately?" *Proc. 49th Ann. IEEE Symp. Foundation of Computer Science (FOCS),* pp. 531-540, 2008.

[17] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, "Releasing Search Queries and Clicks Privately," *Proc. 18th Int'l Conf. World Wide Web (WWW),* 2009.

[18] R. Kumar, J. Novak, B. Pang, and A. Tomkins, "On Anonymizing Query Logs via Token-Based Hashing," *Proc. Int'l Conf. World Wide Web (WWW),* 2007.

[19] Y. Luo, Y. Zhao, and J. Le, "A Survey on the Privacy Preserving Algorithm of Association Rule Mining," *Proc. Int'l Symp. Electronic Commerce and Security,* vol. 1, pp. 241-245, 2009.

[20] A. Machanavajjhala, D. Kifer, J.M. Abowd, J. Gehrke, and L. Vilhuber, "Privacy: Theory Meets Practice on the Map," *Proc. Int'l Conf. Data Eng. (ICDE),* 2008.

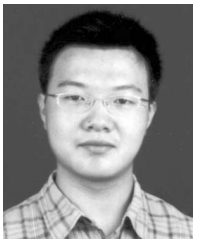
[21] R. Motwani and S. Nabar, "Anonymizing Unstructured Data," Corr, abs/0810.5582, 2008.

[22] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth Sensitivity and Sampling in Private Data Analysis," *Proc. Ann. ACM Symp. Theory of Computing (STOC),* 2007.

[23] P. Samarati, "Protecting Respondents' Identities in Microdata Release," *IEEE Trans. Knowledge and Data Eng.,* vol. 13, no. 6, pp. 1010-1027, Nov./Dec. 2001.

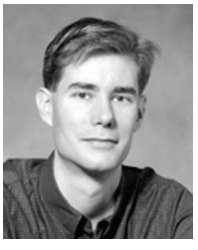
Michaela Götz received the bachelor's degree in computer science at Saarland University in 2007. She is currently working toward the PhD degree in the Computer Science Department at Cornell University in Ithaca, New York.

Ashwin Machanavajjhala received the PhD degree in computer science from Cornell University in 2008 under the supervision of Johannes Gehrke. He is a research scientist at Yahoo! Research in Silicon Valley.



Guozhang Wang received the bachelor's degree from Fudan University. He is currently working toward the PhD degree in the Computer Science Department at Cornell University in Ithaca, New York.

Xiaokui Xiao received the PhD degree in computer science from the Chinese University of Hong Kong in 2008 under the supervision of Yufei Tao. He is an assistant professor in the School of Computer Engineering at Nanyang Technological University (NTU) in Singapore. He spent a year at Cornell University as a postdoctoral associate.



Johannes Gehrke is a professor in the Department of Computer Science at Cornell University in Ithaca, New York. He has received the US National Science Foundation (NSF) Career Award, a Sloan Fellowship, and an IBM Faculty Award. He is the author of numerous publications in data mining and database system. He coauthored the undergrad textbook *Database Management Systems*.