ON USER PRIVACY IN PERSONALIZED MOBILE SERVICES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Michaela Götz

May 2012

ON USER PRIVACY IN PERSONALIZED MOBILE SERVICES

Michaela Götz, Ph.D.

Cornell University 2012

In this thesis, we address privacy concerns in personalized mobile services. While there has been much progress in privacy-preserving data publishing, existing techniques are ill-fitted as they assume static user data. In contrast, in the mobile setting both user contexts and the user population change over time.

- We present *online* frameworks for context-based personalization. Privacy is defined with respect to a set of sensitive contexts specified by the user. Guaranteeing privacy means to limit what an adversary can learn about the user being in a sensitive context. The adversary can have knowledge about the framework and about frequent contexts or patterns of users.
  - For targeting advertisements, our framework generalizes contexts to protect against adversaries knowing the framework. The generalized context is sent to an ad server that selects and returns a set of ads. The most relevant one is displayed to the user. The optimization of selecting the most relevant ad for a user is done *jointly* by the user and the server under constraints on privacy and communication complexity. We show that the optimization problem is NP-hard and present an efficient approximation algorithm.
  - Our second framework creates a filtered stream of contexts made available to services for personalization. We explain which contexts to suppress in order to protect privacy against powerful adversaries knowing the framework and temporal correlations in the stream.

- We also present *offline* algorithms that aggregate user contexts and activities. For example, we can count how many users in a specific context clicked on some ad. This implicit user feedback can be used to update how the personalization is done in the online framework. Our algorithms guarantee probabilistic differential privacy [74] which protects privacy of a user against strong adversaries who know the data of all other users.
  - Our first algorithm is run by a trusted server and – unlike previous work – provides accurate count statistics even for large domains such as Web search queries.
  - Our second algorithm does not require a trusted server. It is the first distributed protocol that efficiently handles dynamic and malicious users.

BIOGRAPHICAL SKETCH

Michaela Götz is a student at Cornell University advised by Johannes Gehrke. Her research interests lie in the area of privacy and data management. She is a recipient of the Microsoft Research Women Scholarship and the Google Engineering Intern Scholarship.

Dedicated to the memory of my grandmother

ACKNOWLEDGEMENTS

I am deeply grateful to the students and researchers I met during grad school for the enlightening discussions we had.

First, I would like to thank my advisor, Johannes Gehrke, for his active support and hands-on guidance. He has continuously encouraged me to think in big pictures and attempt to solve big problems. Johannes had the patience to go with me over the structure of papers, the statement of theorems, the slides of talks and taught me how to present my work.

In addition to Johannes, I am thankful to the other members of my committee, Christoph Koch, Bobby Kleinberg, and Wesley Sine for the input and advice they provided.

Moreover, I would like to thank my mentors during my internships at Microsoft Research, Arvind Arasu and Suman Nath for treating me like a peer and introducing me to important real-world problems. I learned so much from them about what good research is and how to do it. The collaboration was a lot of fun driven by the genuiune interest in the problems and regular foosball matches.

I was fortunate to collaborate with many more people. I would like to thank Truls A. Bjørklund for the fruitful discussions, Raghav Kaushik for invaluable comments, Ashwin Machanavajjhala for mentorship during my first year of grad school, Guozhang Wang for his help and thoughtful suggestions, and Xiaokui Xiao for his sharing insights.

The database group at Cornell provided a great learning environment. In particular, I would like to thank Tuan Chao for the valuable friendship and Gabriel Bender for his insightful feedback. Other people in the computer science department at Cornell provided support on my journey as a PhD student. I would like to thank Steve An, Anna Blasiak, Jeff Chadwick and Vasu Raman for their friendship and Ýmir Vigfússon for his friendship and his travel guidance in Iceland.

Lastly, I would to thank my family for their support and frequent visits. I thank my fiancé, Moritz Hardt, for his love and continuous encouragement. I am grateful to my mother who patiently listened to my attempts to explain my research to non-computer scientists and actually provided interesting comments. I thank my sisters, Sonja, Stefanie and Annika, for keeping me grounded.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Mobile devices today are increasingly equipped with a range of sensors such as GPS, microphone, accelerometer, light, and proximity sensors. These sensors can be effectively used to infer a user's context including her location (at home or in the office? from the GPS), transportation mode (walking or driving? from the accelerometer), social state (alone or in a group? from the microphone), and other activities (in a meeting? from the microphone and the GPS). Consequently, a large and increasing number of services in popular smart phone platforms such as the iPhone, the Android, and the Windows Phone utilize user contexts in order to offer personalized services. Examples of such services include *GeoReminder* that notifies the user when she is at a particular location, *JogBuddy* that monitors how much she jogs in a day, *PhoneWise* that automatically silents the phone during meetings, *SocialGroupon* that delivers coupons or recommendations when she is in a group of friends, etc.

However, these context-aware mobile services raise serious privacy concerns. Today, people already believe that risks of sharing location information outweigh the benefits in many location-based services [96]. One reason why risks are high is that many mobile services aggressively collect much more personal context information than what is needed to provide their functionalities [27] (for example, a calculator service might send user locations to an advertisement server). Moreover, services rarely provide privacy policies that clearly state how users' sensitive information will be used, and with what third-parties it will be shared. Finally, even if the context information is actually used for the functionality and not shared with third-parties, users might simply not trust service providers with some of their context information.

This thesis enhances personalized services with privacy guarantees. This requires (1) to limit the context information that a user shares with the service. In order to personalize based on this limited information only, (2) services need to adapt.

For (1) we explore attractive tradeoffs between privacy and utility. One extreme is to offer no privacy and benefit from the full functionality. Another extreme is to avoid all privacy risks by not to installing these service or not releasing any context information to the services (by explicitly turning off sensors) which destroys utility. Following [58, 96] we can let the user control at a fine granularity when and what personal context information is shared with which service. For example, a user might be okay to release when she is at lunch but she might be hesitant to release when she is at a hospital. With such fine-grained decisions, a user can choose a point in the privacy-utility tradeoff for a service and can still enjoy its full functionality when she chooses to release her context information or when her context information is not actually needed. But how can users assess the risks and decide which point in the privacy-utility tradeoff to choose? With formal privacy guarantees this thesis helps users to understand the consequences of their choices in terms of what an adversary can learn about them.

In order to sanitize services so that user privacy is preserved (2), we need to understand how services utilize user context information for personalization. The basic idea of personalization is to adapt the service based on the users current context. We call this the online personalization framework (2a). Moreover, some services have a feedback loop (2b) and collect historic context information together with information on user activity in order to learn how to personalize. The collection can be run as an offline algorithm periodically. As an example, consider a personalized ad delivery system. This system gathers users' contexts and clicks and interprets them as implicit relevance feedback. This feedback is utilized in the delivery of targeted ads based on the user's current context. Both components utilize context information and need to be adapted to preserve user privacy.

To address the privacy concerns we present offline algorithms for *privacy-preserving statistics gathering* that allow to learn personalization rules and online *personalization frameworks*. Since these two components are fundamentally different, we use different, but state of the art, privacy guarantees for them. For the gathering of statistics, we guarantee differential privacy [24], i.e., that a single user's data will hardly affect the statistics. Such a strong privacy guarantee is needed in practice since statistics are often shared and used extensively. It has been widely used (see for instance [44, 45, 74, 79, 89, 93, 103]). For the online framework, such a strong guarantee is not possible as it conflicts with the idea of personalization. Instead, we follow [5, 17, 106] and define privacy with respect to a set of sensitive contexts and guarantee that an adversary does not learn that a user is in a sensitive context. The two components differ in other key parameters explained below.

1.1 Personalization Frameworks

Frameworks determine how much information about the user's context to share in requests for personalized services and how to honor these requests. The shared context information is limited in order to preserve privacy. We evaluate a framework based on the following parameters:

- **Privacy**: We follow [5, 17, 106] and define privacy with respect to a set of sensitive contexts. Users can decide on the sensitivity of contexts (for example, including "at hospital" but not "walking the dog") with the help of special tools (see for example [94]). A naïve approach to protect sensitive contexts is to simply suppress them. This, however, does not necessarily prevent an adversary from *inferring* sensitive contexts.

  Guaranteeing privacy means to limit what an adversary can learn about a user being in a sensitive context at some point in time from the released contexts. By looking at the released contexts and combining them with her background knowledge, an adversary should not be able to learn that a user was/is/will be in a sensitive context.

- **Adversary model**: In this thesis, we consider Bayesian adversaries with varying degrees of background knowledge. One type of background knowledge we consider is *knowledge about the sanitized personalization framework*. With this knowledge an adversary can infer suppressed sensitive contexts since the suppression itself leaks information: Consider a user who suppresses her context if and only if she is playing a video game at work. An adversary knowing the suppression rule can infer exactly when she is playing a game at work. In general, we want to guard against leakage attacks from adversaries knowing the sanitized personalization framework. Such adversaries are powerful and can reverse-engineer the framework in order to infer information about sensitive contexts. Protecting against them follows Shannon's maxim "The enemy knows the system." and does not rely on privacy through obscurity.

Another type of background we consider is *knowledge about temporal correlations of contexts*. With this knowledge an adversary can infer a sensitive context: Consider a user who suppresses her location when she is at a remote hospital. This, however, might not be sufficient: when she releases her non-sensitive context while she is driving to the remote hospital, the adversary can infer where she is heading. In this case the sensitive context can be inferred from its dependence on non-sensitive contexts. In general, we want to guard against inference attacks from adversaries knowing temporal correlations.

- **Utility**: We measure the utility as information loss of the contexts and as the effect this has on the quality of the personalized service.
- **Efficiency**: We measure the efficiency both on the user-side (deciding what information to share) as well as on the service-side (deciding how to personalize). Moreover, we consider the communication cost between the two.

1.1.1 Targeted Advertising

(1) Our first framework considers the service of delivering targeted advertisements; it can also be used for personalizing local searches of close-by businesses. In order to guarantee privacy against adversaries knowing the framework, we generalize contexts to contain sensitive and non-sensitive contexts.

(2a) We formalize the task of delivering personalized ads from a server to a client as an optimization problem with three important variables: privacy, communication cost, i.e., how few ads are sent to the client, and utility, i.e., how useful the displayed ads are to the user in terms of revenue and relevance.

Several previous works on privacy-preserving search personalization present extreme points in the trade-off space: they propose to personalize search results based on a user's private information either at the server only [66, 105] or at the client only [92]. A server-only solution achieves optimal efficiency at the cost of privacy or utility, while a client-only solution ensures optimal privacy but sacrifices efficiency or utility. While all these prior solutions represent discrete points in a vast trade-off space of privacy, efficiency, and utility, a natural question arises: *can we formalize a common framework for personalization that can be instantiated to any desired trade-off point?* An answer to this question would also provide a better understanding of the trade-offs provided by existing solutions.

We answer the above question with a hybrid framework where the personalization is done jointly by the server and the client. Such a hybrid approach has been previously explored in privacy-preserving location-based services [21, 59, 108]. However, existing techniques focus on answering nearest neighbor queries and cannot be applied to target advertisements.

In our framework users can decide which contexts are sensitive. To protect these sensitive contexts, the current user context is generalized before sending it to the server. Based on the generalized context, the server selects a set of ads or search results, with bounded communication overhead, and sends them to the client. The client then picks and displays the most relevant ad based on the complete context information.

The framework solves the following optimization problem: The ads sent by the server and the ad displayed at the client ought to be chosen in a way that maximizes revenue given constraints on maximum communication cost and privacy. Note that the aforementioned existing privacy-preserving personalization solutions are special cases of our framework; and our framework can be configured to explore other attractive points in the trade-off space of privacy, communication cost, and utility.

The optimization problem is easy when there are no sensitive contexts that need to be protected. However, we show that it becomes NP-hard when the contexts are generalized. For that case, we present an efficient greedy algorithm for hybrid personalization with tight approximation guarantees.

1.1.2 User Context Streams

We propose a framework to sanitize user contexts before releasing them to context-aware services. The goals of the framework are to sanitize user contexts to guarantee privacy against powerful adversaries while making it easy for services to adapt to the sanitized context streams. For the latter goal, we sanitize user contexts by either releasing them or suppressing them. This reflects standard access control mechanisms in existing phones and is the modus operandi of many personalized mobile services [96]. All that is necessary is the ability to deal with suppressed contexts in the stream. Existing services already can tolerate such $\perp$ symbol in a stream of contexts, since sometimes the software fails to extract a high-level context due to the imprecision of sensor readings (e.g., GPS inside a building), the unavailability of sensor data when the sensors are turned off or software bugs. We do not specify how services can use these streams (2) but we expect that it will be easy to adapt to the "suppress or release" paradigm.

(1) We consider adversaries knowing the system and temporal correlations in the context streams. In particular, we use a Markov chain to model user behavior—her frequent contexts and patterns. The central question is: *When and what context should be suppressed in order to protect privacy against adversaries knowing temporal correlations and also the system making this decision?* We present MASKIT, a framework that addresses the above question with two novel privacy checks deciding in an online fashion whether to release or suppress the current context of the user. The *probabilistic check* flips for each context a coin to decide whether to release or suppress it. The bias of the coin is chosen suitable to guarantee privacy. The *simulatable check* makes the decision only based on the released contexts so far and completely ignores the current context. That way, the decision does not leak additional information to the adversary. Both checks provably provide privacy, but interestingly their relative benefit varies across users—there are situations where the probabilistic check provides higher utility than the simulatable check and vice versa. We explain how to select the better one among the two for a given user.

Both checks provide privacy against very strong adversaries who know the system and the Markov chain modeling a user.

To the best of our knowledge, our work is the first to release user context streams while protecting privacy of sensitive contexts against powerful adversaries knowing the system and various temporal correlations including typical user behavior (the user gets up every day at 6am) and correlations (after going to the doctor the user is likely to go to the pharmacy).

We have evaluated MASKIT on a PC as well as on a smart phone, with real public traces from 91 human subjects over the course of nine months, representing user contexts over 266,000 hours. We compare the performance of MaskIt with that of a naïve approach that only suppresses the sensitive contexts. The naïve approach does not guarantee privacy. Our evaluation shows that we do not have to pay a high price in terms of utility and efficiency for the privacy guarantee: MaskIt releases only $< 8\%$ fewer contexts than the naïve approach. Moreover, the suppression decision incurs negligible overhead (≤ 128 ms on a smart phone on average) compared to the context extraction time of typically a few to tens of seconds.

1.2 Privacy-preserving Statistics Gathering

We present privacy-preserving algorithms to gather aggregate statistics across users. Aggregate statistics of interest to personalization are, for example, context-dependent click-through rates of advertisements and frequent keywords, queries and clicks in search logs. They represent implicit user feedback that can be used to personalize Web search results and target advertisements. We evaluate an aggregation algorithm based on the following parameters:

- **Privacy**: We consider a strong privacy guarantee, *differential privacy* [25], which ensures that the output of the algorithm is insensitive to changing/omitting the complete data of a single user. This protects the privacy of a user against strong adversaries who know everyone else's data. We work with a probabilistic relaxation [74] that achieves differential privacy with high probability.
- **Utility**: We can measure utility in terms of the accuracy of the aggregate statistics: How much do the privacy-preserving statistics differ from the

exact statistics? We can also measure utility in terms of the quality of the personalized services relying on these statistics.

- **Setup**: We consider two different set-ups. In the first setup a trusted server is available to gather all user data to compute sanitized statistics. In the second setup no trusted server exists; and we need a distributed aggregation protocol that protects user privacy, even under adversarial scenarios such as when a fraction of the users behave maliciously, send bogus messages, or collude with each other.

- **Efficiency**: We measure computational complexity and in case of a distributed protocol also communication complexity.

- **Scalability**: We need to scale to large domains over which we gather aggregate statistics such as queries in search logs. Moreover, without a trusted party we need a protocol that scales to a large number of users.

- **Robustness**: In case of a distributed protocol we face an additional challenge: The protocol runs across a large and transient population of mobile users. A small fraction of users can become unavailable *during* the course of computing statistics. For example, a user may turn off her mobile device any time or may want to answer an aggregation query only at a convenient time when her phone is being charged and connected through a local WiFi network. Another user might decline to participate in the exchange of certain messages in the protocol. Yet another user might leave or join the community of mobile users. Developing a robust protocol that can deal with these dynamics is challenging with regard to efficiency and utility.

1.2.1 Trusted Server

Previous works on publishing privacy-preserving aggregate statistics such as the Laplacian mechanism [25] are unsuitable for finding frequent items in large domains such as frequent queries in a search log. In fact, we can show that for any algorithm guaranteeing differential privacy, it is impossible to achieve good utility for finding frequent items in large domains.

We describe Algorithm ZEALOUS, developed independently by Korolova et al. [65] and us with the goal to achieve relaxations of differential privacy. Korolova et al. showed how to set the parameters of ZEALOUS to guarantee (ϵ, δ)-indistinguishability [23], and we offer a new analysis that shows how to set the parameters of ZEALOUS to guarantee (ϵ, δ)-probabilistic differential privacy [74], a much stronger privacy guarantee as our analytical comparison shows.

We present an extensive experimental evaluation. We compare the utility of ZEALOUS with that of a k-anonymous algorithm. To protect anonymity, this algorithm publishes all queries that were issued by at least k users. We argue that this guarantee is insufficient and show that the algorithm can be attacked by an adversary who actively influences the search log. Nevertheless, we include it in our comparison to measure the price of a formal privacy guarantee relative to an insufficient k-anonymity guarantee. We find that it is acceptable for two applications we tested that use the sanitized search logs to improve search quality and search performance.

1.2.2 Distributed Protocol

Previous distributed privacy-preserving aggregation protocols [23, 89, 93] do not efficiently handle dynamic user populations, rendering them unsuitable for the estimation of statistics across mobile users. Then, *how can we, in an efficient and privacy-preserving way, gather statistics over a dynamic population?*

To answer this with we present a novel aggregation protocol to compute aggregate statistics without a trusted server that can handle dynamics. To the best of our knowledge, our protocol is the first differentially-private protocol that computes accurate aggregations efficiently even when a fraction of participants become unavailable or behave maliciously.

This protocol can be used to gather context-dependent click-through rates that can be used to personalize advertisements and search results.

We evaluated our protocol with a large trace of location-aware searches in Microsoft Bing for mobile. We measure the utility in terms of the relevance of personalized local searches. We also measure the increased robustness in comparison to efficient previous aggregation protocols [89, 93].

1.3 Organization

This thesis is organized as follows: First, we lay out some background on privacy in Chapter 2, Then we present online frameworks for sanitizing context streams in Chapter 3 and for targeting advertisements in Chapter 4. We present offline algorithms to gather statistics for estimating context-dependent click-through rates without a trusted server in Chapter 5 and for finding frequent keywords, queries and clicks in search logs with a trusted server in Chapter 6.

CHAPTER 2

BACKGROUND

There are many privacy and anonymity definitions in data publishing including *k-anonymity* [91], *ℓ-diversity* [75, 78], *t-closeness* [71], *(ρ_1, ρ_2)-privacy* [28], and variants of *differential privacy* [25, 88]. We refer the reader to an excellent survey [16].

These definitions assume that the data of a user represents a tuple in a database. In this thesis we work with a few of them and adapt them to our dynamic setting with mobile users.

2.1 Anonymity

A simple type of disclosure is the identification of a particular user's data in the release. The concept of k-anonymity has been introduced to avoid such identifications.

Definition 1 (k-anonymity [90])**.** A data release is k-anonymous if the data of every individual is indistinguishable from the data of at least $k-1$ other individuals.

2.1.1 Online Personalization Frameworks

There has been a lot of deep work on anonymity in location-based services. Here, the adversary is assumed to know the location of all users. The goal is to hide the identity of a user issuing a query specifying her location. Using spatial cloaking introduced in the seminal work by Gruteser and Grunwald [39], the user's exact location in a query is replaced by a broader region. Following the principles of k-anonymity [91], a region is broad enough if at least k users are in it. The idea of spatial cloaking has been used extensively since then [18, 5, 33, 35, 81, 39, 104, 56]. Typically, the smallest region containing k users is desired to minimize communication and maximize accuracy.

System Architecture and Efficiency. The proposed systems differ in their system architecture; both centralized systems requiring a trusted third-party and peer-to-peer systems are developed. For the latter a user communicates with her peers to find the smallest region containing at least $k - 1$ other users. These systems further differ in the techniques used to generate cloaked regions (e.g. quad-trees [39, 81], finding cliques in graphs [33], Hilbert curves [56]) yielding differences in their efficiency and accuracy.

Guarantees. Some of these systems are subject to attacks exploiting the knowledge of the system. This basic approach is subject to the "center-of-cloak" attack because an attacker knowing the algorithm used to generate the cloaked area can infer that it is very likely that the user's exact location is in the center of the cloaked region [56, 37, 20]. There are systems preventing this attack [19, 56, 37]. Similarly, inference can be made when users request different levels of anonymity and overlapping regions are published [109].

Most of the work considers single shot queries and thus do not offer protection against adversaries knowing temporal correlations. As such they are not appropriate for tracking services. Notable exceptions include [18, 104].

There is an effort to extend the techniques to general contexts beyond location [87].

2.1.2 Offline Algorithms

There are several proposals in the literature to achieve different variants of k-anonymity for 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 [82], 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 [47]. We call the output a *k-keyword anonymous search log*. Efficient ways to anonymize a search log are also discussed by Yuan et al. [50].

2.2 Privacy

A privacy guarantee limits what an adversary can learn about a user.

2.2.1 Offline Algorithms

Differential privacy guarantees that an adversary learns roughly the same information about a user whether or not the data of that user was included in the release [25].

It has been applied in the centralized setting where a trusted server sanitizes the users' data and in the distributed setting where no such server exists.

Centralized Differential Privacy

Differential privacy has previously been applied in the centralized setting to contingency tables [6, 103, 44, 45], learning problems [9, 57], synthetic data generation [74] and more.

Definition 2 (ϵ-differential privacy [25]). An algorithm $\mathcal{A}$ is ϵ-differentially private if for all user activity logs L and L' differing in the data of a single user and for all output logs O:

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

This definition ensures that the output of the algorithm is insensitive to changing/omitting the complete data of a single user. We will refer to logs that only differ in the data of a single user as *neighboring logs*. Therefore, a user, given the choice of whether or not to supply her data has hardly any incentive to withhold it. The parameter ϵ determines how much the outcome may be affected.

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. *[74] An algorithm M satisfies (ϵ, δ)-probabilistic differential privacy if for all user activity logs L and neighboring ad logs L' obtained from L by adding or deleting the contexts and clicks of a single user we can divide the output space Ω into two sets Ω_1, Ω_2 such that*

$$(1)\, \Pr[M(L) \in \Omega_2] \leq \delta, \textit{ and}$$

and for all $O \in \Omega_1$:

$$(2) e^{-\epsilon} \Pr[M(L') = O] \leq \Pr[M(L) = O] \leq e^{\epsilon} \Pr[M(L') = O]$$

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. For $\delta = 0$ this definition is equivalent to ϵ-differential privacy.

Our motivation for resorting to a probabilistic relaxation instead of differential privacy is twofold: We show that sometimes under differential privacy we cannot maintain any utility. This is the case in publishing frequent queries in search logs, see Section 6.3. Moreover, sometimes it is easier to achieve this relaxation and we do not know how to achieve differential privacy. This is the case in our distributed aggregation protocol which uses Gaussian noise. Gaussian noise can be realized in a distributed manner easily but does only achieve the relaxed privacy guarantee.

The following relaxation has been proposed by Dwork et al. [23].

Definition 4 (indistinguishability [23])**.** An algorithm $\mathcal{A}$ is (ϵ, δ)-indistinguishable if for all user activity logs L, L' differing in the data of a single user and for all subsets $\mathcal{O}$ of the output space Ω:

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

We will compare these two definitions in Section 6.5. 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.

To achieve a relaxation of differential privacy, we can sanitize the output of any real-valued function f : ad logs $\rightarrow \mathbb{R}^d$ by adding Gaussian noise to $f(L)$. The standard deviation of the noise depends on the L_2*-sensitivity* of f which describes how much the value of the function can change if a single user's data is deleted from or added to the input. This change is measured by the L_2-norm.

Theorem 1. *For $\epsilon \leq 1$ and $\sigma^2 \geq s^2 2 \ln(4/\delta)/\epsilon^2$, adding Gaussian noise with variance σ^2 to a function f with L_2-sensitivity s gives (ϵ, δ)-probabilistic differential privacy.*

This theorem has been established for δ-approximate ϵ-indistinguishability [23] and extends to our definition, which is stronger, see Section 6.5.

Proof. Consider two logs L, L' differing in the data of one user. We define $\Omega_1 = \{O = o_1, \ldots, o_d | -\epsilon\sigma^2 \leq \sum_i (o_i - f(L)_i)(f(L)_i - f(L')_i) \leq \epsilon\sigma^2 - s^2/2\}$ We analyze the ratios of outputting some $O \in \Omega_1$ given L and L'.

$$\frac{\Pr[M(L)=O]}{\Pr[M(L')=O]} = \frac{\Pi_{i:1\le i\le d} e^{-(o_i - f(L)_i)^2/(2\sigma^2)}}{\Pi_{i:1\le i\le d} e^{-(o_i - f(L')_i)^2/(2\sigma^2)}}$$

$$= \Pi_{i:1\le i\le d} e^{(-(o_i-f(L)_i)^2+(o_i-f(L')_i)^2)/(2\sigma^2)}$$

$$= e^{1/(2\sigma^2)\cdot\sum_{i:1\le i\le d} -(o_i-f(L)_i)^2+(o_i-f(L')_i)^2}$$

$$= e^{1/(2\sigma^2)\cdot\sum_{i:1\le i\le d} -(o_i-f(L)_i)^2+(o_i-f(L)_i+f(L)_i-f(L')_i)^2}$$

$$= e^{1/(2\sigma^2)\cdot\sum_{i:1\le i\le d} -(o_i-f(L)_i)^2+(o_i-f(L)_i)^2+2(o_i-f(L)_i)(f(L)_i-f(L')_i)+(f(L)_i-f(L')_i)^2}$$

$$= e^{1/(2\sigma^2)\cdot\sum_{i:1\le i\le d} 2(o_i-f(L)_i)(f(L)_i-f(L')_i)+(f(L)_i-f(L')_i)^2}$$

$$\le e^{1/(2\sigma^2)\cdot(s^2+\sum_{i:1\le i\le d} 2(o_i-f(L)_i)(f(L)_i-f(L')_i))}$$

$$\le e^{1/(2\sigma^2)\cdot(s^2+2(\epsilon\sigma^2-s^2/2))} \qquad \text{By Def. of } \Omega_1$$

$$= e^{\epsilon}$$

Similarly, for the other ratio we have for $O \in \Omega_1$:

$$\frac{\Pr[M(L')=O]}{\Pr[M(L)=O]} = \frac{\Pi_{i:1\le i\le d} e^{-(o_i - f(L')_i)^2/(2\sigma^2)}}{\Pi_{i:1\le i\le d} e^{-(o_i - f(L)_i)^2/(2\sigma^2)}}$$

$$= \Pi_{i:1\le i\le d} e^{(-(o_i-f(L')_i)^2+(o_i-f(L)_i)^2)/(2\sigma^2)}$$

$$= e^{1/(2\sigma^2)\cdot\sum_{i:1\le i\le d} -(o_i-f(L')_i)^2+(o_i-f(L)_i)^2}$$

$$= e^{1/(2\sigma^2)\cdot\sum_{i:1\le i\le d} -(o_i-f(L)_i+f(L)_i-f(L')_i)^2+(o_i-f(L)_i)^2}$$

$$= e^{1/(2\sigma^2)\cdot\sum_{i:1\le i\le d} -(o_i-f(L)_i)^2-2(o_i-f(L)_i)(f(L)_i-f(L')_i)-(f(L)_i-f(L')_i)^2+(o_i-f(L)_i)^2}$$

$$= e^{1/(2\sigma^2)\cdot\sum_{i:1\le i\le d} -2(o_i-f(L)_i)(f(L)_i-f(L')_i)-(f(L)_i-f(L')_i)^2}$$

$$\le e^{1/(2\sigma^2)\cdot(\sum_{i:1\le i\le d} -2(o_i-f(L)_i)(f(L)_i-f(L')_i))}$$

$$\le e^{1/(2\sigma^2)\cdot 2(\epsilon\sigma^2)} \qquad \text{By Def. of } \Omega_1$$

$$= e^{\epsilon}$$

Together it follows that

$$e^{-\epsilon}\Pr[M(L')=O] \le \Pr[M(L)=O] \le e^{\epsilon}\Pr[M(L')=O].$$

It remains to bound the probability that the output is not in Ω_1. An output is not in Ω_1 if the noise is really low or really high. We denote by $X_1, \ldots, X_d$ the random variables of the noise added. Note that if the i^{th} output is o_i and the true function value is $f(L)_i$ then the amount of noise added $X_i = o_i - f(L)_i$. These random variables are distributed according to $\mathcal{N}(0, \sigma^2)$.

We will use the following facts:

Fact 1. *The family of Gaussian distributions is closed under linear transformation. In particular, for Gaussian variables $X_1, \ldots, X_d$ with $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, for all $\alpha_i \in \mathbb{R}$*

$$\sum_{i=1}^{d} \alpha_i X_i \sim \mathcal{N}\left(\sum_{i=1}^{d} \alpha_i \mu_i, \sum_{i=1}^{d} \alpha_i^2 \sigma_i^2\right)$$

From this fact it immediately follows that

$$\sum_{i=1}^{d} \alpha_i X_i \sim \sqrt{\sum_{i=1}^{d} \alpha_i^2 \sigma_i^2} Z + \sum_{i=1}^{d} \alpha_i \mu_i, \tag{2.1}$$

where Z is a standard Gaussian random variable (i.e., $Z \sim \mathcal{N}(0, 1)$).

The following fact bounds the tail probability of a standard Gaussian random variable.

Fact 2.

$$\Pr[Z \geq z] \leq \frac{e^{-z^2/2}}{z\sqrt{2\pi}}$$

The proof follows from

$$\Pr[Z \geq z] = \int_z^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \leq \int_z^\infty \frac{t}{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = \frac{e^{-z^2/2}}{z\sqrt{2\pi}}$$

Hence, we can rewrite the probability that a privacy breach happens, i.e,

$$\begin{aligned}
&\Pr[M(L) \notin \Omega_1] \\
\leq &\Pr\left[\sum_{i=1}^{d} X_i(f(L)_i - f(L')_i) < -\epsilon\sigma^2\right] \\
&+ \Pr\left[\sum_{i=1}^{d} X_i(f(L)_i - f(L')_i) > \epsilon\sigma^2 - s^2/2\right] \\
= &\Pr\left[\sqrt{\sum_{i=1}^{d}(f(L)_i - f(L')_i)^2\sigma^2} Z < -\epsilon\sigma^2\right] \\
&+ \Pr\left[\sqrt{\sum_{i=1}^{d}(f(L)_i - f(L')_i)^2\sigma^2} Z > \epsilon\sigma^2 - s^2/2\right] && \text{By Eq. (2.1)} \\
\leq &\Pr\left[\sqrt{s^2\sigma^2} Z < -\epsilon\sigma^2 - s^2/2\right] \Pr\left[\sqrt{s^2\sigma^2} Z > \epsilon\sigma^2 - s^2/2\right] && \text{By Def. of } s \\
\leq &2\Pr\left[\sqrt{s^2\sigma^2} Z > \epsilon\sigma^2 - s^2/2\right] && \text{By symmetry of } Z \\
= &2\Pr\left[Z > \epsilon\sigma/s - s/(2\sigma)\right] \\
\leq &2\frac{e^{-(\epsilon\sigma/s - s/(2\sigma))^2/2}}{(\epsilon\sigma/s - s/(2\sigma))\sqrt{2\pi}} && \text{By Fact (2)} \\
= &2\frac{e^{-\epsilon^2\sigma^2/(2s^2) + \epsilon/2 - s^2/(8\sigma^2)}}{(\epsilon\sigma/s - s/(2\sigma))\sqrt{2\pi}}
\end{aligned}$$

This expression decreases with σ. Thus, it suffices to show that for $\sigma =$

$\sqrt{2\ln(2/\delta)}s/\epsilon$ we have that

$$
\begin{aligned}
2\frac{e^{-\epsilon^2\sigma^2/(2s^2)+\epsilon/2-s^2/(8\sigma^2)}}{(\epsilon\sigma/s-s/(2\sigma))\sqrt{2\pi}} &\leq \delta \\
\Leftrightarrow 2\sigma e^{\epsilon/2} &\leq \delta(\epsilon\sigma^2/s-s/2)\sqrt{2\pi}e^{\epsilon^2\sigma^2/(2s^2)+s^2/(8\sigma^2)} \\
\Leftrightarrow 2\sigma e^{\epsilon/2} &\leq \delta(\epsilon\sigma^2/s-s/2)\sqrt{2\pi}e^{\epsilon^2(\sqrt{2\ln(4/\delta)}s/\epsilon)^2/(2s^2)}e^{s^2/(8\sigma^2)} \\
\Leftrightarrow 2\sigma e^{\epsilon/2} &\leq \delta(\epsilon\sigma^2/s-s/2)\sqrt{2\pi}\frac{4}{\delta}e^{s^2/(8\sigma^2)} \\
\Leftrightarrow \sigma e^{\epsilon/2} &\leq (\epsilon\sigma^2/s-s/2)\sqrt{2\pi}2e^{s^2/(8\sigma^2)} \\
\Leftrightarrow \sqrt{2\ln(2/\delta)}s/\epsilon e^{\epsilon/2} &\leq (\epsilon\sigma^2/s-s/2)\sqrt{2\pi}2e^{s^2/(8\sigma^2)} \\
\Leftrightarrow \sqrt{\ln(2/\delta)}e^{\epsilon/2} &\leq (\epsilon^2\sigma^2/s^2-\epsilon/2)\sqrt{\pi}2e^{s^2/(8\sigma^2)} \\
\Leftrightarrow \sqrt{\ln(2/\delta)}e^{\epsilon/2} &\leq (\epsilon^2 2\ln(2/\delta)s^2/(\epsilon^2 s^2)-\epsilon/2)\sqrt{\pi}2e^{s^2\epsilon^2/(16\ln(4/\delta)s^2)} \\
\Leftrightarrow \sqrt{\ln(2/\delta)}e^{\epsilon/2} &\leq (2\ln(2/\delta)-\epsilon/2)\sqrt{\pi}2e^{\epsilon^2/(16\ln(4/\delta)))} \\
\Leftarrow \sqrt{\ln(2/\delta)}e^{1/2} &\leq (4\ln(2/\delta)-1)\sqrt{\pi} && \text{Since } \epsilon\leq 1 \\
\Leftarrow e^{1/2}+\sqrt{\pi}/\sqrt{\ln(2/\delta)} &\leq 4\sqrt{\ln(2/\delta)}\sqrt{\pi} \\
\Leftarrow e^{1/2}+\sqrt{\pi}/\sqrt{\ln(2)} &\leq 4\sqrt{\ln(2)}\sqrt{\pi} && \text{Since } \delta\leq 1 \\
&\Leftrightarrow \text{true}
\end{aligned}
$$

Thus, $\Pr[M(L)\notin\Omega_1]\leq\delta$. The theorem follows. □

Distributed Probabilistic Differential-Privacy

We can adapt the privacy Definition 3 to the case where no trusted server exists to sanitize the data and instead a distributed protocol is used by considering partial views from all (non-) participants and requiring that a change in an user's data hardly changes the distribution over the messages received. Towards that end we introduce the following notation.

Notation. Consider a user activity log L containing the data of a set of users U.

We can restrict the log L to the data of a subset of the users $U' \subset U$, denoted by $L_{U'}$. If U' contains users not in U, we define $L_{U'}$ to be $L_{U \cap U'}$. For a distributed protocol M involving a set of participants P. Note that the set of users and the set of participants can be overlapping. We define the *view* of a subset of participants $P' \subset P$ in the execution of M on input L, denoted by $V_{P'}$, to be a random variable for all messages received and sent by a participant in P'. For a non-participant we define the view to be the output of the distributed protocol. The set of participants can be partitioned into malicious and honest, but possible unavailable participants. We denote by P_m the set of malicious participants and by P_h the set of honest and possibly unavailable users. We have that $P = P_h \cup P_m$ and $P_h \cap P_m = \emptyset$.

Definition 5. *A distributed protocol M with participants P satisfies (ϵ, δ)-distributed probabilistic differential privacy of the users if for all user activity logs L and all (non-) participants p the following holds.*

In case p is malicious let P'_m denote the set of malicious participant colluding with p which are a subset of the malicious participants P_m. Otherwise let P'_m be $\{p\}$.

There exist randomized algorithms M' and R so that:

(a) *$M'(L_{U \setminus (P_m \cup \{p\})})$ preserves (ϵ, δ)-probabilistic differential privacy.*

(b) *R generates the distribution of the view $V_{P'_m}$ given $M'(L_{U \setminus (P_m \cup \{p\})})$ and $L_{P_m \cup \{p\}}$, i.e., $V_{P'_m}$ and $R(M'(L_{U \setminus (P_m \cup \{p\})}), L_{P_m \cup \{p\}})$ are identically distributed.*

The definition considers the view of a (non-)participant p. This view contains messages sent and received by any participant colluding with p, denoted by P'_m. In case p is not malicious P'_m is $\{p\}$. Privacy means that this view can be simulated from an output that preserves probabilistic differential privacy of the users who are not malicious. This output is generated by some algorithm M' from the users' input that are neither malicious nor equal to p ($L_{U\setminus(P_m\cup\{p\})}$). We do not attempt to protect the privacy of malicious participants. This is indeed impossible, since the adversary controlling them could always sent a message containing L_{P_m} which breaches their privacy. Moreover, we do not protect p's privacy against herself. The simulation takes this output as well as the input from all malicious participants to produce the same view. The second input is indeed necessary for a simulation to be possible at all.

2.2.2 Online Personalization Frameworks

Following [5, 17, 106], guaranteeing privacy means to limit what an adversary can learn about a user being in a sensitive context at some point in time from the released contexts. By looking at the released contexts and combining them with her background knowledge, an adversary should not be able to learn that a user was/is/will be in a sensitive context. Privacy of sensitive locations is not implied by anonymity, e.g., k users can be in the same sensitive location.

Definition 6. *We say that a system $\mathcal{A}$ preserves δ-privacy against an adversary if for all possible inputs $\vec{x}$, for all possible outputs $\vec{o}$ ($\Pr[\mathcal{A}(\vec{x}) = \vec{o}] > 0$), for all times t and all sensitive contexts $s \in S$*

$$\Pr[X_t = s|\vec{o}] - \Pr[X_t = s] \leq \delta.$$

Note, that our privacy definition also limits what an adversary can learn about the user being in *some* (as opposed to a specific) sensitive context at a certain time. In general, for any subset $S' \subset S$, any t, any $\vec{o}$ preserving $\delta/|S'|$-privacy according to the above definition we have that $\Pr[X_t \in S'|\vec{o}] - \Pr[X_t \in$

$S'] \leq \delta$. This is because $\Pr[X_t \in S'|\vec{o}]$ is equal to $\sum_{s \in S'} \Pr[X_t = s|\vec{o}]$. By the $\delta/|S'|$-privacy guarantee this is at most $\sum_{s \in S'} \delta/|S'| + \Pr[X_t = s]$ which is equal to $\Pr[X_t \in S'] + \delta$.

Furthermore, our privacy definition limits what an adversary can learn about the user being in a sensitive context in a time window. For a time window of length Δ any $\vec{o}$ preserving δ/Δ-privacy we have that the posterior belief formed after observing $\vec{o}$ of the user being in a sensitive context s at *some* point in the time window is at most δ larger than her prior belief.

Adversary Model. Different classes of adversaries can be considered. For adversaries with knowing only the system used we assume a uniform prior. To protect privacy we generalize contexts according to a given hierarchy. We choose a cut in the context hierarchy so that each leaf node has exactly one ancestor on the cut to which it is generalized. This basic technique introduces uncertainty of an adversary about whether the user is in a sensitive context or not.

Theorem 2. *Consider a cut in the context hierarchy that is high enough so that for each node on the cut that has a sensitive descendant has at least ℓ descendants. Let n denote the number of contexts. Generalizing each context to its unique ancestor on the cut preserves $1/\ell - 1/n$-privacy against adversaries knowing the system.*

Proof. Consider a sensitive context. The adversary's prior belief is $1/n$. Upon observing a generalized context the adversary conditions his prior belief and infers that the context must have been one of its descendants. The posterior probability among these descendants remains uniform. By construction of the cut it is at most $1/\ell$. By Definition 6 this guarantees $\delta = (1/\ell - 1/n)$-privacy. $\square$

Similar guarantees were given for location-based services. Jensen et al. give an excellent overview of techniques with privacy guarantees for LBS [52] focusing on the "single shot" scenario that does not protect privacy against adversaries knowing temporal correlations. Damiani et al. [22] coarsen the map so that each contains sensitive as well as non-sensitive locations. Bamba et al. [5] provide both privacy and anonymity by dynamically enlarging the cloaking area until privacy and anonymity requirements are met. Khoshgozaran and Shahabi use a Hilbert curve encryption to keep queries private and users anonymous [60]. Another approach is to hide the true query by sending many fake queries [61] or to send a fake query. Yiu and Jensen explain how to answer a nearest neighbor query from results of a fake query [108]. The single shot guarantee assumes weak adversaries.

Later, in Chapter 3 much stronger adversaries knowing temporal correlations and patterns in the user contexts. Previous work providing privacy against adversaries aware of some temporal correlations [17, 35, 36, 40, 84] is closer to these goals.

Gruteser and Liu [40] consider an adversary applying linear interpolation to infer suppressed locations. They introduce uncertainty about sensitive locations by creating zones so that each zone has multiple sensitive locations. If a user visits a sensitive location all locations inside that zone are suppressed until a new zone is entered. This approach does not prevent privacy breaches completely but reduces them in comparison the naïve approach. Cheng et al. [17] consider an adversary knowing the maximum velocity of users. Given two consecutive cloaked regions of a user the adversary can exclude points in the second region that are unreachable from any point in the first one. They protect against this attack but not against adversaries also knowing the system. This work is improved by Ghinita et al. [35] using spatial cloaking and introducing delays. However, the delay can leak information about the user's exact location and is thus vulnerable to an attack from an adversary knowing a little bit about the distribution of the time between consecutive queries: If the delay is just long enough to make every point in the second region accessible from every point in the first region then it is likely that the second region has been artificially delayed. Parate and Miklau release not location but communication traces [86]. A trace is transformed so that the number of possible traces consistent with the transformed trace is maximized subject to a constraint on utility. This technique does not give rise to a semantic privacy guarantee. In summary, the work by [17, 35, 40, 86] does not provably protect privacy against adversaries knowing the system and temporal correlations beyond the max velocity.

To provably provide privacy against these adversaries, cryptographic protocols are employed [36, 84]. Using private information retrieval [69], Ghinita et al. answer nearest neighbor queries [36]. Using on tessellation and private equality testing [29], Narayanan et al. [84] find close-by friends. These strong guarantees come at a high cost – answering a single query takes a few seconds. This work cannot be used to release a privacy-preserving stream of contexts for personalization of many applications. Instead each application has to carry out the personalization on the user's mobile device through expensive NN-queries. This is a severe limitation: Applications have no access to *any* user context data.

Another line of work answers aggregate queries across users [89]. For personalization, though, we need context information about a single user and not aggregate information across a group of users.

To the best of our knowledge, our work presents the first system releasing context streams that protects privacy against very strong adversaries knowing the system and temporal correlations in the form of a Markov chain that go far beyond the max velocity. Moreover, our contexts are not limited to location, but can include the social state and other activities. This enables more powerful personalizations.

CHAPTER 3

PRIVATELY RELEASING USER CONTEXT STREAMS TO PERSONALIZED MOBILE APPLICATIONS

3.1 Introduction

We propose a framework to sanitize user contexts streams. Our framework decides for the user's current context whether to release it to context-aware services or to suppress it. We need to answer the question: *when and what context should be suppressed* to preserve privacy? A naïve approach, that we call MaskSensitive, is to let the user specify sensitive contexts and to simply suppress those. This, however, does not necessarily prevent an adversary from *inferring* sensitive contexts. One reason why an adversary can infer suppressed sensitive contexts is that the suppression itself leaks information as illustrated in Chapter 1. Thus, we want to guard against leakage attacks from *adversaries knowing the suppression system*. Another way an adversary can infer a sensitive context is by exploiting temporal correlations between contexts. Consider a user who suppresses her location when she is at a remote hospital. This, however, might not be sufficient: when she releases the use of a hospital finder app, the adversary can infer where she is heading. In general, we want to guard against inference attacks from *adversaries knowing temporal correlations*. Such adversaries are realistic because human activities and contexts exhibit daily and weekly patterns. For example, Jon is at work at 9am during the week and she always picks up her children from daycare after returning from work. Previous work has shown that human behavior and activities can be modeled well with a simple Markov chain [48, 77]. We use the same approach and model the user behavior as a Markov chain over contexts with transition probabilities that generates the stream of contexts. A Markov chain captures frequencies of contexts and temporal correlations. Adversaries can gain knowledge about patterns and frequencies of contexts from observing a person to create a rough daily schedule or by using common sense; for example, knowing that Jon works full time at a bakery, the adversary can guess that she is most likely to be at work at 9am. An adversary can also extract patterns from the sequence of contexts released to applications. We consider powerful adversaries knowing the Markov chain of a user.

In the presence of such adversaries, the aforementioned question needs to be reformulated as *when and what context should be suppressed in order to protect privacy against adversaries knowing temporal correlations and also the system making this decision?* Here, privacy is defined following Definition 6 with respect to a set of sensitive contexts. Users can decide on the sensitivity of contexts (for example, including "at hospital" but not "walking the dog") with the help of special tools (see for example [94]). Guaranteeing privacy means to limit what an adversary can learn about a user being in a sensitive context at some point in time from the released sequence of contexts. By looking at the released contexts and combining them with her background knowledge, an adversary should not be able to learn that a user was/is/will be in a sensitive state. Our experiments show that the MaskSensitive approach does not meet this requirement: more than half of the masked sensitive states constitute privacy breaches, i.e., upon observing the output generated by MaskSensitive, an adversary can use her background knowledge about a user's Markov chain to gain a lot of confidence in the fact that the user is in a sensitive state. Thus, we need a better system.

Contributions.

In this thesis, we propose MaskIt, a system that addresses the above question with two novel privacy checks deciding in an online fashion whether to release or suppress the current state of the user. The relative benefit of the two privacy checks varies across users. We provide a hybrid privacy check for selecting the better one for a given user.

Both checks provide privacy against very strong adversaries who know the system and the Markov chain modeling a user and her frequent patterns. We also consider weaker adversaries with less background knowledge about the user model. Protecting against these adversaries is challenging because they can learn and gain additional knowledge either from other sources or from the released contexts.[1] We explain how to adapt our checks to preserve privacy against weaker adversaries as they learn.

We have evaluated the performance of MaskIt on a PC as well as on a smart phone, with real public traces from 91 human subjects over the course of nine months, representing user contexts over 266,000 hours.

There is large body of prior work on privacy-preserving publishing of location streams. Most work does not consider adversaries knowing temporal correlations. The only system that we are aware of that provably protects privacy in location streams against adversaries knowing the system and temporal correlation [35] is limited to a *single* type of temporal correlation based on the maximum velocity of a user. It is vulnerable to attacks from adversaries knowing frequencies (e.g., the user is never at home at 2pm) or other temporal correlations (e.g., the distribution of time between two consecutive user locations or

[1]This observation lead to attacks in micro-data publishing [62, 100].

the average velocity). Schemes based on cryptographic protocols [36, 84] provide a strong privacy guarantee, but they cannot release streams of user contexts. To the best of our knowledge, our work is the first to release user context streams while protecting privacy of sensitive contexts against powerful adversaries knowing the system and *various* temporal correlations including typical user behavior (the user gets up every day at 6am) and correlations (after going to the doctor the user is likely to go to the pharmacy). Moreover, previous work offering privacy guarantees for streams focuses exclusively on locations. Our work can handle more general contexts including the social state, the transportation mode, and activities as we illustrate in our experiments.

The remainder of the chapter is organized as follows: Next, we lay out the problem of privately releasing user context streams and describe the architecture of MASKIT. Two alternative privacy checks for MASKIT and a hybrid of the two are presented thereafter in Sections 3.4, 3.5, and 3.6. An experimental evaluation on real user context traces can be found in Sec. 3.7. We review related work in Section 3.8 and conclude in Section 3.9.

3.2 Preliminaries

This section forms the background of the adversarial reasoning. We roughly follow the notation of Manning and Schütze [76].

Markov chains. Markov chains constitute the model of the users and the background knowledge of the adversaries. A Markov chain M has states with labels $C = \{c_1, \ldots, c_n\}$. The transition probability $a_{i,j}^t$ denotes the probability of transitioning from c_i at time $t-1$ to c_j at time t. We use the term *state* to denote the label at a given time (e.g., at home at 9pm).

We denote by X_t the random variable generated from M taking on the value of some label c_i at time t. The independence property of Markov chains states that for two random variables X_{t-1}, X_t

$$\Pr[X_t = c_i | X_1, \ldots, X_{t-1}] = \Pr[X_t = c_i | X_{t-1}]. \tag{3.1}$$

A chain is not necessarily time-homogenous, i.e., the transition probability from one context to another may depend on the time. We consider a model over a fixed time period T. It starts at the "start" state in M and ends T steps later in the "end" state. A way to view such a chain is as a DAG with $T + 2$ levels, in which a state at level t has outgoing edges to states in level $t + 1$. At level 0 we have "start" and at level $T + 1$ we have "end". See Fig. 3.4 for an example.[2] Note that states in different levels might carry the same context label. Thus, we can describe the Markovian process with transition matrices $A^{(1)}, \ldots, A^{(T+1)}$:

$$a_{i,j}^{(t)} = \Pr[X_t = c_j | X_{t-1} = c_i]$$

The joint probability of a sequence of states is:

$$\Pr[X_1, \ldots, X_T] = \prod_{t=1}^{T} a_{X_{t-1}, X_t}^{(t)}$$

In general, we can compute the probability of transitioning from state c_i at time t_1 to state c_j at time t_2 efficiently:

$$\Pr[X_{t_2} = c_j | X_{t_1} = c_i] = (e_i^\intercal A^{(t_1+1)} \cdots A^{(t_2)})_j \tag{3.2}$$

where e_i is the unit vector that is 1 at position i and 0 otherwise.

Hidden Markov Models. Hidden Markov models help us understand how adversaries make inference about suppressed states. Each state has a distribution

[2]This is not necessarily the most compact presentation.

over possible outputs from a set $K = \{k_1, \ldots, k_m\}$. The output at time t is a random variable O_t. The random variable O_t is conditionally independent of other random variables given X_t. We define emission matrices $B^{(t)}$ as:

$$b_{i,k}^{(t)} = \Pr[O_t = k | X_t = c_i]$$

For a given output sequence $\vec{o} = o_1, \ldots, o_T$, we compute the conditional probability that at time t the hidden state was c_i:

$$\Pr[X_t = c_i | \vec{o}] = \frac{\Pr[X_t = c_i, o_1, \ldots, o_{t-1}] \Pr[o_t, \ldots, o_T | X_t = c_i]}{\Pr[\vec{o}]}$$

We use the forward procedure α and the backward procedure β to compute this ratio efficiently.

$$\begin{aligned}
\alpha_i(t) &= \Pr[X_t = c_i, o_1, \ldots, o_{t-1}] \\
&= \sum_j \Pr[X_{t-1} = c_j, o_1, \ldots, o_{t-2}] \Pr[X_t = c_i | X_{t-1} = c_j] \Pr[o_{t-1} | X_{t-1} = c_j] \\
&= \sum_j \alpha_j(t-1) \Pr[X_t = c_i | X_{t-1} = c_j] \Pr[o_{t-1} | X_{t-1} = c_j] \\
&= \sum_j \alpha_j(t-1) a_{j,i}^{(t)} b_{j,o_{t-1}}^{(t-1)}
\end{aligned}$$

We initialize $\alpha_j(1) = a_{\text{"start"},j}^{(1)}$ for all j.

$$\begin{aligned}
\beta_i(t) &= \Pr[o_t, \ldots, o_T | X_t = c_i] \\
&= \sum_j \Pr[o_t | X_t = c_i] \Pr[X_{t+1} = c_j | X_t = c_i] \Pr[o_{t+1}, \ldots, o_T | X_{t+1} = c_j] \\
&= \sum_j \Pr[o_t | X_t = c_i] \Pr[X_{t+1} = c_j | X_t = c_i] \beta_j(t+1) \\
&= \sum_j b_{i,o_t}^{(t)} a_{i,j}^{(t+1)} \beta_j(t+1)
\end{aligned}$$

We initialize $\beta_i(T+1) = 1$ for all i. Putting everything together:

$$\Pr[X_t = c_i | \vec{o}] = \frac{\alpha_i(t) \beta_i(t)}{\sum_j \alpha_j(t) \beta_j(t)} \tag{3.3}$$

3.3 Problem Statement

System Model. We assume a system that models today's sensor-equipped smart phones running context-aware applications (e.g., those mentioned in Sec. 3.1). Untrusted applications access user contexts through MaskIt and do not have access to raw sensor data.[3] For energy-efficient support of continuous applications, MaskIt senses user contexts $x_1, x_2, \ldots$ periodically at discrete points in time (like [8, 80]). Upon extracting a context x_t at time t, MaskIt produces a privacy-preserving output o_t. Continuously running applications can subscribe to the full privacy-preserving context stream $o_1, o_2, \ldots$ MaskIt can also serve applications issuing sporadic queries over the stream (e.g. asking for the current context), although these applications are not the main focus of this work.

To compute o_t, MaskIt employs a check deciding whether to release or suppress the current context x_t. The check follows the "release or suppress" paradigm and restricts the output o_t to be either the true state x_t or the suppression symbol $\perp$, i.e., $o_t \in \{x_t, \perp\}$.[4] We make this restriction because it reflects standard access control mechanisms in existing phones and the modus operandi of many location-based mobile applications [96]. This restriction makes it easy to port existing applications to MaskIt—all that is necessary is the ability to deal with suppressed states in the stream.

User Model. We assume that a user's various contexts and transitions between them can be captured by a Markov chain M; i.e., the user behaves like a sample from M. Previous work has shown that human behavior and activ-

[3]Trusted applications, however, can access raw data and contexts directly if needed.
[4]Other output types (e.g. generalizations) are left for future work.

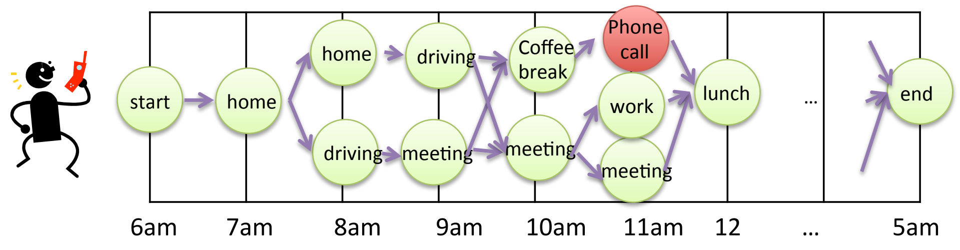


Figure 3.1: Example user model.

ities extracted from smartphone sensors can be modeled well with a simple Markov chain [48, 77]. Markov chains have also been used to model user behavior in other domains including computer-aided manufacturing [67], Web search [11, 51] and search in entity relation graphs [14].

The states in M are labeled with contexts $\{c_1, \ldots, c_n\}$. We consider a model over a day with T time steps: the states in M represent all possible contexts of a user in a day.[5] Figure 3.1 shows an example. The transition probabilities are uniform.

Adversary Model. We consider strong adversaries with full knowledge about the Markov chain M of a user. We further assume that adversaries can access the full output sequence generated by a general suppression system $\mathcal{A}$. They also know $\mathcal{A}$[6]. In the following, we assume that the adversaries apply Bayesian reasoning. They form their prior belief about the user being in context c_i at time t, denoted by $\Pr[X_t = c_i]$, where the randomness comes from the process M

[5]To capture correlations across days, we can consider a larger model capturing a week or a month.

[6]This type of knowledge got overlooked frequently with privacy-preserving micro-data [98, 102].

generating $X_1, \dots, X_T$.

Proposition 1. *The prior belief of an adversary (who knows a user's chain M) about the user being in a sensitive context $c_i \in S$ at time t is equal to*

$$\Pr[X_t = c_i] = (A^{(1)} \cdot A^{(2)} \cdots A^{(t)})_i.$$

Upon observing a released output sequence, they infer as much as possible about contexts and update their belief. The posterior belief, denoted by $\Pr[X_t = c_i | \mathcal{A}(\vec{x}) = \vec{o}]$ is computed by conditioning the prior belief on the observed sequence $\vec{o}$ that was generated from the user's sequence $\vec{x}$ by the system $\mathcal{A}$. The randomness comes from M and $\mathcal{A}$. When clear which system $\mathcal{A}$ we are referencing, we drop it and simply condition the posterior belief on $\vec{o}$. More details about the computation of beliefs can be found the next subsection.

Privacy Goal. Consider a user u with a Markov chain M over contexts $c_1, \dots, c_n$. The user declares a subset of these contexts $S \subset \{c_1, \dots, c_n\}$ as sensitive (e.g., by using special tools [94]). Informally, a released sequence $\vec{o}$ preserves privacy if the adversary cannot not learn much about the user being in a sensitive state from $\vec{o}$. That is for all sensitive contexts and all times we apply Definition 6 and require the posterior belief about the user being in the sensitive context at that time not to be too much larger than the prior belief.[7]

Example 1. *In Figure 3.1 the user declared "Phone call" to be a sensitive context marked in red. Note that in order to preserve privacy, it does not suffice to simply suppress "Phone call". To protect against an adversary knowing the Markov chain as well as the system we can suppress "Coffee break", "Phone call", and "meeting". This*

[7]If the sensitivity of a context depends on the time then we can generalize this definition to sensitive *states*. Extending our system is straight-forward.

gives $\delta = 1/6$-privacy, since the prior belief of $X_1 2 =$ "Phone call" is $1/2$ and the posterior belief is $2/3$. Note, that if we ever released "Coffee break" then the posterior belief of the sensitive state would be 1.

Utility Goal. We want to release as many states as possible, while satisfying the privacy goal. We measure the utility of a system for a user with chain M as the expected number of released states in an output sequence. The randomness comes from M and the system.

The MASKIT **System.** Alg. 1 shows the MASKIT system. It takes as input the user's model M, sensitive contexts S, and the privacy parameter δ. MASKIT learns M from observing the context stream of a user. The transition probability $a^t_{i,j}$ can be estimated as the number of times the user went from c_i at time $t-1$ to c_j at time t divided by the number of times the user went from c_i at time $t-1$ to some other context at time t. The other two parameters (S, δ) can be configured by the user to obtain the desired level of privacy. At its heart, there is a privacy check deciding whether to release or suppress the current state. This privacy check supports two methods initialize and okayToRelease. After the initialization, MASKIT filters a stream of user contexts by checking for each context whether it is okay to be released or needs to be suppressed. MASKIT releases an output sequence "start", $o_1, \ldots, o_T$, "end" for a single day. We can use MASKIT repeatedly to publish longer context streams. It suffices to prove privacy of a single day due to our assumption that there are no correlations across days. In the following sections we present two checks for our framework. They differ based on what information they make the decision whether to release or suppress the current context. The probabilistic check pre-computes each context a probability of suppression during the initialization phase. To determine if a context is

```
procedure MASKIT($\delta$, Markov chain $M$, sensitive contexts $S$)
    initialize($\delta, M, S$)
    $\vec{o}$ =“start”
    for current time $t \in [1, 2, \ldots, T]$ do
        $c_i$ = USERGETCURRENTCONTEXT()
        if okayToRelease($c_i$, $t$, $\vec{o}$) then
            $o_t = c_i$
        else$o_t = \perp$
        end if
        $\vec{o} \leftarrow \vec{o}, o_t$
        Release $o_t$
    end for
    $\vec{o} \leftarrow \vec{o}$, “end”
end procedure
```

Algorithm 1: System to generate δ-private streams.

okayToRelease the check flips a random coin and returns false with the context's suppression probability. Thus, the decision solely depends on the current context. The simulatable check does nothing during the initialization. It determines whether a context is okayToRelease in a only based on the released contexts so far completely ignoring the current context. That way, an adversary can simulate this decision.

3.3.1 Notation

We summarize our notation in Table 3.1.

3.4 Probabilistic Privacy Check

In this section we develop a probabilistic privacy check that specifies for each state c_i at time t' a suppression probability $p_i^{t'}$ with which this state is sup-

$t_1, t, t_2 \in \{1 \ldots, T\}$	times $t_1 \leq t \leq t_2$
$t' \in \{1 \ldots, T\}$	t' current time
$t'' \in \{1 \ldots, T\}$	t'' last time a state has been released
$c_1, \ldots, c_n$	set context
$X_1, \ldots, X_T$	random variables generated from a Markovian process
$a_{i,j}^{(t)}$	transition probability from c_i to c_j at time t
$K = \{k_1, \ldots, k_m\}$	outputs
$O_1, \ldots, O_T$	random output variables generated from HMM
$b_{i,k}^{(t)}$	output probability of $k \in K$ at time t given $X_t = c_i$
$o_t, \ldots, o_T \in K^T$	output sequence
e_i	unit vector that has 1 at position i and 0 in other positions

Table 3.1: Symbols

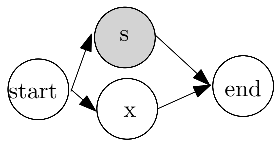


Figure 3.2: Example Markov Chain.

pressed. With probability $1 - p_i^{t'}$, c_i is released at time t'. Among all vectors of suppression probabilities $\vec{p}$ that preserve δ-privacy, we seek one with the maximum utility. We measure utility as the expected number of released contexts:

$$\mathsf{utility}(\vec{p}) = \sum_{\vec{o}=o_1,\ldots,o_T} \Pr[\vec{o}]|\{i|o_i \neq \perp\}| = \sum_{t' \in [T], i \in [n]} \Pr[X_{t'} = c_i](1 - p_i^{t'})$$

Example 2. *Consider the Markov chain in Fig 3.2. Two states s, x are reachable from the "start" state with equal probability of* $1/2$. *Both immediately transition to the "end" state. The sensitive context is s. To achieve* $\delta = 1/4$*-privacy it suffices for the probabilistic check to suppress s with probability 1 and x with probability* $1/3$*: The prior belief of* $X_1 = s$ *is* $1/2$*. The posterior belief upon observing* $\perp$ *is* $\Pr[X_1 = s]p_s^1 / (\Pr[X_1 = s]p_s^1 + \Pr[X_1 = x]p_x^1) = 3/4$. *Suppressing s with probability*

< 1 *breaches privacy. If s was ever released then the posterior belief would be 1 which is more than* δ *larger than the prior belief. Also, if x was suppressed with probability* $< 1/3$ *then the posterior belief of* $X_1 = s$ *upon observing* $\perp$ *would be more than* δ *larger than the prior belief. Thus,* $p_s^1 = 1, p_x^1 = 1/3$ *preserves privacy and maximizes utility.*

Usually, we expect to always suppress a sensitive state s and other states with sufficiently high probability so that upon observing $\perp$, an adversary is uncertain whether the suppressed state is s.

The probabilistic privacy check is outlined in Algorithm 2, where initialize formalizes the optimization problem of finding a suitable suppression probability vector $\vec{p}$. The okayToRelease method simply uses this vector to release or suppress current states. These two methods are used by the MASKIT system described in Section 3.3.

In the remainder of the section, we focus on the optimization problem in the initialize method. It makes use of the ISPRIVATE method that checks if a suppression vector $\vec{p}$ preserves δ-privacy.

3.4.1 Details of the Check – ISPRIVATE

Following Definition 6 we compute whether a vector of suppression probabilities $\vec{p}$ preserves δ-privacy as follows: We enumerate all possible output sequences $\vec{o}$ up to length T and iterate over all times t and all sensitive contexts s to make sure that the posterior belief is at most δ larger than the prior belief. The process is shown in the Procedure ISPRIVATE in Algorithm 2.

Details on computing beliefs. The user's chain M together with the probabilis-

```
procedure initialize($(\delta, M, S)$)
    $\vec{p} \leftarrow \arg\max_{\vec{p}}$ utility($\vec{p}$)
    subject to IsPrivate($\delta, \vec{p}, S, M$) = true
end procedure

procedure okayToRelease($c_i$, $t'$, $\cdot$)
    with probability $p_i^{t'}$ return false
    return true
end procedure

procedure IsPrivate($\delta, \vec{p}, S, M$)
    for each $s \in S$ do
        for $t \in [T]$ do
            Compute prior $\Pr[X_t = s]$.
            for output sequences $\vec{o}$ do
                if $\Pr[\vec{o}] == 0$ then continue
                end if
                Compute posterior $\Pr[X_t = s|\vec{o}]$.
                if posterior $-$ prior $> \delta$ then
                    return false
                end if
            end for
        end for
    end for
    return true
end procedure
```

Algorithm 2: Probabilistic Privacy Check.

tic check using $\vec{p}$ form a hidden Markov model generating $\vec{x}$ as hidden states and $\vec{o}$ as output states. The hidden Markov model extends M with emission matrices. The probability of outputting $\perp$ in state $X_t = c_i$ is p_i^t. The probability of outputting c_i in state $X_t = c_i$ is $1 - p_i^t$. All other outputs in state $X_t = c_i$ have zero probability.

Proposition 2. *An adversary who knows the Markov chain M of a user and the probabilistic check with suppression probabilities $\vec{p}$ computes her posterior belief as*

$\Pr[X_t = s|\vec{o}]$ *in the hidden Markov model defined by* M *and the emission matrices:*

$$b_{i,k}^{(t)} = \Pr[O_t = k|X_t = c_i] = \begin{cases} p_i^t & \text{if } k = \perp \\ 1 - p_i^t & \text{if } k = i \\ 0 & \text{o.w.} \end{cases}$$

Proof. By Proposition 1 the adversary computes her prior belief about the user being in a sensitive context s at time t as $\Pr[X_t = s]$, where the randomness comes from the Markov chain M. As explained in Section 3.3 the adversary computes her posterior belief, denoted by $\Pr[X_t = s|MaskIt(\vec{x}) = \vec{o}]$ by conditioning the prior belief on the observed sequence $\vec{o}$ that was generated from the user's sequence $\vec{x}$ by the MASKIT using the probabilistic privacy check. The randomness comes from M and the probabilistic check.

The hidden Markov model describes the random process of the probabilistic check as it is used by MASKIT: If the user is in context c_i at time t the probability of MASKIT outputting $\perp$ is p_i^t, that is $b_{i,\perp} = p_i^t$. The probability of releasing c_i is $1 - p_i^t$, that is $b_{i,i} = 1 - p_i^t$. The probability of outputting any other context is zero. Thus, conditioning the prior belief on the hidden Markov model as defined above outputting a sequence $\vec{o}$ is equivalent to conditioning the prior belief on $MaskIt(\vec{x}) = \vec{o}$. □

We can efficiently compute this posterior belief using Equation (3.3).

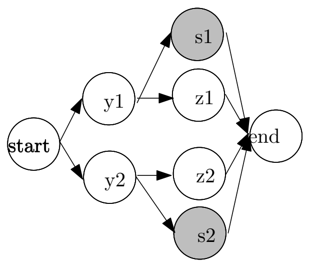


Figure 3.3: δ-privacy is neither convex nor concave.

3.4.2 Details of the Check – Search Algorithm

We can solve the optimization problem of choosing the best suppression probabilities by iterating over all vectors $\vec{p}$ and checking if $\textsc{IsPrivate}(\delta, \vec{p}, S, M)$ returns true. For those passing the check we can compute their $\mathsf{utility}(\vec{p})$ and return the one with the maximum utility. This approach, however, is impractical: There is an infinite number of suppression probabilities and even if we discretized the space $[0, \ldots, 1] \rightarrow \{0, 1/d, 2/d, \ldots, 1\}$ there are still $d^{n \cdot T}$ many vectors to check. One might hope to apply efficient techniques for convex optimization. Convexity and concavity of a function are defined as follows.

Definition 7. *A function f is* convex *if for all $\vec{p}, \vec{q}$ and for all Θ ($0 \leq \Theta \leq 1$)*

$$f(\Theta\vec{p} + (1-\Theta)\vec{q}) \leq \Theta f(\vec{p}) + (1-\Theta) f(\vec{q}) \tag{3.4}$$

A function f is concave *if $-f$ is convex.*

Here, we the function we study is the privacy guarantee. We denote by $\mathsf{privacy}(\vec{p})$ the smallest δ value for which $\vec{p}$ preserves δ-privacy.

However, $\mathsf{privacy}$ is neither a convex nor a concave as the following example illustrates.

Example 3. *We give an example of $\vec{p}, \vec{q}$ and Θ for which Equation 3.4 does not hold. Consider Figure 3.3 with uniform transition probabilities out of each state. The prior belief of a X_2 =s1 and X_2 =s2 is $1/4$. Let $\vec{p}$ be 1 for y1, y2, s1, s2 and 0 otherwise. Let $\vec{q}$ be 1 for z1, z2, s1, s2 and 0 otherwise. It is easy to check that the posterior belief of s1 and s2 is at most $1/2$ and* $\mathsf{privacy}(\vec{p}) = \mathsf{privacy}(\vec{q}) = 1/4$. *Now consider $\Theta = 1/2$. Then $\Theta\vec{p} + (1-\Theta)\vec{q}$ is 1 for s1, s2 and $1/2$ otherwise. To analyze the privacy of this vector, consider the output y1, $\perp$. The posterior belief of X_2 = s1 is $2/3$. Thus,* $\mathsf{privacy}(\Theta\vec{p} + (1-\Theta)\vec{q})$ *is at least $2/3 - 1/4 = 5/12$. However, if* privacy *was convex then it would have to be at most $1/4$. We conclude that* privacy *is not convex.*

We also give an example of $\vec{p}, \vec{q}$ and Θ illustrating that privacy *is not concave. Consider Figure 3.3. Let $\vec{p}$ be 1 or y1, z1, s1, s2 and 0 otherwise. Let $\vec{q}$ be 1 or y2, z2, s1, s2 and 0 otherwise. Given output y2, $\perp$ for $\vec{p}$ (y1, $\perp$ for $\vec{q}$) the posterior belief of X_2 =s2 (X_2 =s1) is 1 and thus* $-\mathsf{privacy}(\vec{p}) = -\mathsf{privacy}(\vec{q}) = -3/4$. *Now consider $\Theta = 1/2$. Then $\Theta\vec{p} + (1-\Theta)\vec{q}$ is 1 for s1, s2 and $1/2$ otherwise. From the calculation above we already now that for this vector the posterior belief of X_2 = s1 given y1, $\perp$ is $2/3$. By symmetry, the posterior belief of X_2 = s2 given y2, $\perp$ is $2/3$ as well. Now consider the output $\perp, \perp$. The posterior belief of X_2 = s1 (or s2) is equal to $1/3$. For other outputs the posterior belief is 0. Thus,* $-\mathsf{privacy}(\Theta\vec{p} + (1-\Theta)\vec{q})$ *is equal to $-5/12$. However, if* privacy *was concave then it would have to be at most $-3/4$. We conclude that* privacy *is not concave.*

Thus, we cannot simply apply techniques for convex optimization. However, we can dramatically reduce the search space by exploiting the *monotonicity* property of privacy. To define monotonicity, we introduce a total ordering of suppression probabilities.

Definition 8. *We say vector $\vec{q}$ that dominates $\vec{p}$, denoted by $\vec{p} \preceq \vec{q}$, if for all i, t :*

$p_i^t \leq q_i^t$.

The monotonicity property says that if we increase the suppression probability we can only improve privacy.

Theorem 3. *Privacy is a monotone property: If $\vec{p}$ preserves δ-privacy then so does any $\vec{q}$ dominating $\vec{p}$.*

Proof. Consider vectors $\vec{p}, \vec{q}$ so that $\vec{q}$ dominates $\vec{p}$ and is larger by ϵ in exactly one dimension: $q_i^{t'} = p_i^{t'} + \epsilon$. Suppose that $\vec{p}$ preserves δ-privacy.

To simplify exposition, we introduce a notation. With two different suppression probabilities $\vec{p}, \vec{q}$, we use a subscript to specify which one is used in the computation of a particular probability. For example, we write $\Pr_{\vec{p}}[X_t = s|\vec{o}]$. We might change one of the values p_j^t to v and write $\vec{p}[p_j^t = v]$ to denote the new suppression probabilities.

In order to prove that also $\vec{q}$ preserves δ-privacy we need to show that the maximum difference (over sensitive states s, time t, outputs $\vec{o}$) between posterior and prior belief does not increase when going from $\vec{p}$ to $\vec{q}$. Fix a sensitive state s and a time t. It suffices to show that the maximum (over outputs $\vec{o}$) of the posterior belief does not increase. In particular, we show that for all $\vec{o}$ either $\Pr_{\vec{p}}[X_t = s|\vec{o}] \geq \Pr_{\vec{q}}[X_t = s|\vec{o}]$ or if that is not the case, then there exists an $\vec{o'}$ such that $\Pr_{\vec{p}}[X_t = s|\vec{o'}] = \Pr_{\vec{q}}[X_t = s|\vec{o'}] \geq \Pr_{\vec{q}}[X_t = s|\vec{o}]$. We consider three cases: Either $o_{t'} = c_i$ or $o_{t'} = c_j$ (for some $j \neq i$) or $o_{t'} = \perp$.

$\underline{o_{t'} = c_j \textbf{ for } j \neq i\textbf{:}}$ Recall that according to Proposition 2 the posterior belief of $X_t = s|\vec{o}$ is computed in the hidden Markov model defined by M and the emission matrices. Changing $p_i^{t'}$ only changes $b_{i,\perp}^{t'}$ and $b_{i,i}^{t'}$. All other emission probabilities remain unchanged. As we can see from Equation (3.3) and the definition of α and β the changed emission probabilities are not part of the computation of $\Pr[X_t = s|\vec{o}]$. Thus, we have that $\Pr_{\vec{p}}[X_t = s|\vec{o}] = \Pr_{\vec{q}}[X_t = s|\vec{o}]$.

$\underline{o_{t'} = c_i}$: In that case increasing $p_i^{t'}$ has no effect on the probability of being in a sensitive state given $\vec{o}$.

$$\Pr_{\vec{p}}[X_t = s|\vec{o}] = \frac{\Pr_{\vec{p}}[X_t = s, \vec{o}]}{\Pr_{\vec{p}}[\vec{o}]} = \frac{p_i^{t'} \Pr_{\vec{p}[p_i^{t'}=1]}[X_t = s, \vec{o}]}{p_i^{t'} \Pr_{\vec{p},[p_i^{t'}=1]=1}[\vec{o}]}$$
$$= \frac{\Pr_{\vec{q}[q_i^t=1]}[X_t = s, \vec{o}]}{\Pr_{\vec{q}[q_i^t=1]}[\vec{o}]} = \frac{q_i^{t'} \Pr_{\vec{q}[q_i^{t'}=1]}[X_t = s, \vec{o}]}{q_i^{t'} \Pr_{\vec{q}[q_i^{t'}=1]}[\vec{o}]} = \Pr_{\vec{q}}[X_t = s|\vec{o}]$$

$\underline{o_{t'} = \perp}$: If $\Pr_{\vec{p}}[X_t = s|\vec{o}] \geq \Pr_{\vec{q}}[X_t = s|\vec{o}]$ we are done. So consider the case where $\kappa \Pr_{\vec{p}}[X_t = s|\vec{o}] = \Pr_{\vec{q}}[X_t = s|\vec{o}]$ for some $\kappa > 1$. To complete the proof, we show that for $\vec{o'} = \vec{o}$ except for $o'_{t'} = i$: $\Pr_{\vec{p}}[X_t = s|\vec{o'}] = \Pr_{\vec{q}}[X_t = s|\vec{o'}] \geq \Pr_{\vec{q}}[X_t = s|\vec{o}]$. The first equality is a result of the calculations above. Thus to suffices to show the following claim:

Claim 1.

$$\Pr_{\vec{p}}[X_t = s|\vec{o'}] \geq \Pr_{\vec{q}}[X_t = s|\vec{o}]. \tag{3.5}$$

We have that

$$\begin{aligned}
&\Pr_{\vec{q}}[X_t = s|\vec{o}] = \kappa \cdot \Pr_{\vec{p}}[X_t = s|\vec{o}] \\
\Leftrightarrow &\frac{\Pr_{\vec{p}}[X_t = s, \vec{o}] + \epsilon \Pr_{\vec{p}[p_i^{t'}=1]}[X_t = s, X_{t'} = i, \vec{o}]}{\Pr_{\vec{p}}[\vec{o}] + \epsilon \Pr_{\vec{p}[p_i^{t'}=1]}[X_{t'} = i, \vec{o}]} = \kappa \cdot \frac{\Pr_{\vec{p}}[X_t = s, \vec{o}]}{\Pr_{\vec{p}}[\vec{o}]} \\
\Leftrightarrow &1 + \frac{\epsilon \Pr_{\vec{p}[p_i^{t'}=1]}[X_t = s, X_{t'} = i, \vec{o}]}{\Pr_{\vec{p}}[X_t = s, \vec{o}]} = \kappa \left(1 + \frac{\epsilon \Pr_{\vec{p}[p_i^{t'}=1]}[X_{t'} = i, \vec{o}]}{\Pr_{\vec{p}}[\vec{o}]}\right) \\
\Leftrightarrow &\frac{\Pr_{\vec{p}[p_i^{t'}=1]}[X_t = s, X_{t'} = i, \vec{o}]}{\Pr_{\vec{p}[p_i^{t'}=1]}[X_{t'} = i, \vec{o}]} = \frac{(\kappa - 1)\Pr_{\vec{p}}[X_t = s, \vec{o}]}{\epsilon \Pr_{\vec{p}[p_i^{t'}=1]}[X_{t'} = i, \vec{o}]} + \kappa \frac{\Pr_{\vec{p}}[X_t = s, \vec{o}]}{\Pr_{\vec{p}}[\vec{o}]}
\end{aligned}$$

We use this to prove the Claim (3.5). We rewrite the left-hand side.

$$\begin{aligned}\frac{\Pr_{\vec{p}}[X_t = s, \vec{o'}]}{\Pr_{\vec{p}}[\vec{o'}]} &= \frac{\Pr_{\vec{p}[p_i^{t'}=1]}[X_t = s, X_{t'} = i, \vec{o}]}{\Pr_{\vec{p}[p_i^{t'}=1]}[X_{t'} = i, \vec{o}]} \\ &= \frac{(\kappa - 1)\Pr_{\vec{p}}[X_t = s, \vec{o}]}{\epsilon \Pr_{\vec{p}[p_i^{t'}=1]}[X_{t'} = i, \vec{o}]} + \kappa \frac{\Pr_{\vec{p}}[X_t = s, \vec{o}]}{\Pr_{\vec{p}}[\vec{o}]} \\ &\geq \kappa \Pr_{\vec{p}}[X_t = s|\vec{o}] = \Pr_{\vec{q}}[X_t = s|\vec{o}]\end{aligned}$$

We arrive at the right-hand side of Claim (3.5) which completes the proof.

□

Furthermore, utility is an anti-monotone property, i.e. we can only decrease utility if we increase suppression probabilities.

Observation 1. *Utility is an anti-monotone property: Any vector dominating $\vec{p}$ cannot have more utility than $\vec{p}$.*

Proof. Consider any vectors $\vec{p}$ and $\vec{q}$ so that $\vec{q}$ dominates $\vec{p}$. Recall that the utility is defined as the expected number of released contexts in the output. We have that

$$\begin{aligned}\mathsf{utility}(\vec{p}) &= \sum_{t' \in [T], i \in [n]} \Pr[X_{t'} = c_i](1 - p_i^{t'}) \\ &\leq \sum_{t' \in [T], i \in [n]} \Pr[X_{t'} = c_i](1 - q_i^{t'}) = \mathsf{utility}(\vec{q})\end{aligned}$$

Here, the inequality follows from the definition of dominance: For all i, t': $q_i^{t'} \geq p_i^{t'}$. Thus, $1 - q_i^{t'} \leq 1 - p_i^{t'}$. □

Our privacy definition has the monotonicity property in common with other definitions such as k-anonymity [91] and ℓ-diversity [75].[8] This monotonicity property allows us to adapt existing efficient search algorithms. We can adapt the greedy approach of MONDRIAN [70] proposed for k-anonymization by starting with the vector $(1, \ldots, 1)$ and gradually reducing the suppression probabilities until reducing any suppression probability further would violate privacy. We end up with a minimal vector. There might be other minimal vectors with more utility, though. To find those we can use the algorithm ALGPR [3] that only relies on the monotonicity of privacy and the anti-monotonicity of utility.

3.4.3 Privacy

It is easy to see that the probabilistic check preserves δ-privacy if ISPRIVATE correctly determines if the suppression probabilities preserve privacy. ISPRIVATE is correct because it follows the definition of privacy considering an adversary knowing the probabilistic check and the Markov chain of the user.

Lemma 1. MASKIT *preserves δ-privacy instantiated with the probabilistic check.*

Proof. According to Definition 6 MASKIT preserves δ-privacy if for all possible outputs $\vec{o}$, all times t and all sensitive contexts $s \in S$ the posterior belief is at most δ larger than the prior belief. This is exactly, what ISPRIVATE determines for a particular vector of suppression probabilities $\vec{p}$. During initialize, the probabilistic check chooses a particular vector of suppression probabilities $\vec{p}$ that has been determined privacy-preserving by ISPRIVATE. This vector is used later to

[8] In their case monotonicity is defined over the lattice of full-domain generalizations of the micro-data.

answer calls of okayToRelease from MASKIT. It follows that MASKIT preserves privacy. □

3.4.4 Utility

The following lemma analyzes the utility of using ALGPR in the search of privacy-preserving suppression probabilities that maximize utility.

Lemma 2. ALGPR *[3] using* ISPRIVATE *solves the optimization problem from* initialize*: It finds suppression probabilities that maximize utility among all suppression probabilities that preserve δ-privacy.*

Proof. ALGPR has been shown to solve the following optimization problem

$$\arg \max_{\vec{p}:p_i^t \in \{0/d,\ldots,d/d\}} \mathsf{P}(\vec{p}) \text{ subject to } \mathsf{Q}(\vec{p}) = \text{ true}$$

where P is an anti-monotone property and Q is a monotone property [3]. In our case, we set P to utility, which is anti-monotone by Observation 1 and we set Q to isPrivate, which is monotone by Theorem 3. □

However, there might be other privacy checks (making the decision not based on the current context) with more utility.

3.4.5 Efficiency

The initialize method of the probabilistic privacy check is expensive. It calls one of the search algorithms, which in turn makes many calls to ISPRIVATE, each of which can take exponential time in the number of states due to the iteration over possible output sequences. In particular, MONDRIAN calls ISPRIVATE $O(Tn\log(d))$ times when using binary search. The number of calls to ISPRIVATE from ALGPR is $O(Tn\log(d))$ times the number of minimally privacy-preserving vectors plus the number of maximally non-privacy-preserving vectors [3]. We now explore optimizations to improve the running time of ISPRIVATE to be polynomial. Across calls to ISPRIVATE, we explain how to re-use partial computations.

Speeding Up ISPRIVATE. To improve the running time of ISPRIVATE we exploit the independence property of Markov chains (3.1). Instead of iterating over all possible output sequences $\vec{o}$ in Line (13) in Algorithm 2 to compute the posterior belief of $X_t = s$ given $\vec{o}$, it suffices to consider output subsequences $o_{t_1}, \ldots, o_{t_2}$ of the form $c_i, \perp, \ldots, \perp, c_j$ with $t_1 \leq t \leq t_2$. We replace Line (13) with

```
1: Let $\mathcal{O} = \{o_{t_1}, \perp, \ldots, \perp | t_1 \leq t, o_{t_1} \in \{c_1, \ldots, c_n, \text{“start”}\}\}$
2: $\mathcal{O} \cup= \{o_{t_1}, \perp, \ldots, \perp, o_{t_2} | t_1 \leq t \leq t_2 \ \wedge \ o_{t_1}, o_{t_2} \in \{c_1, \ldots, c_n, \text{“start”}, \text{“end”}\}\}$
3: for Partial output sequence $\vec{o} \in \mathcal{O}$ do
4:     ...
5: end for
```

The next proposition states that any possible sequence completing a possible partial output sequence $o_{t_1}, \ldots, o_{t_2}$ results in the same conditional probability of $X_t = s$. Thus, the speed-up avoids some redundant checks.

Proposition 3. *Let s be a sensitive context, t_1, t, t_2 be time stamps in $[0, \ldots, T+1]$ such that $t_1 \leq t \leq t_2$. Consider a partial output sequence $o_{t_1}, \perp, \ldots, \perp, o_{t_2}$ with $o_{t_1}, o_{t_2} \in \{c_1, \ldots, c_n, \text{“start”}, \text{“end”}\}$ with non-zero probability. For any possible out-*

put sequence completing this sequence $o_0, \dots, o_{t_1-1}$, $o_{t_2+1}, \dots, o_{T+1}$ we have that

$$\Pr[X_t = s | o_0, \dots, o_{T+1}] = \Pr[X_t = s | o_{t_1}, \dots, o_{t_2}]$$

Proof. It suffices to prove that in our hidden Markov model X_t is conditionally independent of the output variables $O_1, \dots, O_{t_1-1}$, $O_{t_2+1}, \dots, O_T$ given $X_{t_1}, \dots, X_{t_2}$. Two sets of variables $\mathbf{X}, \mathbf{Y}$ are conditionally independent given a third set of variables $\mathbf{Z}$ if $\mathbf{X}$ and $\mathbf{Y}$ are *d-separated* given $\mathbf{Z}$ [34]. This is the case if in any trail (path ignoring the directions of the edges) between a node in $\mathbf{X}$ and a node in $\mathbf{Y}$ there exists a node Z such that

- Z has two incoming arrows on the trail $\dots \rightarrow Z \leftarrow \dots$ and neither Z nor any of its descendants are in $\mathbf{Z}$, or
- Z does not have two incoming arrows on the trail, that is $\dots \rightarrow Z \leftarrow \dots$, and is in $\mathbf{Z}$

This is the case for our HMM. Any node in $O_1, \dots, O_{t_1-1}$ has to go through X_{t_1} to reach X_t. Similarly, any node in $O_{t_2+1}, \dots, O_T$ has to go through X_{t_2} to reach X_t. $\square$

The next proposition states that any possible sequence preceding $o_{t_1}, \perp \dots, \perp$ results in the same conditional probability of $X_t = s$. Thus, the speed-up avoids some redundant checks.

Proposition 4. *Let s be a sensitive context, t_1, t be time stamps in $[0, \dots, T+1]$ such that $t_1 \leq t$. Consider a partial output sequence $\vec{o} = o_{t_1}, \perp, \dots, \perp$ with $o_{t_1} \in \{c_1, \dots, c_n, \text{"start"}\}$ with non-zero probability. For any possible output sequence preceding this sequence $o_0, \dots, o_{t_1-1}$ we have that*

$$\Pr[X_t = s | o_0, \dots, o_{t_1-1}, \vec{o}] = \Pr[X_t = s | \vec{o}]$$

Proof. It suffices to prove that in our hidden Markov model X_t is conditionally independent of the output variables $O_1, \ldots, O_{t_1-1}$, given X_{t_1}. Any node in $O_1, \ldots, O_{t_1-1}$ has to go through X_{t_1} to reach X_t. Thus, X_t and $O_1, \ldots, O_{t_1-1}$ are *d-separated* given X_{t_1} implying independence [34]. $\square$

To compute $\Pr[X_t = s | o_{t_1}, \perp, \ldots, \perp, o_{t_2}]$ with $c_i = o_{t_1}$ and $c_j = o_{t_2}$, we adapt Equation (3.3): We set $\alpha_i(t_1) = 1$, $\alpha_l(t_1) = 0$ (for $l \neq i$) and $\beta_j(t_2) = 1 - p_j$, $\beta_l(t_2) = 0$ (for $l \neq j$). Finally, we can test if $\Pr[o_{t_1}, \ldots, o_{t_2}] > 0$ by testing the following equivalent condition (1) $p_i^{(t_1)}, p_j^{(t_2)} < 1$, (2) $\Pr[X_{t_1} = c_i] > 0$ and (3) c_j is reachable from c_i at time t_1 by a path of length $t_2 - t_1$ through states that have non-zero probability of being suppressed.

Lemma 3. *The running time of the refined* IsPrivate *with Line (13) replaced with Lines (1 to 3) is polynomial in the number of contexts n and the number of time-steps T in the Markov chain M.*

Proof. IsPrivate iterates over all $s \in S$, all times in T, and all output sequences of the form $c_i, \perp, \ldots, \perp, c_j$ and computes prior and posterior beliefs. Thus, at most n^3T^3 times beliefs are computed. Each belief computation takes at most n^2T time. The α, β values can be cached and re-used across belief computations. Overall, the running time is polynomial in the n and T. $\square$

Lemma 4. *Replacing Line (13) replaced with Lines (1 to 3) in* IsPrivate *does not change the result of* IsPrivate.

Proof. Consider a call to the original version of IsPrivate. We distinguish the case where it returns false from the case where it returns true. In both cases, we show that the new version of IsPrivate has the same return value.

- <u>return false:</u> This means that there exists a sensitive state s at time t and an output $\vec{o}$ so that the posterior belief of $X_t = s$ given $\vec{o}$ is more than δ larger than the prior belief of $X_t = s$. Let t_1 denote the latest output in $\vec{o}$ before or at t so that $o_{t_1} \neq \perp$. We distinguish two cases:
  - <u>$o_{t'} = \perp \forall t' \geq t_1$:</u> The new version of IsPrivate computed the posterior belief $\Pr[X_t = s | o_{t_1}, \perp, \ldots, \perp]$, where the number of $\perp$ following o_{t_1} is the same as in $\vec{o}$. By Proposition 4 this is equal to $\Pr[X_t = s|\vec{o}]$. Thus, also the new version of IsPrivate returns false.
  - <u>otherwise:</u> Let t_2 denote the earliest output in $\vec{o}$ after or at t so that $o_{t_2} \neq \perp$. The new version of IsPrivate computed the posterior belief $\Pr[X_t = s|o_{t_1}, \perp, \ldots, \perp, o_{t_2}]$. Proposition 3 states that $\Pr[X_t = s|o_{t_1}, \perp, \ldots, \perp, o_{t_2}]$ is equal to $\Pr[X_t = s|\vec{o'} = o'_0, \ldots, o'_{T+1}]$ for all possible completions $\vec{o'}$ of $\overrightarrow{o_{t_1}, \perp, \ldots, \perp, o_{t_2}}$. Let t' denote the length of $\vec{o}$, i.e., $\vec{o} = o_0, \ldots, o_{t'}$. We consider two cases:
    - <u>$t' = T + 1$:</u> In this case it follows immediately from Proposition 3 that $\Pr[X_t = s|\vec{o}] = \Pr[X_t = s|o_{t_1}, \perp, \ldots, \perp, o_{t_2}]$. Thus, also the new version of IsPrivate returns false.
    - <u>otherwise:</u>

$$\begin{aligned}
&\Pr[X_t = s|\vec{o}] \\
&= \sum_{o_{t'+1},\ldots,o_{T+1}} \Pr[X_t = s|\vec{o}, o_{t'+1}, \ldots, o_{T+1}] \Pr[o_{t'+1}, \ldots, o_{T+1}|\vec{o}] \\
&= \sum_{o_{t'+1},\ldots,o_{T+1}} \Pr[X_t = s|o_{t_1}, \perp, \ldots, \perp, o_{t_2}] \Pr[o_{t'+1}, \ldots, o_{T+1}|\vec{o}] \quad \text{By Prop. 3} \\
&= \Pr[X_t = s|o_{t_1}, \perp, \ldots, \perp, o_{t_2}]
\end{aligned}$$

      Again, it follows that also the new version of IsPrivate returns false.

- return true: Suppose for contradiction that the new version returns false. Then there exists $s, t, t_1, t_2, \vec{o} \in \mathcal{O}$ so that the posterior belief, $\Pr[X_t = s|\vec{o}]$, is more than δ larger than prior belief $\Pr[X_t = s]$.

  - $\vec{o} = o_{t_1}, \perp, \ldots, \perp$: The fact that $\Pr[\vec{o}] > 0$ implies that there exists $o_1, \ldots, o_{t_1-1}$ so that $\Pr[o_0, \ldots, o_{t_1-1}, \vec{o}] > 0$. By Proposition 4, $\Pr[X_t = s|o_0, \ldots, o_{t_1-1}, \vec{o}] = \Pr[X_t = s|\vec{o}]$. This output sequence would have caused the original version of ISPRIVATE to return false, since it iterates over all output sequences and it would have found this particular one to result in a posterior belief more than δ larger than the prior belief. We arrive at a contradiction since the original version of ISPRIVATE did not return false. We conclude that the new version returns true.

  - otherwise: In this case $\vec{o} = o_{t_1}, \perp, \ldots, \perp, o_{t_2}$ with $o_{t_2} \neq \perp$. The fact that $\Pr[o_{t_1}, \perp, \ldots, \perp, o_{t_2}] > 0$ implies that there exists $o_0, \ldots, o_{t_1-1}$ and $o_{t_2+1}, \ldots, o_{T+1}$ so that $\Pr[o_0, \ldots, o_{T+1}] > 0$. By Proposition 3, $\Pr[X_t = s|o_0, \ldots, o_{T+1}] = \Pr[X_t = s|o_{t_1}, \perp, \ldots, \perp, o_{t_2}]$. This output sequence would have caused the original version of ISPRIVATE to return false, since it iterates over all output sequences and it would have found this particular one to result in a posterior belief more than δ larger than the prior belief. We arrive at a contradiction since the original version of ISPRIVATE did not return false. We conclude that the new version returns true.

□

Speeding Up the Search Algorithm. Both search algorithms, MONDRIAN and

ALGPR, start from high suppression probabilities $\vec{p}$ that preserve privacy and use binary search over each probability to see how much it can be decreased without breaching privacy. We can re-use the results from checking privacy of $\vec{p}$ in order to check privacy of $\vec{q}$ dominating $\vec{p}$. If $\vec{p}$ passes the check then so does $\vec{q}$. This fact is already exploited by the two search algorithms. However, we observe that we can get an additional speed-up in these algorithms by caching intermediate results if $\vec{p}$ failed ISPRIVATE.

Lemma 5. *Let $P(\vec{p})$ (for Passed) denote the set of triplets $\langle t, s, \vec{o}\rangle$ that passed the check, i.e., did not result in false in Line (18) in* ISPRIVATE$(\delta, \vec{p}, S, M)$. *For $\vec{q}$ dominating $\vec{p}$, it suffices to check triplets not in $P(\vec{p})$, i.e., the result of* ISPRIVATE$(\delta, \vec{q}, S, M)$ *will not change if the posterior belief of triples in $P(\vec{p})$ is not computed and not verified to be at most δ larger than the prior belief.*

Proof. Consider $\vec{q}$ dominating $\vec{p}$. We distinguish two cases based on the return value of ISPRIVATE$(\delta, \vec{q}, S, M)$. For both cases we show that the output remains the same if triples in $P(\vec{p})$ are not checked in ISPRIVATE$(\delta, \vec{q}, S, M)$.

return true: If ISPRIVATE$(\delta, \vec{q}, S, M)$ returns true then this will also be the case when checking fewer triplets each of which could potentially result in a return value of false.

return false: If ISPRIVATE$(\delta, \vec{q}, S, M)$ returns false then there is a triplet $\langle t, s, \vec{o}\rangle$ which causes it to return false in Line (18). If this triplet is not in $P(\vec{p})$ then this triplet will still be checked (when skipping triplets in $P(\vec{p})$) and cause the return value to be false. Otherwise, we have that $\vec{q}$ dominates $\vec{p}$ and $\Pr_{\vec{q}}[X_t = s|\vec{o}] > \Pr_{\vec{p}}[X_t = s|\vec{o}]$. The proof of Theorem 3 states that then there exists an $\vec{o'}$ so that $\Pr_{\vec{p}}[X_t = s|\vec{o'}] = \Pr_{\vec{q}}[X_t = s|\vec{o'}] \geq \Pr_{\vec{q}}[X_t = s|\vec{o}]$.

By definition of $P(\vec{p})$ the triplet $\langle t, s, \vec{o'} \rangle$ is not in $P(\vec{p})$. Thus, it will still be checked (even when skipping triplets in $P(\vec{p})$). This triplet will fail the check causing a return value of false, since already $\Pr_{\vec{q}}[X_t = s|\vec{o}] - \Pr[X_t = s] > \delta$ and $\Pr_{\vec{q}}[X_t = s|\vec{o'}] \geq \Pr_{\vec{q}}[X_t = s|\vec{o}]$.

$\square$

3.4.6 Main Results

Algorithm 3 shows the improved probabilistic privacy check. The following theorem analyzes privacy, utility and efficiency of this check.

Theorem 4. ***Privacy:*** MASKIT *preserves δ-privacy instantiated with the probabilistic check in Algorithm 3.* ***Utility:*** *The probabilistic check specifies for each state a probability with which the state should be suppressed in* okayToRelease*. It finds suppression probabilities in* initialize *that maximize utility among all suppression probabilities that preserve δ-privacy. This is done by employing* ALGPR *[3] using* ISPRIVATE*.* ***Efficiency:*** *Calling* initialize *with* ALGPR *[3] results in at most $O(Tn\log(d))$ times the number of minimally privacy-preserving vectors plus the number of maximally non-privacy-preserving vectors many calls to* ISPRIVATE*. Each call to* ISPRIVATE *in Algorithm 3 has a running time polynomial in the number of contexts and time-steps. A call to* okayToRelease *takes constant time.*

Proof. We analyze MASKIT using the probabilistic check.

- Privacy: Lemma 1 states that the original privacy check from Algorithm 2 preserves δ-privacy when employed in MASKIT. The two changes made in

```
    procedure initialize($(\delta, M, S)$)
 2:     $P = \emptyset$
        $\vec{p} \leftarrow \arg\max_{\vec{p}}$ utility($\vec{p}$)
 4:     subject to IsPrivate($\delta, \vec{p}, S, M$) = true
    end procedure

 6: procedure okayToRelease($c_i$, $t'$, $\cdot$)
        with probability $p_i^{t'}$ return false
 8:     return true
    end procedure

10: procedure IsPrivate($\delta, \vec{p}, S, M$)
        for each $t \in [T], s \in S$ do
12:         Compute prior $\Pr[X_t = s]$ with Proposition 1.
            Let $\mathcal{O} = \{o_{t_1}, \perp, \ldots, \perp | t_1 \leq t, o_{t_1} \in \{c_1, \ldots, c_n, \text{“start”}\}\}$
14:         $\mathcal{O} \cup= \{o_{t_1}, \perp, \ldots, \perp, o_{t_2} | t_1 \leq t \leq t_2 \wedge o_{t_1}, o_{t_2} \in \{c_1, \ldots, c_n, \text{“start”}, \text{“end”}\}\}$
            for Partial output sequence $\vec{o} \in \mathcal{O}$ do
16:             if $\Pr[\vec{o}] == 0$ then continue
                end if
18:             if $\exists \vec{q}$ such that $\vec{q} \preceq \vec{p}$ and $\langle \vec{p}, t, s, \vec{o} \rangle \in P$ then continue
                end if
20:             Compute posterior $\Pr[X_t = s | \vec{o}]$ with Proposition 2 and Eq. (3.3).
                if posterior $-$ prior $> \delta$ then
22:                 return false
                else
24:                 $P \leftarrow P \cup \{\langle \vec{p}, t, s, \vec{o} \rangle\}$
                end if
26:         end for
        end for
28:     return true
    end procedure
```

Algorithm 3: Improved Probabilistic Privacy Check.

Algorithm 3 to speed-up the check do not change the result of IsPrivate, see Lemmata 4 and 5.

- Utility: Lemma 2 states that the original privacy check from Algorithm 2 employing AlgPR [3] finds suppression probabilities in initialize that maximize utility among all suppression probabilities that preserve δ-privacy.

Again, the two changes made in Algorithm 3 to speed-up the check do not change the result of ISPRIVATE, see Lemmata 4 and 5.

- Efficiency: Lemma 3 shows that using the first speed-up each call to ISPRIVATE has a running time polynomial in the number of contexts and time-steps. ALGPR has been shown to make $O(Tn\log(d))$ times the number of minimally privacy-preserving vectors plus the number of maximally non-privacy-preserving vectors to ISPRIVATE [3]. Finally, okayToRelease flips a coin with a bias specified by the suppression probabilities, which takes constant time.

□

3.5 Simulatable Privacy Check

At current time t' our simulatable check uses Algorithm 4 to decide whether to release or suppress the current state. This decision is made in a *simulatable* way[9], i.e., only based on information available to the adversary at that time, namely, the Markov chain M and the output sequence $o_1, \ldots, o_{t'-1}$. The current state is ignored. The simulatable check decides to release the current state if for any possible state c_j at time t', releasing c_j does not violate privacy.

[9]The notion of simulatability goes back to query auditing [83].

```
    procedure initialize($(\delta, M, S)$) return
2:  end procedure

    procedure okayToRelease($\cdot$, $t'$, $\vec{o}$)
4:      for each possible state $j$ at time $t'$ given $\vec{o}$ do
            for each $s \in S$ do
6:              for $t \in [T]$ do
                    Compute prior $\Pr[X_t = s]$ .
8:                  Compute posterior $\Pr[X_t = s | \langle \vec{o}, c_j \rangle]$.
                    if posterior $-$ prior $> \delta$ then
10:                     return false
                    end if
12:             end for
            end for
14:     end for
        return true
16: end procedure
```

Algorithm 4: Simulatable privacy check.

3.5.1 Details of the Check

Generation of Possible States. To compute all possible states at time t' given $\vec{o}$, let t'' denote the time of the last output $\neq \perp$. State c_j is a possible state if it is reachable from $o_{t''}$ within $t' - t''$ steps: $(e_{o_{t''}} A^{(t''+1)} \cdots A^{(t')})_j > 0$, where $e_{o_{t''}}$ denotes the $o_{t''}$th unit vector that has 1 at position $o_{t''}$ and 0 in other positions.

Details on computing beliefs. The following proposition describes how an adversary computes her posterior belief.

Proposition 5. *Consider an output sequence $\vec{o} = o_1, \ldots, o_{t'}$ computed by the simulatable check. Consider a time t. Let t_1 be the last time before or at t at which a context was released. Let t_2 be the earliest time after t at which a context was released. If no such time exists, set $t_2 = T + 1$ and $o_{T+1} =$ "end". The adversary's posterior belief (knowing*

M and the simulatable check) about a user being in a sensitive context s at time t is

$$\Pr[X_t = s | X_{t_1} = o_{t_1}, X_{t_2} = o_{t_2}].$$

where the randomness comes from M.

The proof follows from the simulatability of the check and the independence property of Markov chains.

Proof. Consider any output $\vec{o} = o_1, \ldots, o_{t'}$ and any output t. Let t_1, t_2 be as specified in the proposition. Further, set $\hat{t}_1, \ldots, \hat{t}_l$ denote the times at which $o_{\hat{t}_i} \neq \perp$. We denote by X the set of sequences $\vec{x}$ that agree with the output on the released states (i.e., $x_{\hat{t}_i} = o_{\hat{t}_i}$ for all $1 \leq i \leq l$).

Claim 2. *Any sequence $\vec{x} \in X$ would have generated the same output $\vec{o}$ if used as input to the simulatable check.*

We prove this claim by induction on the length of the output t'', $1 \leq t'' \leq t'$. Consider any sequence $\vec{x} \in X$.

$\underline{t'' = 1}$**:** For the base case, observe that the suppression decision is made without any information on $\vec{x}$. Therefore, the decision is the same for all inputs. If the decision is to suppress then the output o_1 will be equal to $\perp$ for all inputs. On the other hand, if the decision is to release the state at time $t'' = 1$ then by definition $\hat{t}_1 = 1$ and the output will be $x_1 = o_{\hat{t}_1} = o_1$.

$\underline{t'' \leftarrow t'' + 1}$**:** For the inductive step, suppose the released output for $\vec{x}$ up until $t'' \geq 1$ is equal to $o_1, \ldots, o_{t''}$. The decision of whether to suppress or release the next state at time $t'' + 1$ is made only based on this output. If the decision is to suppress then indeed the output will be $\perp = o_{t''+1}$. If the decision is to release the state at time $t'' + 1$ then there exists some i such that $\hat{t}_i = t'' + 1$ and the output will be $x_{t''+1} = o_{\hat{t}_i} = o_{t''+1}$.

Hence, for any $x \in X$ the simulatable check outputs $\vec{o}$. This is never the case for all other outputs.

Claim 3. *For any $x \notin X$ the check never outputs $\vec{o}$.*

To prove this claim, consider any $x \notin X$. By definition of X, there exists a time t'' so that $x_{t''} \neq o_{t''}$ and $o_{t''} \neq \perp$. Since, the simulatable check either suppresses a context or releases the true context it would never output $o_{t''}$ when the true context is $x_{t''}$.

Putting both claims together we have that

$$\Pr[\vec{o}|\vec{x}] = \begin{cases} 1 & x \in X \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Thus, the posterior belief of a Bayesian adversary who conditions the prior belief on the output $\vec{o}$ is

$$\begin{aligned}
\Pr[X_t = s|\vec{o}] &= \frac{\sum_{\vec{x}:x_t=s} \Pr[\vec{x}|\vec{o}]}{\sum_{\vec{x}} \Pr[\vec{x}|\vec{o}]} \\
&= \frac{\sum_{\vec{x}:x_t=s} \Pr[\vec{o}|\vec{x}] \Pr[\vec{x}]/\Pr[\vec{o}]}{\sum_{\vec{x}} \Pr[\vec{o}|\vec{x}] \Pr[\vec{x}]/\Pr[\vec{o}]} && \text{Due to Bayes Theorem} \\
&= \frac{\sum_{\vec{x}:x_t=s} \Pr[\vec{o}|\vec{x}] \Pr[\vec{x}]}{\sum_{\vec{x}} \Pr[\vec{o}|\vec{x}] \Pr[\vec{x}]} \\
&= \frac{\sum_{\vec{x}\in X:x_t=s} 1 \cdot \Pr[\vec{x}]}{\sum_{\vec{x}\in X} 1 \cdot \Pr[\vec{x}]} && \text{By Equation (3.6)} \\
&= \Pr[X_t = s|X_{\hat{t}_1}, \ldots, X_{\hat{t}_l}]
\end{aligned}$$

Among the states $X_{\hat{t}_1}, \ldots, X_{\hat{t}_l}$ the two states X_{t_1} and X_{t_2} are the closest before and after X_t. Due to the independence property (3.1) given X_{t_1} and X_{t_2} the random variable $X_t = s$ is independent of the other random variables $X_{\hat{t}_i}$. In our Markov chain any trail from a node in $X_{\hat{t}_1}, \ldots, X_{\hat{t}_l}$ has to go through either X_{t_1} or X_{t_2} to reach X_t by definition of X_{t_1} and X_{t_2}. Thus, $X_{\hat{t}_1}, \ldots, X_{\hat{t}_l}$ and X_t are d-separated given X_{t_1}, X_{t_2} which implies conditional independence. Hence, the posterior belief is equal to $\Pr[X_t = s | X_{t_1} = o_{t_1}, X_{t_2} = o_{t_2}]$. $\square$

Corollary 1. *We can compute the posterior belief of $X_t = s$ given $\vec{o}$ as:*

$$\Pr[X_t = s|\vec{o}] = \frac{\Pr[X_t = s|X_{t_1} = o_{t_1}] \Pr[X_{t_2} = o_{t_2}|X_t = s]}{\Pr[X_{t_2} = o_{t_2}|X_{t_1} = o_{t_1}]}$$

We use Equation (3.2) to efficiently compute the transition probability between two states.

Proof. We rewrite the posterior belief using the independence property of Markov chains and Proposition 5.

$$\begin{aligned}
\Pr[X_t = s|\vec{o}] &= \Pr[X_t = s|X_{t_1} = o_{t_1}, X_{t_2} = o_{t_2}] && \text{By Prop. 5} \\
&= \frac{\Pr[X_t = s, X_{t_1} = o_{t_1}, X_{t_2} = o_{t_2}]}{\Pr[X_{t_1} = o_{t_1}, X_{t_2} = o_{t_2}]} \\
&= \frac{\Pr[X_{t_2} = o_{t_2}|X_{t_1} = o_{t_1}, X_t = s] \Pr[X_t = s, X_{t_1} = o_{t_1}]}{\Pr[X_{t_2} = o_{t_2}|X_{t_1} = o_{t_1}] \Pr[X_{t_1} = o_{t_1}]} \\
&= \frac{\Pr[X_{t_2} = o_{t_2}|X_t = s] \Pr[X_t = s|X_{t_1} = o_{t_1}] \Pr[X_{t_1} = o_{t_1}]}{\Pr[X_{t_2} = o_{t_2}|X_{t_1} = o_{t_1}] \Pr[X_{t_1} = o_{t_1}]} && \text{By Eq. 3.1} \\
&= \frac{\Pr[X_t = s|X_{t_1} = o_{t_1}] \Pr[X_{t_2} = o_{t_2}|X_t = s]}{\Pr[X_{t_2} = o_{t_2}|X_{t_1} = o_{t_1}]}
\end{aligned}$$

This completes the proof. $\square$

3.5.2 Privacy

Our check preserves privacy.

Lemma 6. MASKIT *preserves δ-privacy instantiated with the simulatable check in Algorithm 4.*

Informally, this is the case because we only publish a state if it does not increase the probability of any sensitive states too much.

Proof. Consider a sensitive context s and a time t and an output $\vec{o}$ produced by the simulatable check. We argue that the posterior belief of $X_t = s$ given $\vec{o}$ is at most δ larger than the prior belief. Theorem 5 states that the posterior belief of $X_t = s$ given $\vec{o}$ is equal to $\Pr[X_t = s | X_{t_1} = o_{t_1}, X_{t_2} = o_{t_2}]$, where t_1, t_2 denote the time of the two released states closest to t. (If no such time t_2 exists, we set $t_2 = T + 1$ and o_{t_2} ="end".) We distinguish two cases based on whether $t_2 = T + 1$.

- $\underline{t_2 < T + 1}$: In this case, the check verified that this posterior belief is not too large before releasing o_{t_2} at time t_2.
- $\underline{t_2 = T + 1}$: In this case, consider the decision to release t_1: If $t_1 = 0$ then the prior belief is equal to the posterior belief. Otherwise, when o_{t_1} was released the check verified that the posterior belief of $X_t = s$ given $X_{t_1} = o_{t_1}$ and X_{T+1} ="end" is not too large.

□

3.5.3 Utility

The simulatable check is locally optimal as the following lemma states.

Lemma 7. *If the next state is published despite the indication of the privacy check to suppress it (improving the utility) then there is a chance that future states will inevitably breach privacy.*

Proof. Consider a context c_i at time t' that the simulatable check suppresses after having outputted $o_1, \ldots, o_{t'-1}$. This means there exists a possible state c_j at time t' given $o_1, \ldots, o_{t'-1}$ and there exists $s \in S, t$ so that

$$\Pr[X_t = s | o_1, \ldots, o_{t'-1}, c_j] - \Pr[X_t = s] > \delta.$$

We consider two cases:

- $\underline{t \leq t'}$: In this case, we have by Proposition 5 that for any output sequence completing $o_1, \ldots, o_{t'-1}, c_j$ the posterior belief of $X_t = s$ is the same and we have a privacy breach.
- $\underline{t > t'}$: In this case, the output $o_1, \ldots, o_{t'-1}, c_j$ already constitutes a privacy breach. We further show that independent of the following release or suppression decisions based on the outputs so far, we can find possible inputs $x_{t'+1}, \ldots, x_T$ so that the output $o_1, \ldots, o_T$ breaches privacy.

  Claim 4. *Let $\vec{o}$ denote the output sequence so far. If the decision is to release the next state at time t_2 we can always find a possible context c_k at time t_2 so that* $\Pr[X_t = s | \vec{o}] \leq \Pr[X_t = s | \vec{o}, c_k]$.

  Let t_1 denote the last released output and $c_j = o_{t_1}$. We consider three cases based on the order of t_1, t, t_2.

  - $\underline{t_1 > t}$: In this case, by Proposition 5 we have that for any possible output sequence completing $\vec{o}$ the posterior belief of $X_t = s$ is the same and we have a privacy breach.

- $\underline{t_1 \leq t_2 \leq t:}$ Suppose for contradiction that for all $1 \leq k \leq n$ we have that

$$\Pr[X_t = s | X_{t_2} = c_k] < \Pr[X_t = s | X_{t_1} = o_{t_1}] \tag{3.7}$$

  This means that,

$$\begin{aligned}
&\Pr[X_t = s | X_{t_1} = o_{t_1}] = \frac{\Pr[X_t = s, X_{t_1} = c_j]}{\Pr[X_{t_1} = c_j]} \\
&= \frac{\sum_k \Pr[X_t = s, X_{t_1} = o_{t_1}, X_{t_2} = c_k]}{\Pr[X_{t_1} = o_{t_1}]} \\
&= \sum_k \Pr[X_t = s | X_{t_2} = c_k] \Pr[X_{t_2} = c_k | X_{t_1} = o_{t_1}] \qquad \text{By Eq. (3.1)} \\
&< \sum_k \Pr[X_t = s | X_{t_1} = o_{t_1}] \Pr[X_{t_2} = c_k | X_{t_1} = o_{t_1}] \qquad \text{By Ass. (3.7)} \\
&= \Pr[X_t = s | X_{t_1} = o_{t_1}]
\end{aligned}$$

  We arrive at a contradiction. We conclude that there exists a k so that

$$\Pr[X_t = s | X_{t_2} = c_k] \geq \Pr[X_t = s | X_{t_1} = o_{t_1}]$$

  By Proposition 5 this means that $\Pr[X_t = s | \vec{o}] \leq \Pr[X_t = s | \vec{o}, c_k]$.

- $\underline{t_1 \leq t \leq t_2:}$ Suppose for contradiction that for all $1 \leq k \leq n$ we have that

$$\Pr[X_t = s | X_{t_1} = o_{t_1}, X_{t_2} = c_k] < \Pr[X_t = s | X_{t_1} = o_{t_1}] \tag{3.8}$$

  We rewrite the left-hand side:

$$\begin{aligned}
&\Pr[X_t = s | X_{t_1} = o_{t_1}, X_{t_2} = c_k] = \frac{\Pr[X_t = s, X_{t_1} = o_{t_1}, X_{t_2} = c_k]}{\Pr[X_{t_1} = o_{t_1}, X_{t_2} = c_k]} \\
&= \frac{\Pr[X_t = s | X_{t_1} = o_{t_1}] \Pr[X_{t_2} = c_k | X_t = s]}{\Pr[X_{t_2} = c_k | X_{t_1} = o_{t_1}]}
\end{aligned}$$

  We re-state our Assumption (3.8):

$$\Pr[X_{t_2} = c_k | X_t = s] < \Pr[X_{t_2} = c_k | X_{t_1} = o_{t_1}] \tag{3.9}$$

We rewrite the right-hand side of our Assumption (3.8):

$$\begin{aligned}
&\Pr[X_t = s|X_{t_1} = o_{t_1}] \\
=&\frac{\Pr[X_t = s, X_{t_1} = c_j]}{\Pr[X_{t_1} = c_j]} \\
=&\frac{\sum_k \Pr[X_t = s, X_{t_1} = o_{t_1}, X_{t_2} = c_k]}{\Pr[X_{t_1} = o_{t_1}]} \\
=&\sum_k \Pr[X_t = s|X_{t_1} = o_{t_1}] \Pr[X_{t_2} = c_k|X_t = s] && \text{By Eq. (3.1)} \\
<&\sum_k \Pr[X_t = s|X_{t_1} = o_{t_1}] \Pr[X_{t_2} = c_k|X_{t_1} = o_{t_1}] && \text{By Ass. (3.9)} \\
=&\Pr[X_t = s|X_{t_1} = o_{t_1}]
\end{aligned}$$

We arrive at a contradiction. We conclude that there exists a k so that

$$\Pr[X_t = s|X_{t_1} = o_{t_1}, X_{t_2} = c_k] \geq \Pr[X_t = s|X_{t_1} = o_{t_1}]$$

By Proposition 5 this means that $\Pr[X_t = s|\vec{o}] \leq \Pr[X_t = s|\vec{o}, c_k]$.

From Claim 4 it follows that if the simulatable check indicated that the current state ought to be suppressed, but instead it was released, then we can construct a sequence of possible inputs so that privacy is breached. Privacy is breached no matter how the following release or suppress decisions are made as long as they are made only by considering the output released so far. For example, even when suppressing all following states privacy is still breached.

□

However, the privacy check might not globally maximize the total number of released states. At times it might be wiser to suppress a state even when the check indicates that publishing it will not breach privacy. Moreover, there might be other privacy checks (making the decision not based on the previously released states) with more utility.

3.5.4 Efficiency

Lemma 8. *The running time of each call to* okayToRelease *of the simulatable check is polynomial in the number of contexts n and time-steps T.*

Proof. okayToRelease iterates over $s \in S, t \in [T]$ and all possible states at time t' and computes prior and posterior beliefs. Thus, at most n^2T beliefs are computed. The computation of a prior belief according to Proposition 1 takes time $O(n^2T)$. The computation of a posterior belief according to Corollary 1 using Equation 3.2 takes time $O(n^2T)$. Overall the running time is $O(n^4T^2)$. Note that the running time can be improved by caching beliefs. $\square$

We can speed up the check by noticing that many checks in Line (9) in Algorithm 4 are carried out over and over again for consecutive calls of okayToRelease from the system. Some of these checks are redundant.

Lemma 9. *Consider* MaskIt *using the simulatable check. At current time t', let t'' denote the time of the last output $\neq \perp$ before t'. In Line (6) of Algorithm 4 only iterating t over $t''+1, \ldots, T$ does not change the output of* okayToRelease.

Proof. Consider t', t'' as in the Lemma. Let $o_1, \ldots, o_{t'-1}$ denote the output sequence so far. We argue that

Claim 5. *For $t \leq t''$ we have that for all s the posterior belief of $X_t = s$ given $o_1, \ldots, o_{t'-1}$ is at most δ larger than the prior belief.*

```
    procedure initialize($(\delta, M, S)$) return
 2: end procedure

    procedure okayToRelease($\cdot$, $t'$, $\vec{o}$)
 4:     Let $t''$ be the last time at which $o_{t''} \neq \perp$.
        for each possible state $j$ at time $t'$ given $\vec{o}$ do
 6:         for each $s \in S$ do
                for $t \in [t''+1, \ldots, T]$ do
 8:                 Compute prior $\Pr[X_t = s]$ .
                    Compute posterior $\Pr[X_t = s | \langle \vec{o}, c_j \rangle]$.
10:                 if posterior $-$ prior $> \delta$ then
                        return false
12:                 end if
                end for
14:         end for
        end for
16:     return true
    end procedure
```

Algorithm 5: Improved simulatable privacy check.

To prove the claim, let t_1 denote the earliest time after or at t at which a context was released. When o_{t_1} was released it was verified that $\Pr[X_t = s | o_1, \ldots, o_{t_1}] - \Pr[X_t = s] \leq \delta$. Otherwise, the context at time t_1 would never have been released. Due to Proposition 5 we have that the output after t_1 does not affect the posterior belief of $X_t = s$ given $\vec{o}$ for all s. In particular, $\Pr[X_t = s | o_1, \ldots, o_{t''}] = \Pr[X_t = s | o_1, \ldots, o_{t_1}]$. Thus, $\Pr[X_t = s | o_1, \ldots, o_{t''}] - \Pr[X_t = s] \leq \delta$ and the claim follows.

From the claim it follows immediately, that skipping over $t \in [1, \ldots, t'']$ does not change the return value of okayToRelease. $\square$

3.5.5 Main Results

Algorithm 5 shows the improved simulatable privacy check. The following theorem analyzes privacy, utility and efficiency of this check.

Theorem 5. *We analyze* MaskIt *using the simulatable check.* ***Privacy:*** MaskIt *preserves δ-privacy instantiated with the simulatable check in Algorithm 5.* ***Utility:*** *The simulatable check greedily releases states.* ***Efficiency:*** *Calling* initialize *takes no time. A call to* okayToRelease *takes polynomial time in the number of contexts and time-steps.*

Proof. We analyze the simulatable check.

- Privacy: Lemma 1 states that the original privacy check from Algorithm 4 preserves δ-privacy when employed in MaskIt. The change made in Algorithm 5 to speed-up the check does not change the result of IsPrivate, see Lemma 9.
- Utility: Lemma 7 states that Algorithm 4 releases a state whenever it is safe to do so. This is not affected by the speed-up which does not change the return value of okayToRelease, see Lemma 9.
- Efficiency: Lemma 8 states that the running time of Algorithm 4 is polynomial in the number of contexts n and time-steps T. The change made in Algorithm 5 further improves the running time.

□

3.6 Comparative Analysis

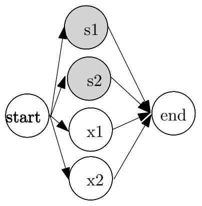
3.6.1 Utility

A natural question is which of the two checks (probabilistic or simulatable) provides more utility. We study this question from an analytical point of view in this section and from an experimental point of view in Sec. 3.7. The relative benefit of the two checks depends on a user's Markov chain and the sensitive contexts. We give an example of a Markov chain where the simulatable check performs better than the probabilistic check and one where the probabilistic check performs better than the simulatable check.

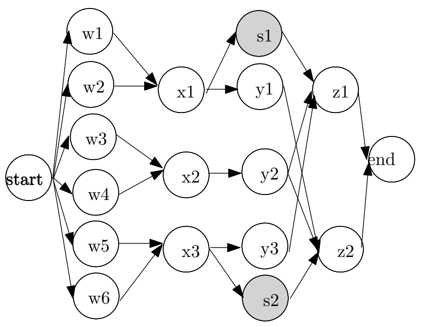
Probabilistic check better.

Example 4. Consider the example Markov chain in Figure 3.4(a). The transition probabilities are uniform across the outgoing edges of a node. States $s1, s2$ are sensitive. Suppose we want $\delta = 1/4$-privacy. The simulatable check suppresses X_1. The probabilistic check, however, only suppresses the sensitive contexts. This suffices to protect privacy. The prior belief of $X_1 = s_i$ is $1/4$ and the posterior belief given $\perp$ is $1/2$.

This example illustrates a weakness of the simulatable check: It makes the suppression decision without looking at the current state. If there is a chance of currently being in a sensitive state that has a prior belief $< 1 - \delta$ then the simulatable check always suppresses the current state. The probabilistic check considers the current state and in such a case does not necessarily have to suppress it.



(a) Probabilistic Better



(b) Simulatable Better

Figure 3.4: Two Markov chains.

Simulatable check better.

Example 5. Consider the example Markov chain in Figure 3.4(b). States $s1, s2$ are sensitive. Suppose we want $\delta = 1/3$-privacy. The simulatable check outputs one of these sequences: ⟨start, w1, x1 $\perp, \perp$⟩, ⟨start, w2, x1 $\perp, \perp$⟩, ⟨start, w3, x2, y2, z1 ⟩, ⟨start, w3, x2, y2, z2 ⟩, ⟨start, w4, x2, y2, z1 ⟩, ⟨start, w4, x2, y2, z2 ⟩, ⟨start, w5, x3 $\perp, \perp$⟩, ⟨start, w6, x3 $\perp, \perp$⟩. The expected number of released states is $8/3$. The utility of the probabilistic check is smaller than that.

Claim 6. *For the probabilistic check the utility of all minimally privacy-preserving suppression probabilities is at most* $7/3$.

To prove this claim, observe that the suppression probabilities of s1, s2 have to be 1 because otherwise the output with $o_3 =$ s1 or s2 increases the posterior belief to 1 which exceeds the prior belief (1/6) by more than δ. We first note that in addition all of w1, w2, x1 have to output $\perp$ with probability 1 or both z1 and y1 have to output $\perp$ with probability 1. Otherwise, a possible output sequence includes one of w1, w2, x1 and z1 which gives away that the user must have been in $s1$ or a possible output sequence includes one of w1, w2, x1 and $\perp$ which increases the probability that the user was in $s1$ by more than δ. Similarly, all of w4, w5, x3 or both z2, y3 have to be always suppressed. Consider the case where z1, y1, z2, y3 are always suppressed. Here, the utility is $7/3$, which is less than that of the simulatable check. The same is true if you consider always suppressing w1, w2, x1 and w4, w5, x3 where again the utility is $7/3$. In order to guarantee privacy the suppression probabilities would have to be increased further. Lastly, consider the case where z1, y1 and w4, w5, x3 (z2, y3 and w1, w2, x1, resp.) are being suppressed. Again, this does not guarantee privacy but any dominating suppression probabilities will have utility no better than $7/3$. In summary, any privacy-preserving suppression probabilities will have utility less than that of the simulatable check.

This example illustrates a weakness of the probabilistic check: Its decision ignores the previously released states. It might have to suppress a state because there exists some $\vec{o}$ in Line (13) for which otherwise the posterior belief of some sensitive state is too high. Now, if this $\vec{o}$ is inconsistent with the outputs released so far it might be okay to release the state. For example, if the output released so far is ⟨start, w3⟩ then everything following can be released. The simulatable algorithm makes decisions based on the released states so far and can thus achieve higher utility.

Hybrid Privacy Check. How can we analytically determine which one of the two checks is more suitable for a particular user? We explain how to compute the utility of both checks. Then it is easy to pick the better one. Recall from Sec. 3.3 that the utility is defined as the expected number of released states in an output.

For the probabilistic check with suppression probabilities $\vec{p}$ we compute the utility as:

$$\mathsf{utility}^{\mathsf{Prob}}(M) = \sum_{i,t} (1 - p_i^t) \Pr[X_t = c_i].$$

For the simulatable check we introduce a short-hand, $\mathsf{supp}_i(t)$, for the number of suppression symbols immediately following the release of c_i at time t.

$$\begin{aligned} \mathsf{supp}(i,t) = &\arg\max_{t_2} t_2 - t \text{ s.t. } \forall t_2' : t < t_2' \leq t_2 : \\ &\mathsf{okayToRelease}(\,\cdot\,, t_2', \langle o_1, \ldots, o_{t-1} c_i, \perp^{t_2'-t-1}\rangle) == \text{false}, \end{aligned}$$

where $o_1, \ldots, o_{t-1}$ is some output sequence that is consistent with $o_t = c_i$. If no such sequence exists, then o_t can never be c_i and we define $\mathsf{supp}(i,t)$ to be 0. Using $\mathsf{supp}_i(t)$ we can compute recursively the expected number of suppressions following the release of $X_t = c_i$:

$$\begin{aligned} &E[|\{t_2 | o_{t_2} = \perp, t < t_2 \leq T\}| | o_t = c_i] = \gamma_i(t) \\ =&\mathsf{supp}_i(t) + \sum_j \Pr[X_{t+\mathsf{supp}_i(t)+1} = c_j | X_t = c_i] \gamma_j(t + \mathsf{supp}_i(t) + 1) \end{aligned}$$

Our base case is $\gamma_j(T+1) = 0$ for all j. Overall, the utility of the simulatable check is

$$\mathsf{utility}^{\mathsf{Simulatable}}(M) = T - \gamma_{\text{"start"}}(0).$$

Our hybrid check computes the utility of both the simulatable and the probabilistic check and chooses the one with the higher utility.

Theorem 6. *The hybrid check correctly computes the utility of the simulatable and the probabilistic check employed by* MASKIT.

Proof. Recall that the utility is defined as the expected number of released states in an output sequence. The randomness comes from M and the check. For the probabilistic check with suppression probabilities $\vec{p}$ the computation of the utility closely follows this definition:

$$\begin{aligned}\mathsf{utility}^{\mathsf{Prob}}(M) &= \sum_{\vec{o}} \Pr[\vec{o}] |\{i | o_i \neq \bot\}| \\ &= \sum_{i,t} \Pr[X_t = c_i] \cdot \Pr[O_t = \bot | X_t = c_i] \\ &= \sum_{i,t} (1 - p_i^t) \Pr[X_t = c_i]\end{aligned}$$

For the simulatable check we first show that the notion of $\mathsf{supp}_i(t)$ is well defined, i.e., that the number of suppressed states following the release of c_i at time t depends only on i, t and not on anything else.

Claim 7. *Consider two output sequences $\vec{o}, \vec{o'}$ both of length $t-1$. If both sequences agree on the most recently released context then for all i, j*

$$\mathsf{okayToRelease}(c_i, t, \vec{o}) == \mathsf{okayToRelease}(c_j, t, \vec{o})$$

To prove the claim first note that okayToRelease is independent of its first input argument. It remains to argue, why its decision only depends on the last released context. Let t'' denote the most recent time most for which $o_{t''} \neq \bot$. We have that $o_{t''}, \ldots, o_{t-1}$ is equal to $o'_{t''}, \ldots, o'_{t-1}$. By Proposition 5 for all contexts c_k, all $s \in S$ and all $t' \geq t''$

$$\begin{aligned}\Pr[X_{t'} = s | \vec{o}, c_k] &= \Pr[X_{t'} = s | o_{t''}, \ldots, o_{t-1}, c_k] \\ &= \Pr[X_{t'} = s | o'_{t''}, \ldots, o'_{t-1}, c_k] = \Pr[X_{t'} = s | \vec{o'}, c_k].\end{aligned}$$

Thus, for all contexts c_k, all $s \in S$ and all $t' \geq t''$ we have that $\mathsf{okayToRelease}(c_i, t, \vec{o})$ returns false in Line (11) in Algorithm 5 if and only $\mathsf{okayToRelease}(c_i, t, \vec{o})$ returns false. By Lemma 9 for all contexts c_k, all $s \in S$

and all $t' < t''$ we have that

$$\Pr[X_{t'} = s|\vec{o}, c_k] - \Pr[X_{t'} = s] \leq \delta \qquad \text{and}$$
$$\Pr[X_{t'} = s|\vec{o'}, c_k] - \Pr[X_{t'} = s] \leq \delta$$

We conclude that for all s, t' $\mathsf{okayToRelease}(c_i, t, \vec{o})$ returns false in Line (11) in Algorithm 5 if and only $\mathsf{okayToRelease}(c_i, t, \vec{o})$ returns false. Thus their return value is equal.

Thus, $\mathsf{supp}_i(t)$ is well defined as the number of suppressed states following the release of c_i at time t. Its computation is correct: It chooses one possible output sequence $o_1, \ldots, o_{t-1}$ consistent with $o_t = c_i$ and counts how often okayToRelease returns $\perp$ given $o_1, \ldots, o_{t-1}, c_i$ and a growing number of $\perp$. Note that by Claim 7 the choice of $o_1, \ldots, o_{t-1}$ is arbitrary. We conclude that $\mathsf{supp}_i(t)$ indeed computes the number of suppression symbols immediately following the release of c_i at time t.

$\gamma_i(t)$ correctly computes the number of suppressions following the release of $X_t = c_i$ by adding to $\mathsf{supp}_i(t)$ the sum over all possible next releases of their probability times the number of suppressions following them.

Finally, $\mathsf{utility}^{\mathsf{Simulatable}}(M)$ correctly computes the utility by subtracting from T the expected number of suppressed states. □

3.6.2 Efficiency

In the MASKIT system, initialize is computed once offline, while okayToRelease is computed online whenever a new context is extracted. Our privacy checks present different tradeoffs between offline and online computation. The simulatable check does not require any initialization; all its computational overhead is incurred during the filtering. If MaskIt has to go live and create a stream immediately then the simulatable check is the only option. The probabilistic and hybrid checks, conversely, perform most of the computation offline during initialize and are suitable when the offline computation can be performed by a server. With a server we can also speedup the simulatable check by precomputing $\mathsf{supp}(i, t)$. We experimentally measure the computational costs in Sec. 3.7.

3.7 Experiments

3.7.1 Setup

Dataset. We evaluate our system using the Reality Mining dataset [26][10] that contains continuous data on daily activities of 100 students and staff at MIT, recorded by Nokia 6600 smartphones over the 2004-2005 academic year. The trace contains various continuous information such as a user's location (at granularity of cell towers), proximity to others (through Bluetooth), activities (e.g., making calls, using phone apps), transportation mode (driving, walking, stationary), etc. over different times of the day. We consider 91 users who have at least 1 month of data. The total length of all users' traces combined is 266,200 hours. The average, minimum, and maximum trace length of a user is 122 days, 30 days, and 269 days, respectively. For each user, we train a Markov chain on

[10] `http://reality.media.mit.edu/dataset.php`.

the first half of her trace; the remaining half is used to for evaluation.

Most of our experiments use location contexts of all 91 users (as location represents the most complete and fine-grained context in the dataset). The average, minimum, and maximum number of locations per user is 19, 7, and 40, respectively. We also report an experiment with contexts based on users' activities and transportation modes to demonstrate the generality of MaskIt. This experiment is limited to 23 users for whom this information is available in the dataset.

Systems. We compare MaskIt using the simulatable check, the probabilistic check (with a granularity of $d = 10$) and the hybrid check with the naïve approach, called MaskSensitive, which suppresses all sensitive states.

Privacy Configuration. Unless otherwise stated, we choose $\delta = 0.1$. We experiment with two different ways of choosing sensitive contexts. Unless otherwise stated, we choose sensitive contexts uniformly at random for each user. Alternatively, we choose the home location of a user as the sensitive context.

Measures. We measure utility as the fraction of released states in the second half of the trace. We measure the privacy breaches as the fraction of sensitive states for which the posterior belief is more than δ larger than the prior belief. Note, that MaskIt will always assure that there are no privacy breaches. For MaskSensitive an adversary computes her posterior belief as follows: Consider

Check	initialize	okayToRelease	
	PC	PC	Phone
Simulatable	-	36 ms	128 ms
Probabilistic	15 min	< 1 ms	< 1 ms
Hybrid	18 min	≤ 36 ms	≤ 128 ms

Table 3.2: Average processing times.

a hidden Markov model defined by M with emission probabilities

$$\begin{aligned} b_{i,k}^{(t)} &= \Pr[O_t = k | X_t = c_i] \\ &= \begin{cases} 1 & \text{if } k = \perp \text{ and } c_i \in S \text{ or } k = i \text{ and } c_i \notin S \\ 0 & \text{o.w.} \end{cases} \end{aligned}$$

This hidden Markov model correctly describes the behavior of MaskSensitive. An adversary who knows M and MaskSensitive computes her posterior belief simply as $\Pr[X_t = s|\vec{o}]$ in this hidden Markov model. We can efficiently compute this posterior belief using Eq. (3.3). We say that the privacy of the user's sensitive state $s \in S$ at time t is breached by the output $\vec{o}$ of MaskSensitive if the adversary's posterior belief, $\Pr[X_t = s|\vec{o}]$, is more than δ larger than her prior belief $\Pr[X_t = s]$. We measure the privacy breaches as the number of sensitive states in the user's sequence that are breached divided by the length of the user's sequence.

Hardware. Most of our experiments are run on an Intel Xeon 2.33 GHz machine. To measure the overhead of MASKIT when run on a smart phone, we also conduct experiments on a Samsung Focus SGH-i917 phone with the Windows Phone 7.5 operating system.

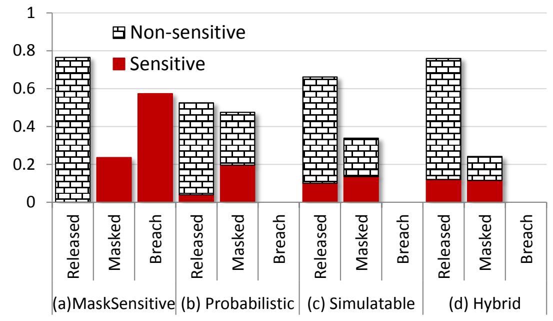


Figure 3.5: Comparison of various privacy checks

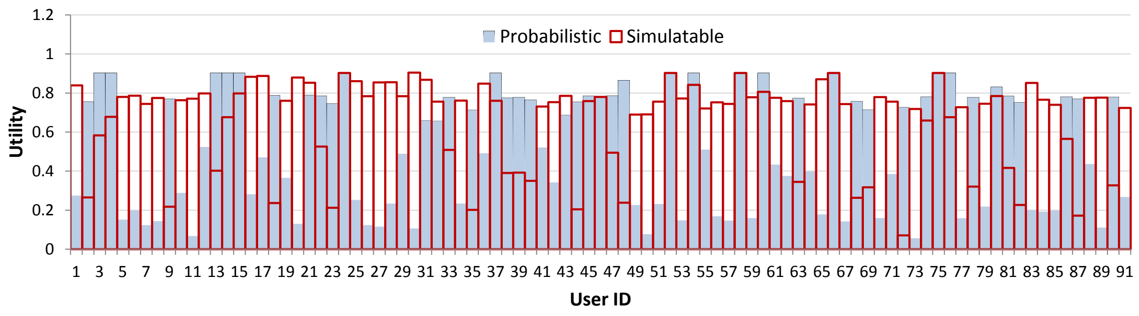


Figure 3.6: Utility of two privacy checks for various users

3.7.2 Results

Efficiency. Before we explore the privacy-utility trade-off, we want to shed light onto the efficiency of various checks. We measure the average time it takes for MaskIt to initialize the various privacy checks and to filter the trace, see Table 3.2. We note that on average the suppression decision takes at most 128 ms on the phone. If we exclude the slowest 5% of the users this average goes down to 46 ms. This is a negligible overhead compared to the context extraction time of a few to tens of seconds [8, 80].

The probabilistic and the hybrid check have an expensive initialization even after implementing the speed-up discussed in Sec. 3.4 (without which the running time would be exponential). This initialization can be offloaded to a remote server. Overall, the performance of MASKIT is practical for smart phones.

Privacy Breaches. Figure 3.5 reports results from an experiment where we choose 3 sensitive contexts for each user at random.[11] We report the average fraction of released and suppressed states by various checks. MaskSensitive suppresses the sensitive states accounting for 24% of all states. However, this does not prevent an adversary knowing the Markov chain and MaskSensitive from inferring sensitive states: 54% of the suppressed sensitive states constitute privacy breaches. For these sensitive states the adversary's posterior belief exceeds her prior belief by more than δ. This illustrates the value of having a formal privacy guarantee: With MASKIT no such privacy breaches can happen. We can see that our privacy checks suppress not just sensitive states but also non-sensitive states. (Interestingly, they manage to release some sensitive states without breaching privacy. Those are states with a high prior belief $\geq 1 - \delta$.)

The question is: what is the price in terms of utility that we have to pay for a formal privacy guarantee? As the graph shows, the probabilistic check and the simulatable check sacrifice only 31% and 13% of the utility of MaskSensitive. This appears to be a price well worth the privacy guarantee.

Hybrid.

From Figure 3.5(b) and (c), one might get the impression that the simulatable check is superior to the probabilistic check. Despite having a higher average utility across users, the simulatable check is not better for *all* users. Figure 3.6 shows the utility of both checks for each of the 91 users in our dataset. While for

[11]Recall that states specify time and context. Thus, there are a lot more than 3 sensitive states in the trace.

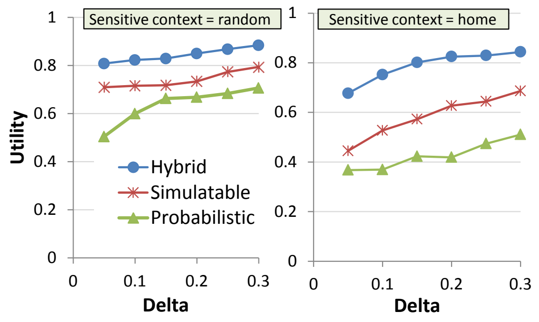


Figure 3.7: Privacy-utility tradeoff

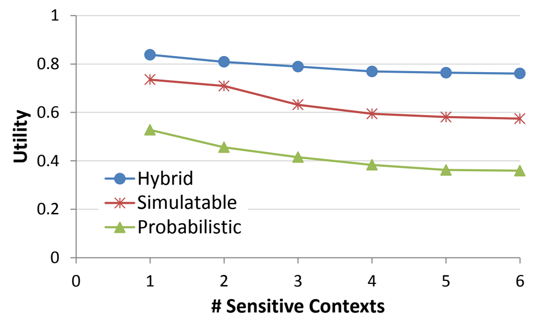


Figure 3.8: Utility for a varying number of sensitive contexts.

roughly 58% of the users the simulatable check is better, for 42% of the users the probabilistic check is better. The goal of the hybrid check is to choose the better check for each user. In our experiment, for 95% of the users the hybrid check picks indeed the check suppressing fewer states in the trace. The hybrid check makes mistakes only for users for which the fraction of suppressed states in the trace differs significantly from the expected fraction of suppressions. Note that by the law of larger numbers, for longer traces the fraction of suppressions will be more concentrated around the expectation decreasing the number of mistakes of the hybrid. Overall, the hybrid check achieves an average utility of 75.8% (see Figure 3.5(d)) which is much higher than both the utility of the probabilistic and the simulatable check and almost matches that of MaskSensitive (76.4%).

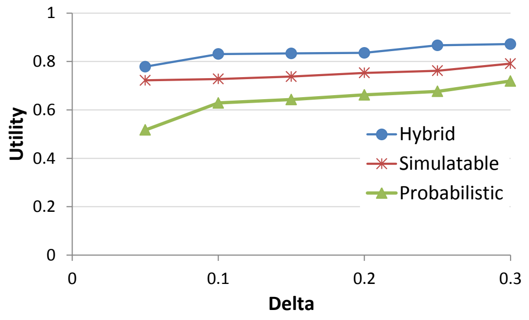


Figure 3.9: Activity contexts

In fact, our hybrid checks provides more utility than MaskSensitive when we increase the number of sensitive states (not shown). Here, unlike MaskSensitive our hybrid check releases some of the sensitive states without breaching privacy and suppresses fewer states in total. MaskSensitive provides the highest utility relative to the hybrid check when there is only one sensitive context per user; nevertheless, our hybrid check provides a utility of 84% in this case, which is within 8% of MaskSensitive's utility (91%). This shows that the price for a provable privacy guarantee is minor.

Privacy-Utility Tradeoff. We now vary the target privacy by varying the value of δ. We conduct two sets of experiments: in the first set, we choose one sensitive context for a user at random; in the second set, we choose the sensitive context for a user to be her home. When we increase privacy (by decreasing δ), we expect utility to decrease. As we can see from Figure 3.7, for both sets of experiments, the overall decrease in utility is small. This implies that we can afford strong privacy guarantees (by choosing a smaller value of δ) without sacrificing too much utility.

Varying the Number of Sensitive Contexts. In this experiment we study the effect that an increase in the number of sensitive contexts has on the utility.

We picked a subset of 25 users with a number of contexts between 15 and 20. The effect of increased sensitivity will be more (less, resp.) drastic for users with fewer (more, resp.) contexts. Figure 3.8 shows the effect on utility as the user considers more contexts sensitive. While the utility decreases for all three checks, the rates of decrease differ; the hybrid's utility decreases slowest.

Beyond Location. The experiments so far used location as a context. To show that MaskIt can operate with other types of context, we now consider user contexts that are combinations of the user's activities (making a phone call, sending sms, using an application in the phone) and her transportation mode (sitting, walking, riding a bus or car).[12] As in Figure 3.7(left), we choose a single sensitive context for each user at random. Figure 3.9 shows the privacy-utility tradeoff for the activity contexts—the results are very similar to the results for location contexts.

3.8 Related Work

Great progress has been made in the area of location-based services (LBS) to protect anonymity of users and privacy of their locations.

We compare previous work along the dimensions of privacy guarantees, efficiency, accuracy, and their applications.

Privacy issues with user location data has been studied in various applications. Gruteser and Liu [40] distinguish between applications mining a log aggregated across users, sporadic queries, and tracking services. Our work aims

[12]The transportation mode is inferred from survey responses.

to serve applications falling into the last category by providing a stream of user contexts.

Prior work offers various guarantees. The most prevalent guarantees focus around the notion of anonymity and privacy. This thesis focuses on privacy. The guarantees further differ in their assumptions about the power of adversaries. Here, the adversary does *not* know the location of all users. In fact, it is the goal to not leak information about a user being in a sensitive location to the adversary. Privacy of sensitive locations is not achieved by the widely employed naïve approach [96], that masks sensitive locations or sensitive patterns [46]. One class of adversaries is aware of the *algorithm* and can reverse-engineer the algorithm to infer information. Another class of adversaries is aware of some *temporal correlations* which is considered by [17, 35, 40, 86]. However, the knowledge of temporal correlation is either extremely limited [17, 35, 40, 86] or the technique is limited to very specific queries and does not allow to release context streams [36, 84].

Related Work in Micro-Data Publishing. The privacy concepts in LBS were inspired by privacy concepts in micro-data publishing such as k-anonymity [91] and ℓ-diversity [75, 78]. Although the nature of the data is different, challenges similar to those faced by MASKIT arise when adversaries know the system [98, 102] or correlations [101, 97, 63]. Most work assumes adversarial background knowledge in the form of facts with a few exceptions that consider adversaries who have a probabilistic model [72, 99] or infer one from the released data [62, 100] similar to the adversaries considered in MASKIT.

3.9 Conclusions

We addressed the problem of privately releasing user context streams. Our system, MaskIt, employs a privacy check that decides whether to release or suppress the current user context. We presented two privacy checks that both provably provide privacy against powerful adversaries knowing the system and temporal correlations in the stream. They differ, though, in their utility for a user. Our hybrid determines the one with the higher utility. To also protect against weaker adversaries, who can learn and become stronger, we adapted our privacy checks. Our experimental evaluation on real context traces demonstrates that we do not have to sacrifice much utility in order to guarantee privacy.

CHAPTER 4

PRIVACY-AWARE TARGETING OF ADVERTISING

4.1 Introduction

In this chapter we develop a privacy-preserving targeted advertising framework.

4.1.1 Motivation

By targeting advertisements, an advertiser can use users' contexts and activities, along with their browsing and click history, to show ads preferentially to the users who are more likely to be influenced by the ads. For example, if a user who likes Italian food (inferred from her past browsing history) is found to be walking (inferred from the accelerometer sensor data) alone (inferred from the audio data) around lunch time, she can be shown ads of popular (inferred from other users' past locations) Italian restaurants within walking distance of her current location (inferred from the GPS). Such highly targeted advertising can significantly increase the success of an ad campaign in terms of the number of resulting views, clicks or purchases on an advertised web page. This approach of context-aware personalization is not limited to advertising; it can also be used for personalizing local searches of close-by businesses.

However, such personalization raises serious privacy concerns. Personalized services rely on private and possibly sensitive information about a user's preferences and current and past activities. Such information might allow for identification of the user and her activities, and hence the user may not be willing to share information useful for personalization. In the previous example, the user may not be willing to reveal the fact that she is out of the office during business hours. Moreover, clicks on ads personalized based on private data can also leak information about the user [41, 64].

A personalized ad delivery system has three main components involving private user data: *statistics gathering* to learn personalization rules by interpreting users' contexts and clicks as implicit relevance feedback, *ad delivery* to select the best ad for a user based on her current context, and *billing* advertisers to collect money for impressions or clicks. Each component leads to privacy concerns that must be addressed to ensure end-to-end privacy. In this chapter, we focus on the second component. The first component is discussed in Chapter 5. We refer the reader to [95] for a private billing component.

4.1.2 Contributions

We first formalize the task of delivering personalized ads from a server to a client as an optimization problem with three important variables: (1) privacy, i.e. how much an adversary can learn about a user being in a sensitive context (Definition 6), (2) communication efficiency, i.e., how few ads are sent to the client, and (3) utility, i.e., how useful the displayed ads are to the user in terms of revenue and relevance. We show in Section 4.2 that it is impossible to maximize all three design goals simultaneously. We present a framework in Section 4.3 that allows to tradeoff the variables. In particular, it allows to specify constraints on privacy and communication efficiency and seeks to maximize utility. This optimization problem is NP-hard. However, we present an efficient greedy algorithm for hybrid personalization with tight approximation guarantees.

We have evaluated our framework with a large trace of location-aware searches in Microsoft Bing for mobile (Section 4.4). Our experimental results illustrate trade-offs between privacy, communication efficiency and utility and show that even though these are conflicting goals, reasonable levels of all these three goals can be achieved simultaneously.

Note that our results can be applied to personalize not just online advertising but also other online services based on users' fine-grained contextual information including local search. For concreteness, we consider advertising throughout the chapter.

4.2 The Framework

Our framework has three classes of participants: The *users* who are served ads (also referred to as clients) in their mobile contexts, the *advertisers* who pay for clicks on their ads, and the *ad service provider* (also referred to as the *server*) who decides which ads to display and who is paid for clicks by the advertisers.

►**Privacy Guarantees.** Our two components of ad delivery and statistics gathering use private data differently—ad delivery uses a user's current context, while statistics gathering uses historical context and click data from many users. For statistics gathering, we provide a version of differential privacy [24] which is a very strong privacy guarantee. However, it seems to be incompatible with personalization that requires a single user's current context. Therefore, in the spirit of many existing personalization systems and the modus operandi in many mobile applications [31, 52, 66, 105], we ensure user privacy with respect to a set of sensitive contexts by limiting what an adversary can learn about a user being in a sensitive context (following Definition 6).

The information disclosure about a user in context c can be limited by generalizing the user's context obtaining $\hat{c}$ and only sending $\hat{c}$ to the server, e.g. instead of revealing that the user is *skating* in *Central Park*, the user only discloses to be *exercising* in *Manhattan*. The generalization of context is done over a hierarchy described later. For a context c that can be generalized to $\hat{c}$ we write $c \to \hat{c}$.

To help a user define a set of sensitive contexts, we can use existing tools from the UI community (e.g. [94]). Following Theorem 2, we can protect user privacy against adversaries knowing the system by generalizing the context.

4.2.1 Desiderata for Ad Delivery

Our desiderata include goals of the individual participants as well as general system requirements. Informally, we have the three orthogonal goals: Privacy, efficiency, and revenue/relevance.

▶ **Privacy:** The system needs to ensure user privacy with respect to a set of sensitive contexts by limiting what the server can learn about a user being in a sensitive context (following Definition 6).

▶ **Efficiency:** The ad serving system should be efficient both in terms of communication and computational cost—the user wants ads fast and without draining much battery power on her mobile device, while the ad service provider wants to run her system at low operating cost. For simplicity, we focus on communication cost between the server and a client since it is the most dominant cost of serving ads to mobile devices. Our results can be extended to consider computational cost of the server and the client as well.

▶ **Revenue and Relevance:** The ad service provider seeks to maximize revenue. The user is only interested in relevant ads. The goal of the ad service provider is to display an ad from a given set of ads $\mathcal{A}$ that maximizes the expected revenue. For a click on ad a the ad service provider is being paid p_a from the advertiser. Clearly, not all users click on an ad. We denote by $\mathsf{CTR}(a|c)$ the context-dependent click-through-rate, i.e., the fraction of users who actually clicked on it in context c among those who were served the ad in context c. The expected revenue of displaying an ad a to a user in context c is $p_a \cdot \mathsf{CTR}(a|c)$. We view clicks as an indicator for relevance: users who are interested in an ad click on it. Maximizing relevance means maximizing the expected number of clicks by displaying to a user in context c the ad a with the highest context-dependent $\mathsf{CTR}(a|c)$, which is related to the goal of maximizing the expected revenue.

4.2.2 Trade-offs

Our three design parameters—privacy, efficiency, and relevance—are conflicting. Without a trusted third party, *optimizing all three design goals simultaneously is impossible*. Consider the task of showing only one ad to the user. Then, in case of maximum privacy (i.e., the user does not send any information about her context) and highest communication efficiency (i.e., the server may only return a single ad), the server needs to choose the ad without any knowledge of the user's context. Whatever the server does, yields suboptimal relevance and revenue, as long as there is an ad whose CTR depends on the context. If we want to improve the relevance, either the user needs to send some information to the server, or the server needs to send more than one ad for the user to perform personalization on the client.

If we drop any of our three design goals the problem becomes easy. If there were no concerns about privacy, we could use a *server-only* scheme, where the user sends her context c to the ad service provider, who serves the ad that maximizes the expected revenue, i.e., $p_a \cdot \mathsf{CTR}(a|c)$. This is a very efficient scheme that maximizes revenue. If there were no efficiency concerns, we could use a *client-only* scheme, where the server simply sends all ads $\mathcal{A}$ so that the user can pick the ad that maximizes expected revenue. It has been estimated that due to ad churn this requires sending 2GB of compressed ads per month [42]. Alternatively, we could use expensive cryptographic protocols for private information retrieval [32]. No user information is disclosed to the server and optimal revenue is achieved, but performance is bad. Finally, if there were no financial incentive and no interest in relevant ads, one could stop serving ads altogether to avoid any concerns regarding efficiency and privacy. In practice, one has to find reasonable trade-offs between the three design goals.

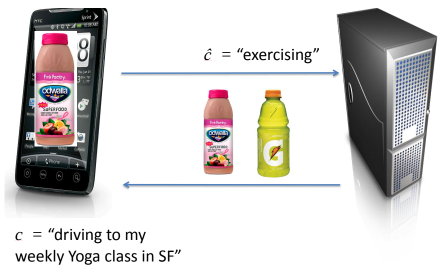


Figure 4.1: Example use of our framework.

4.2.3 Our Optimization Goal

In our framework, the user gets to decide what information about her context to share with the server. Based on this information the server selects some k ads $A \subset \mathcal{A}$ that are sent to the user. Here, the parameter k determines the communication cost. Computational cost can also be folded into k if needed. The user then picks an ad $a^* \in A$ to display. The set of ads and the ad to display should be chosen in a way that maximizes revenue. Figure 4.1 shows an example of an execution of our framework.

Our flexible framework can be optimized for various objective functions over privacy, efficiency, and revenue. For concreteness, we now assume that there are constraints on both δ-privacy (with sensitive contexts determined by users) and communication cost (determined based on current network load and response times); we seek to maximize revenue under these constraints. We will discuss alternative objective functions later in this section.

Client-Side Computation

For a given set of ads A chosen by the server, a client in context c maximizes the revenue by selecting the ad

$$a^* = \arg\max_{a \in A} p_a \cdot \mathsf{CTR}(a|c).$$

Server-Side Computation

The server needs to determine the best k ads to send to the user given only the partial information $\hat{c}$ it has. There are many possible contexts c that generalize to $\hat{c}$ the user could be in. For a context c the expected revenue is maximized by $\arg\max_{a \in A} p_a \cdot \mathsf{CTR}(a|c)$ which the server can compute assuming he knows the context-dependent click-through-rates. Suppose further that the server knows the frequency of each context. Then the server can reason about the probability with which the user is in a context c for all descendant leaves $c \to \hat{c}$. The expected revenue is then the sum of the expected revenue of the descendant leaves c weighted by their probability $\Pr[c|\hat{c}]$. Hence, with this limited information the expected revenue of a set of ads A for a generalized context $\hat{c}$ is

$$\mathsf{E}[\mathsf{Revenue}(A|\hat{c})] = \sum_{c : c \to \hat{c}} \Pr[c|\hat{c}] \cdot \max_{a \in A} p_a \cdot \mathsf{CTR}(a|c).$$

It is the server's task to select the set A^* of k ads from $\mathcal{A}$ that maximizes the expected revenue, given only the generalized context $\hat{c}$ of the user, i.e.,

$$A^* = \arg \max_{A \subset \mathcal{A} : |A| = k} \mathsf{E}[\mathsf{Revenue}(A|\hat{c})]$$

Finding these k ads is NP hard as we will show in the next section. However, we can employ approximation techniques to efficiently select a set of k ads with revenue close to the optimal revenue.

Instantiations of the Framework

Our framework encompasses client-only personalization by setting $\hat{c}$ to the most generalized context that does not leak any information about the client's true context. In this case the personalization takes place exclusively on the client side. Our framework also encompasses server-only personalization by setting $k = 1$ in which case the client simply displays the ad sent by the server without further personalization. However, higher revenue can be achieved in our framework when the server sends back $k > 1$ results.

Extensions

Alternative Objective Functions. So far, we treated both the privacy and the communication cost k as hard constraints and tried to maximize revenue under these constraints. Instead, one might consider the communication cost as a variable and include it in an objective function that maximizes the value of (revenue $-\alpha k$).

Additional Constraints. While high revenue and high relevance of ads are related goals, they are not the same. Suppose the ad service provider receives a request from a user in context c. Suppose further there are two ads a_1, a_2 with $\mathsf{CTR}(a_1|c) = 0.1, \mathsf{CTR}(a_2|c) = 0.9$ and $p_{a_1} = \$0.1, p_{a_2} = \$0.01$. Ad a_1 has higher expected revenue but a_2 is more relevant. While displaying a_1 maximizes short-term revenue it might not be the best long-term strategy. Recent work has found that the time users spend viewing ads depends on the predictability of the quality of the ads [13]. Our framework can reconcile relevance and short-term and long-term revenue goals by adding a constraint on CTRs of displayed ads as proxies for relevance.

4.3 Optimization Algorithms

In this section we explain how client and server can efficiently compute their parts of the optimization. We consider a specific instantiation of the optimization problem where the user fixes her privacy requirement; the client and the server then try to maximize revenue for a given bounded communication complexity k. At the end of the section we discuss extensions and alternatives.

The client can quickly compute the equation in Sec. 4.2.3, since the number of ads from which the client picks one is small ($\leq k$).

However, the server's task—to select a set of k ads from $\mathcal{A}$ that maximize the expected revenue given only the generalized context $\hat{c}$ of the user—is much more demanding. In fact, a reduction from the maximum coverage problem shows:

Proposition 6. *For a generalized context $\hat{c}$ it is NP-hard to select the revenue-maximizing set of k ads A^* such that:*

$$A^* = \arg \max_{A \subset \mathcal{A} : |A| = k} \sum_{c : c \to \hat{c}} \Pr[c|\hat{c}] \cdot \max_{a \in A} p_a \cdot \mathsf{CTR}(a|c)$$

Proof. We prove this by a reduction from the maximum coverage problem. In the maximum coverage problem we are given a collection of sets $\mathcal{S}$ over some finite universe U and our goal is to find a subset of these sets S of size at most k that maximizes the number of covered elements in the universe $\left|\bigcup_{s \in S} s\right|$.

We set the set of contexts equal to the universe U. For each context $c \in U$ we set $\Pr[c] = 1/|U|$, and $\Pr[c|\hat{c}]$ therefore to be uniform across the descendant leaves. For each set s in the collection $\mathcal{S}$ we create an ad a so that for the elements c in s we set $\mathsf{CTR}(a|c) = 1$ and 0 for all other elements. We set $p_a = 1$ for all ads.

```
Greedy(ads $\mathcal{A}$, generalized context $\hat{c}$, threshold $k$)
Init $A = \emptyset$
**while** $|A| < k$ **do**
    **for** $a \in \mathcal{A}$ **do**
        $b_a \leftarrow \mathsf{E}[\mathsf{Revenue}(A \cup \{a\}|\hat{c})] - \mathsf{E}[\mathsf{Revenue}(A|\hat{c})]$
    **end for**
    $A \leftarrow A \cup \{\text{argmax}_a b_a\}$
**end while****return** $A$.
```

Algorithm 6: Greedy algorithm for selecting ads maximizing the expected revenue.

Consider a generalized context $\hat{c}$ that generalizes all contexts c. Then for a given set A of k ads, the expected revenue is $1/|U| * \sum_c \max_{a \in A} \mathsf{CTR}(a|c)$, which is equal to the number of covered elements by the corresponding set of sets S divided by the universe size. Thus an optimal solution to the ad selection problem yields an optimal solution to the maximum coverage problem. $\square$

Moreover, the maximum coverage problem cannot be approximated within $\frac{e}{e-1} - o(1)$ assuming $P \neq NP$ [30].

4.3.1 Approximation Algorithm

Algorithm 6 shows a greedy algorithm, called Greedy, that constructs a set A of k ads incrementally. It starts with A empty and in each round, the ad that increases the expected revenue the most is added to A.

Interestingly, the output of this simple greedy algorithm approximates the optimal value to within a factor of $(1 - 1/e)$. The greedy algorithm is known to provide such a guarantee for the maximum coverage problem [49], but our problem is considerably more complex: In the coverage problem a set either *fully* covers an element or not. In our case an ad a can *partially* "cover" a context c that can be generalized to $\hat{c}$. Thus a new analysis is required. We extend the analysis of Hochbaum et al. [49]. We first define a *benefit function* of adding a set A' to a set A:

$$\mathsf{B}(A, A') = \mathsf{E}[\mathsf{Revenue}(A \cup A'|\hat{c})] - \mathsf{E}[\mathsf{Revenue}(A|\hat{c})].$$

We analyze the benefit function

Fact 3. *The benefit function is submodular, i.e., for all sets of ads $A_1 \subseteq A_2$ and for all A, $B(A_1, A) \geq B(A_2, A)$.*

Proof. Fix an exact context c. $A_1 \subseteq A_2$ implies that

$$\mathsf{E}[\mathsf{Revenue}(A_1|c)] \leq \mathsf{E}[\mathsf{Revenue}(A_2|c)] \tag{4.1}$$

Now consider an ad a' that is added to both sets. If a' is the $\arg\max_{a \in A_2 \cup \{a'\}} p_a \cdot \mathsf{CTR}(a|c)$ then sdf,

$$\mathsf{E}[\mathsf{Revenue}(A_2 \cup \{a'\}|c)] = p_a \cdot \mathsf{CTR}(a'|c) = \mathsf{E}[\mathsf{Revenue}(A_1 \cup \{a'\}|c)] \tag{4.2}$$

Equations 4.1 and 4.2 imply $\mathsf{B}(A_1, \{a'\}) \geq \mathsf{B}(A_2, \{a'\})$. On the other hand, if a' is *not* the $\arg\max_{a \in A_2 \cup \{a'\}} p_a \cdot \mathsf{CTR}(a|c)$, then $\mathsf{E}[\mathsf{Revenue}(A_2 \cup \{a'\}|c)] = \mathsf{E}[\mathsf{Revenue}(A_2|c)]$. Therefore, the benefit $\mathsf{B}(A_2, \{a'\})$ is 0, which is a lower bound of the function B and hence at most $\mathsf{B}(A_1, \{a'\})$. Thus the submodularity holds for adding the singleton set $\{a'\}$. Submodularity for adding a larger set A follows by rewriting the benefit of adding A as the sum of adding each ad $a \in A$ one-by-one. Finally, submodularity follows for a generalized context $\tilde{c}$ since its revenue is a sum of positively weighted revenues of exact contexts. □

However, due to the complex nature of our problem, the submodularity property alone does not imply our approximation guarantee.

Let $a_1, \ldots, a_k$ be the k ads chosen by Greedy in the order they were chosen. To simplify the analysis, we define the benefit of the i^{th} ad to be b_i and the expected revenue after adding the first l ads to be $b(l) = \sum_{i=1}^{l} b_i$. Similarly, let $a_1^*, \ldots, a_k^*$ be the k optimal ads in any fixed order. We define the benefit of the i^{th} ad to be b_i^* and the expected revenue after adding the first l ads to be $b^*(l) = \sum_{i=1}^{l} b_i^*$.

Lemma 10. *$\forall l \in [k]$: $b_l \geq \frac{b^*(k)-b(l-1)}{k}$.*

Proof. The benefit of adding $a_1^*, \ldots, a_k^*$ to $a_1, \ldots, a_{l-1}$ is at least $b^*(k) - b(l-1)$:

$$\begin{aligned}
&\mathsf{B}(\{a_1, \ldots, a_{l-1}\}, \{a_1^*, \ldots, a_k^*\}) \\
&= \mathsf{E}[\mathsf{Revenue}(\{a_1, \ldots, a_{l-1}\} \cup \{a_1^*, \ldots, a_k^*\}|\hat{c})] - \mathsf{E}[\mathsf{Revenue}(\{a_1, \ldots, a_{l-1}\}|\hat{c}) \\
&\geq \mathsf{E}[\mathsf{Revenue}(\{a_1^*, \ldots, a_k^*\}|\hat{c})] - \mathsf{E}[\mathsf{Revenue}(\{a_1, \ldots, a_{l-1}\}|\hat{c}) \\
&= b^*(k) - b(l-1)
\end{aligned}$$

It is also equal to $\sum_{i=0}^{k} \mathsf{B}(\{a_1, \ldots, a_{l-1}\} \cup \{a_1^*, \ldots, a_{i-1}^*\}, \{a_i^*\})$. Thus, it follows from an averaging argument that $\exists i, 1 \leq i \leq k : \mathsf{B}(\{a_1, \ldots, a_{l-1}\} \cup \{a_1^*, \ldots, a_{i-1}^*\}, \{a_i^*\}) \geq \frac{b^*(k)-b(l-1)}{k}$. By submodularity this implies that

$$\exists i, 1 \leq i \leq k : \mathsf{B}(\{a_1, \ldots, a_{l-1}\}, \{a_i^*\}) \geq \frac{b^*(k) - b(l-1)}{k}.$$

Since the greedy algorithm in round l selected the ad a_l that maximizes $\mathsf{B}(\{a_1, \ldots, a_{l-1}\}, \cdot)$, the benefit of that ad, b_l, has to be at least $\frac{b^*(k)-b(l-1)}{k}$ which completes the proof. □

We use this lemma to show:

Lemma 11. *$\forall l \in [k]$: $b(l) \geq (1 - (1 - 1/k)^l) b^*(k)$.*

Proof. Proof by induction on l.

$l = 1$. Lemma 10 tells us that $b_1 \geq \frac{b^*(k)}{k} = (1 - (1 - 1/k)^1)b^*(k)$.

$l \to l + 1$.

$$\begin{aligned}
b(l+1) &= b(l) + b_{l+1} \\
&\geq b(l) + \frac{b^*(k) - b(l)}{k} && \text{Due to Lemma 10} \\
&= \frac{b^*(k)}{k} + b(l)(1 - 1/k) \\
&\geq \frac{b^*(k)}{k} + (1 - (1 - 1/k)^l)b^*(k)(1 - 1/k) && \text{By induction hypothesis} \\
&= \frac{b^*(k)}{k} + ((1 - 1/k) - (1 - 1/k)^{l+1})b^*(k) \\
&= (1 - (1 - 1/k)^{l+1})b^*(k)
\end{aligned}$$

□

The main theorem on the approximation guarantee follows.

Theorem 7. *The greedy algorithm approximates the optimal value to within a factor of $(1 - 1/e)$.*

Proof. By Lemma 11 we have that $b(k) \geq (1 - (1 - 1/k)^k)b^*(k)$. As a function of k, $(1 - (1 - 1/k)^k)$ is decreasing and approaches $(1 - 1/e)$. Hence, it is always greater than $(1 - 1/e)$. □

4.3.2 Extensions

Alternate Objective Functions. As mentioned in Section 4.2.3, an objective function may include communication cost as well. We then seek a number k

and a set of k ads A that maximize $\mathrm{E}[\mathsf{Revenue}(A)] - \alpha k$, i.e.,

$$\max_k \max_{A \subset \mathcal{A}: |A|=k} \sum_{c: c \rightarrow \hat{c}} \Pr[c|\hat{c}] \cdot \max_{a \in A} p_a \cdot \mathsf{CTR}(a|c) - \alpha k \tag{4.3}$$

One straightforward way to handle such an objective function is to run Greedy for all values of k and pick the outcome that maximizes our new objective function. We refer to this algorithm as Greedy$'$. We only need to run it once for $k = |\mathcal{A}|$ and keep track of the order in which the ads are chosen $a_1, \ldots, a_{|\mathcal{A}|}$. Due to the greedy nature of the algorithm, we have that running the greedy algorithm for smaller values of k yields a prefix of length k: $a_1, \ldots, a_k$.

Proposition 7. *Let k^*, A^* denote the optimal solution of Equation (4.3). The algorithm* Greedy$'$ *runs* Greedy *with $k = |\mathcal{A}|$ obtaining $a_1, \ldots, a_{|\mathcal{A}|}$ and outputs $k' = \arg\max_{k'} E[\mathsf{Revenue}(\{a_1, \ldots, a_{k'}\})] - \alpha k'$ and $\{a_1, \ldots, a_{k'}\}$. Algorithm* Greedy$'$ *has the following approximation guarantee:*

$$(1 - 1/e)\mathsf{Revenue}(A^*) - \alpha k^* = \mathsf{Greedy}'$$

Proof. By Theorem 7 we have that $(1-1/e)\mathsf{Revenue}(A^*) \leq \mathsf{Revenue}(\{a_1, \ldots, a_{k^*}\})$. Thus,

$$\begin{aligned}(1 - 1/e)\mathsf{Revenue}(A^*) - \alpha k^* &\leq \mathsf{Revenue}(\{a_1, \ldots, a_{k^*}\}) - \alpha k^* \\ &\leq \max_k \mathsf{Revenue}(\{a_1, \ldots, a_k\}) - \alpha k \\ &= \mathsf{Greedy}'\end{aligned}$$

completing the proof. □

We can further reduce the number of iterations in the Greedy algorithm. All we have to do is to replace the **while** condition in Algorithm 6 by a new one that checks whether the current value of $\mathrm{E}[\mathsf{Revenue}(A)] - \alpha \cdot |A|$ is increasing. This speed-up exploits the submodularity of the benefit function.

Theorem 8. *Let k^*, A^* denote the optimal solution of Equation (4.3). The algorithm* Greedy'' *runs* Greedy *for as long as $E[\mathsf{Revenue}(A)] - \alpha \cdot |A|$ is increasing and outputs the last set of ads A. Algorithm* Greedy'' *has the following approximation guarantee:*

$$(1 - 1/e)\mathsf{Revenue}(A^*) - \alpha k^* = \mathsf{Greedy}''$$

The number of iterations of the **while***-loop of* Greedy'' *is at most $1 + \arg\max_{k'} E[\mathsf{Revenue}(\{a_1, \dots, a_{k'}\})] - \alpha k'$.*

Proof. Using Proposition 7 it suffices to show that $\mathsf{Greedy}'' = \mathsf{Greedy}'$. We argue that as we increase k, the expected revenue of $\{a_1, \dots, a_k\}$ increases until at some point it starts to decrease and never increases again. Suppose in round k' the expected revenue of $A = \{a_1, \dots, a_{k'}\}$ minus $\alpha \cdot k'$ is not increasing any longer, i.e.,

$$\mathrm{E}[\mathsf{Revenue}(\{a_1, \dots, a_{k'}\})] - \alpha k' \leq \mathrm{E}[\mathsf{Revenue}(\{a_1, \dots, a_{k'-1}\})] - \alpha(k' - 1).$$

At this point the benefit of adding $a_{k'}$ is at most α. Due to submodularity, the benefit of any future ad being added to A can only be smaller and thus will never lead to an increase of the objective function. This means that Greedy'' finds indeed $k' = \arg\max_{1 \leq k' \leq |\mathcal{A}|} \mathrm{E}[\mathsf{Revenue}(\{a_1, \dots, a_{k'}\})] - \alpha k'$ and $\{a_1, \dots, a_{k'}\}$. Hence, the output of Greedy'' is the same as that of Greedy'.

Note, that the number of iterations of Greedy' is $|\mathcal{A}|$ while that of Greedy'' is only at most one larger than the output k'. □

Additional Constraints. We can incorporate a constraint on ad relevance by setting the CTR to zero whenever it is below a certain threshold. Then, no ad with CTR below this threshold will ever be displayed at the client.

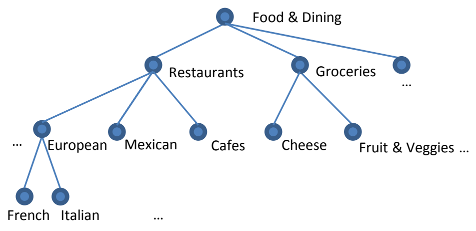


Figure 4.2: Hierarchy over businesses.

Advertisers' Control. Our algorithm can incorporate additional restrictions posed by advertisers on the contexts in which their ads are being displayed. Very much like advertisers for sponsored results in Web search can bid on keywords in a query, our advertisers can bid on contexts of users. To make sure the ad is only displayed on these contexts, we can make the payment p_a context-dependent and set it to 0 for all but the contexts the advertiser bids on.

4.4 Experiments

4.4.1 Experimental Setup

Dataset. Ideally, we would like to evaluate our algorithms with real usage logs from a context-aware ad service. However, since no such real systems exist, we emulate such a system by using a log of location-aware searches in Microsoft Bing for mobile.[1] The log has a schema: ⟨*user-ID, query, user-location, business-ID*⟩. Each record in the log describes an event of a user issuing a query from a location and then clicking on a business. The log consists of $1{,}519{,}307$ records. In our evaluation we focus on clicks to "Food & Dining" businesses, which con-

[1] `http://m.bing.com`

stitute the largest category of businesses in the log. We also filter out any user with fewer than three clicks in the log, as we cannot generate an interest profile for such a user. This leaves us with $116{,}432$ unique user-IDs. We use the first 90% of the log as training data and the remainder to evaluate our framework and to compute targeted ads (i.e., businesses).

Context. We use the above log to emulate a context-aware ad service as follows. We assume that each business with id i has an ad with the same id i, and hence our goal is to deliver target business-IDs to the users. Ideally, we would like to use contexts extracted from the sensor readings of smart phones for personalization. However, this information is not present in our log and we therefore limit our evaluation to contexts consisting of the following set of attributes.

▶ Location: The user's location as latitude and longitude.

▶ User Interest: A multi-set of the business-IDs the user clicked on previously.

▶ Query: The search query the user sends.

Attribute Generalization. To limit information disclosure, we let users generalize context attributes according to fixed hierarchies.

▶ Location: We use five levels of generalization for user location, depending on how many decimal points we truncate from her latitude and longitude. More specifically, `Level-i` location, for $i = 0, 1, 2, 3, 4, 5$ of a user is her latitude and longitude, after keeping all, 4, 3, 2, 1, and 0 decimal points respectively.

▶ Interest: We generalize user interest using a fixed hierarchy for the businesses, as shown in Figure 4.2. In `Level-0`, `Level-1`, and `Level-2`, the interest set contains business categories, generalized business categories, and only the most general business category ("Food and Dining"), respectively, of the user's clicks.

▶ Query: Again, we use the business hierarchy to generalize the query at three levels. `Level-0` is the exact query issued by the user, `Level-1` is the business category of the clicked business, and `Level-2` is the generalized category of the business.

For all attributes, `Level-i` is more general, and hence more privacy-preserving, than `Level-j` for $i > j$. As a short-hand, we use (x, y, z) to denote (`Level-`x location, `Level-`y interest, `Level-`z query).

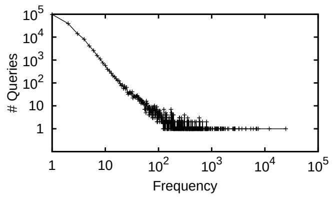
Context Hierarchy. We combine the attribute hierarchies into a context hierarchy. We generalize one attribute at a time using the following sequence: $(0,0,0) \rightarrow (0,0,1) \rightarrow (0,1,1) \rightarrow (1,1,1) \rightarrow (1,2,1) \rightarrow (2,2,1) \rightarrow (3,2,1) \rightarrow (3,2,2) \rightarrow (4,2,2)$. As an example, consider the context at level $(0,0,0)$

$\langle$(61.22913, -149.912044), [B-ID2011, B-ID124], "Starbucks"$\rangle$.

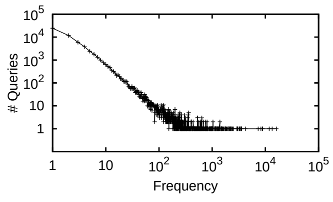
Generalizing each attribute one level yields, at level $(1,1,1)$,

$\langle$(61.2291, -149.9120), [Peruvian Restaurants, Wine], "Coffee"$\rangle$.

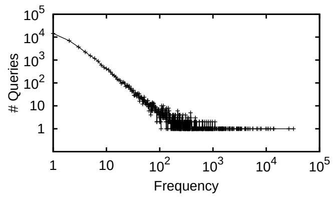
Click-Through-Rates. The optimization framework requires the distribution of contexts, $\Pr[c]$, and the context-dependent click-through rates, $\mathsf{CTR}(a|c)$. $\Pr[c]$ can be estimated as the number of times a user reported to be in context c divided by the total number of reported contexts. For an ad a and a context c, $\mathsf{CTR}(a|c)$ can be estimated as the number of times a user in context c reported to have clicked on a divided by the number of times a user in context c reported to have viewed a.



(a) Exact query



(b) `Level-1` gen.



(c) `Level-2` gen.

Figure 4.3: Effect of generalization of queries

Note that the above statistics can be estimated well for contexts with a lot of user data (e.g., clicks). However, we consider context attributes beyond location (unlike location-based services) and one big challenge is that sufficient click data may not be available for a large number of contexts. For such rare or new contexts, the estimates can be noisy or may not exist. One option would be to simply not show any ads to a user in a rare context. However, this would seriously harm utility since there is typically a long tail of rare contexts. Instead, we suggest estimating $\Pr[c]$ and $\mathsf{CTR}(a|c)$ for a rare context c based on contexts similar to c for which we have enough click data. Here similarity is defined through the context hierarchy. Coming back to our example from Section 4.2, if we do not have enough click data for users who were skating in Central Park, we might use clicks from users in close-by locations who were doing some sort of physical activity ($\tilde{c}$) to estimate the statistics for the context c. This helps us increase coverage of targeted ads, albeit at the possible cost of lower quality ads. We can trade-off coverage and relevance by adding a constraint on CTR for displayed ads as discussed in Sections 4.2.3 and 4.3.2.

We show how generalization helps personalization with sparse data. Figure 4.3 shows the frequency distributions in log-log scale of queries (Figure 4.3(a)), and their generalizations (Figures 4.3(b) and (c)), in our dataset. The query distribution has a power-law shape in which a small fraction of unique queries account for a large fraction of the query log. We can see that roughly $100{,}000$ queries appear only once each in our data. For these queries it is impossible to personalize the search results because we have not seen the same query before. However, if we generalize queries to the categories of the businesses they are referring to we can reduce this number by an order of magnitude. Similarly, we can deal with the sparsity of the other context attributes by generalizing them. This is how we increase coverage.

Metrics. We use the following two metrics for our prediction.

▶ **Precision:** The fraction of targeted ads in our framework on which users actually click. Precision is an indicator of relevance.

▶ **Coverage:** The fraction of contexts for which our framework computes and displays a targeted business.

The higher the precision and coverage values, the better the performance of our framework. We report average precision and coverage for $1{,}000$ random contexts from the testing data; the averages become fairly stable after $1{,}000$ predictions.

Parameters. Unless otherwise stated, we use the following default configuration. For limited information disclosure, we use $(4, 2, 2)$ generalization. We set the upper bound on communication complexity, k, to be 10, the threshold on click-through rate to be 0.3, and the threshold on support to be 2.

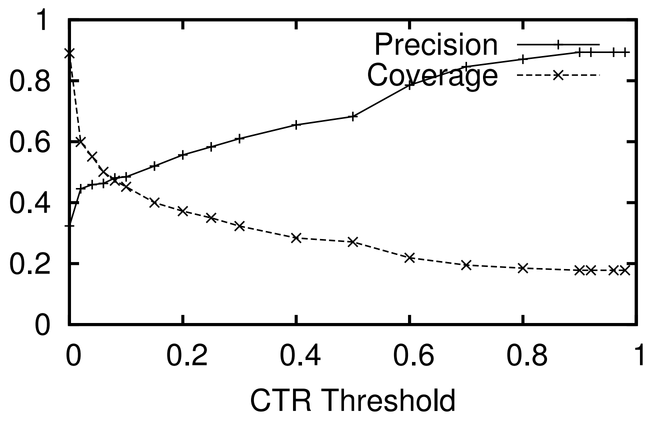


Figure 4.4: Varying the minimum CTR.

4.4.2 Evaluating Trade-offs

Effect of CTR Threshold. The CTR threshold trades off precision and coverage. Figure 4.4 shows this trade-off. For a high value of the CTR threshold, an ad will be shown only if it is highly relevant. Thus, this increases the precision of our algorithm and improves the relevance of the displayed ads. On the other hand, a high threshold reduces the number of ads displayed and with that the number of clicks and the revenue. Interestingly, as we can see, *high levels of both precision (0.48) and coverage (0.47) can be achieved simultaneously*.[2]

Effect of Communication Complexity. Figure 4.5 shows the effect of increasing the communication complexity k (i.e. having the server return more ads to the client) on precision and coverage. We expect both to improve with increasing k since the client can choose an ad from a larger set. The graph shows further that *increasing k has diminishing returns*. In the beginning the precision and coverage increase quickly with every additional ad being sent, however, as more ads are sent, the increase in precision and coverage becomes smaller.

[2]Precisions and coverages close to 0.5 are considered high in predicting user clicks. Our numbers are higher than the ones reported for other personalization techniques [107].

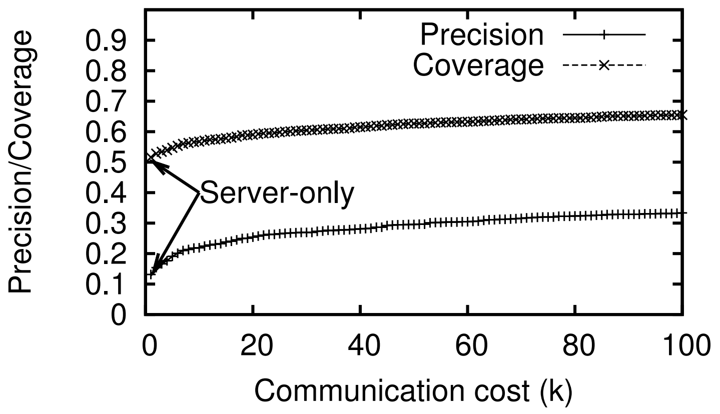


Figure 4.5: Varying communication cost.

Effect of Information Disclosure. Figure 4.6 shows the precision and coverage (for various CTR thresholds) of our framework with various levels of generalization. As expected, precision and coverage of our framework increases as more specific context information is sent to the server. Interestingly, *limited privacy does not hurt utility in a significant way*; as shown in the graph, precision and coverage values are very close for limited privacy (shown as $(1, 1, 1)$) and no privacy (shown as $(0, 0, 0)$).

Trading-off Constraints. To see how communication overhead affects the performance of our framework, we increase k from 10 to 50 in Figures 4.6(a) and (b). The graphs show that *privacy can be improved without hurting utility by a small increase in the communication cost*. For example, when $k = 10$, a privacy level of $(4, 2, 2)$ does not achieve a precision of at least 0.85 and a coverage of at least 0.3. But it does, when increasing k to 50. Overall, we conclude that reasonable levels of limited information disclosure, efficiency, and relevance can be achieved simultaneously.

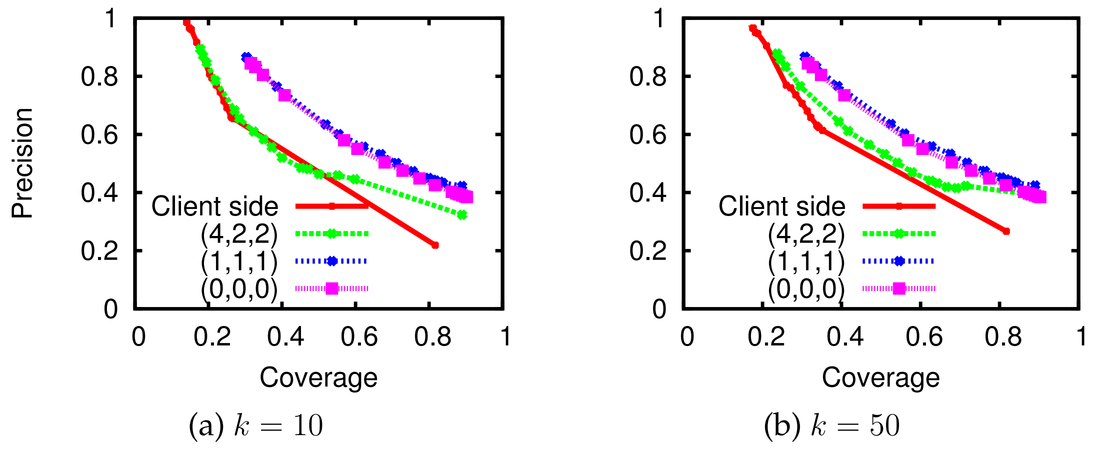


Figure 4.6: Varying information disclosure.

4.4.3 Comparison with Other Strategies

Server-only Personalization. Here, the server performs personalization based on the limited private information it has and sends only one ad to the client. As shown in Figure 4.5, this strategy gives a precision of 0.12. We can do much better with our optimization: When instead sending 5 ads and letting the client pick the most relevant one, the precision rises by 35%.

Client-only Personalization. Here, the client sends only the query to the server, which then sends k ads matching the query to the client. The client chooses the best ad based on the exact user context. Precision and coverage of this strategy are also shown in Figure 4.6 with the label "Client-side". As shown, our optimization can provide better utility than the client-only strategy. For example, for a target precision of 0.75, the client-side strategy can achieve coverage of 0.2, while our framework with $(1, 1, 1)$ generalization can achieve a coverage of 0.4.

4.5 Related Work

Personalization has been successfully implemented in a varieties of areas including Web search and advertising. Recent work has raised privacy concerns about personalization based on private information because a click on an ad or Web page can leak some private information about the user [41, 64]. Therefore, privacy-preserving personalization has recently received a lot of attention not just in the academic community but also in the media.[3]

Location-Based Services. Most privacy-preserving location-based services (e.g., [81]) follow a hybrid approach in which, given a generalized context, the server returns a superset of the results [52], which can lead to high communication cost. Notable exceptions approximate the results and allow to trade off efficiency and accuracy [21, 59, 108]. We follow the same goals of LBS: privacy, utility and efficiency. While LBS focus on nearest neighbor queries measuring utility as proximity, our work focuses on target advertisements measuring utility as revenue or ad relevance. Previous techniques cannot be applied to this problem.

Targeted Advertising. Before we discuss previous work on privacy-aware targeted advertising, let us briefly review how existing search engines personalize ads to be displayed alongside a search result. Advertisers bid money on keywords. For an incoming query, the subset of ads bidding on keywords in the query is determined. From this subset the ads with the highest bids multiplied by their quality score are chosen. The most important factor of the quality score is the click-through-rate. The context considered encompasses the exact query

[3]`http://www.nytimes.com/2010/10/23/technology/23facebook.html`

and also geographic information available from the user submitting the query. Other factors of the quality score are the quality of the landing page and that page's loading time.[4] Our work builds on this approach by incorporating private data from sensors on mobile phones into the context and adding privacy guarantees to the overall scheme.

Closest to our privacy-aware ad serving framework are the works of [31, 42, 55]. Repriv [31] verifies that applications only access the limited information about a user that was granted and proposes techniques for client only personalization. Privad [42] and the work by Juels [55] anonymize user profiles. Neither work explains how ads should be chosen based on limited user information by the ad server and based on more private information on the client. Thus, there is a potential benefit of integrating our framework into these systems to trade off privacy, efficiency and utility.

Personalized Search. A user's interest profile is established based on her browsing history and search results are being re-ranked based on how well the content of the page matches her interest. The re-ranking can either be done by the user [92] or the search engine [66, 105]. If the re-ranking is done by the search engine, then users can decide how much information they are willing to share about their profile (by using techniques in [66, 105]). Our approach to personalization is very different than these existing approaches: we personalize at the granularity of contexts, instead of individual users (i.e., the same user will see different results/ads under different contexts) and we personalize jointly at the server and the client.

[4]See for instance `http://adwords.blogspot.com/2008/08/quality-score-improvements.html`

CHAPTER 5

A DISTRIBUTED PROTOCOL FOR GATHERING PRIVACY-PRESERVING AGGREGATE STATISTICS OF MOBILE USERS

In this chapter we present an efficient distributed count protocol, that preserves privacy.

This protocol can be used to learn how to target ads. Personalized ads are chosen based on historical information about which ads users in a context clicked on, i.e., context-dependent click-through rates or CTRs of ads. However, estimating CTRs constitutes a big privacy challenge: users are often unwilling to reveal their exact context and clicks. We need to address this challenge in order to ensure end-to-end privacy of the overall ad service.

One might use a privacy-preserving aggregation protocol [23, 89, 93] to compute such statistics. These protocols do not rely on a trusted server and preserve privacy even if some of the users are malicious. However, a unique aspect of dealing with a large population of mobile users is that a small fraction of users can become unavailable *during* the course of computing CTRs. For example, a user may turn off her mobile device any time or may want to answer an aggregation query only at a convenient time when her phone is being charged and connected through a local WiFi network. Another user might decline to participate in the exchange of certain messages in the protocol. Yet another user might leave or join the community of mobile users. Existing protocols [23, 89, 93] do not efficiently handle such dynamics (more details in Section 5.4), making them unsuitable for estimating CTRs from mobile users. Then, *how can we, in an efficient and privacy-preserving way, gather statistics over a dynamic population?*

We answer this with a novel aggregation protocol to compute CTRs without a trusted server that can handle dynamics. To the best of our knowledge, our protocol is the first differentially-private protocol that computes accurate aggregations efficiently even when a fraction of participants become unavailable or behave maliciously.

5.1 Desiderata

We have the following goals in computing statistics.

▶ **Privacy in the Absence of a Trusted Server.** We do not assume the availability of a trusted server to collect all user data to compute statistics. Without a trusted server, we need a distributed aggregation protocol that protects user privacy, even under adversarial scenarios such as when a fraction of the participants behave maliciously, send bogus messages, or collude with each other. In particular, we seek (ϵ, δ)-probabilistic differential privacy of the distributed protocol (Definition 5). This requirement sets our work apart from previous work on publishing privacy-preserving count statistics of a search log that all assume a trusted third party (see [12] and the references therein).

▶ **Scalability.** We need to scale the computation to a large number of users and contexts.

▶ **Robustness to a Dynamic User Population.** With a large number of transient mobile phone users, not all of them are available and willing to engage in all rounds of our protocol. Users decide which queries they are willing to answer and when (e.g., when the phone is being charged and connected through a WiFi network). Therefore, our protocol should be able to deal with a dynamic user population without sacrificing privacy or scalability.

5.2 Preliminaries and Assumptions

The main mechanism we employ to build a scalable and robust protocol is to use two servers: one responsible for key distribution and the other responsible for aggregation. For concreteness, the key distribution could be done by verisign and the aggregation by the advertising network. The idea of using two servers to build secure protocols has been used previously [2, 32, 42] in different applications; we use it here for privacy-preserving aggregation. We assume secure, reliable, and authenticated communication channels between servers and users. In addition, we make the following two key assumptions, similar to those made in previous works [89, 93].

1. Honest-but-Curious Servers. *The two servers honestly follow the protocol. They are curious but do not collude with anyone.*

2. Honest Fraction of Users. *At most a t fraction of users are malicious or unavailable during the protocol. This means, at least a fraction of $1 - t$ users are available and honestly follow the protocol. The honest users can be curious but they do not collude with anyone.*

The distributed sum protocol relies on a homomorphic encryption scheme. We here work with an ideal model of the encryption scheme. In the following analysis we assume perfect security, i.e. that the ciphertext leaks no information at all about the message it encrypts. Technically, though, our privacy guarantee now depends on the computational limits of the adversary and the assumptions used to prove the encryption scheme secure.

Definition 9. *Let (E, D) be a homomorphic encryption scheme. This scheme is* correct, *i.e., for all plain texts $p, p' : D(E(p)) = p$. It is* secret, *i.e.,* $\Pr[E(p) = m] = \Pr[E(p') = m]$. *Lastly, it is homomorphic, i.e., $E(p) + E(p') = E(p + p')$.*

5.3 Previous Work

Previous work on distributed counting protocols [23, 89, 93] provides strong privacy guarantees. However, they are inefficient when the user population changes quickly. This is problematic in our setting with a large number of transient mobile users.

Early work by Dwork et al. [23] exchanges a large number of messages that is quadratic in the number of users. This is prohibitively expensive in our setting. Follow-up work by Rastogi et al. [89] and Shi et al. [93] reduces the number of messages to be linear in the number of users. Briefly, both the protocols start with a setup phase in which an honest server generates secrets and distributes them to users such that the secrets of all users add up to a constant (e.g., 0 as in [93]). After this setup phase, a series of count queries can be computed in a privacy-preserving manner assuming that all available users in the setup phase participate in the aggregation phase. However, when a single user becomes unavailable, no further queries can be answered until a new setup is performed or the user returns. Thus, for a query to be successfully answered, the setup phase followed by the aggregation phase must be repeated until they both run on the same stable set of users. In Section 5.6.2, we empirically show that even under

Protocol 1 Robust, distributed count computing a privacy-preserving version of the sum over all private user bits b_i

```
Count(σ², t)
  1. Each user $i$ samples $r_i$ from $\mathcal{N}(\sigma^2/((1-t)N-1))$.
  2. Each user $i$ sends $m_i = E(b_i + r_i)$ to Server 2.
  3. Server 2 sums up all incoming messages $m_i$. It forwards $S = \sum m_i$
     to Server 1.
  4. Server 1 decrypts $s$ and releases the result $D(s)$.
```

modest assumptions on user dynamics, these protocols require impractically high numbers of setup phases for each query.

5.4 A Robust, Privacy-Preserving, Distributed Aggregation Protocol

Protocol 1 describes our protocol Count(t, σ^2). Each user u_i for $i = 1, \ldots, N$ holds a bit b_i. The protocol computes a noisy version of the sum $\sum b_i$.[1] The parameter t is an upper bound on the fraction of malicious or unavailable users, and σ^2 is the amount of noise. If the upper bound t is violated and more users turn out to be malicious or unavailable, the privacy guarantee degrades and/or the protocol needs to be aborted before Step 4 and restarted (with a larger value for t). As t increases, the share of noise each participant adds to her or her bit increases.

[1]In practice, we would use a discretized version of Gaussian noise and do all computations modulo some large number q as done in [93]. In Steps 1, user i would pick her noise r_i from $0, \ldots, q-1$.

5.4.1 Correctness

Let us consider the case where all participants behave as specified. Then Server 2 releases

$$\begin{aligned} D(s) &= D(\sum_i E(b_i + r_i)) \\ &= D(E(\sum_i b_i + r_i)) && \text{By homomorphism Def. 9} \\ &= \sum_i b_i + r_i && \text{By correctness Def. 9,} \end{aligned}$$

which is the sum of the users' private values plus some noise added to protect their privacy.

5.4.2 Efficiency

Count matches the communication complexity of the most efficient previous solutions [89, 93].

Lemma 12. *The number of messages exchanged in* Count *is linear in the number of users.*

Proof. Each user sends two messages. Server 2 sends one message. Overall, the number of messages exchanged is $2N + 1$. □

Note that messages across Count computations can be batched.

5.4.3 Robustness

Unlike previous protocols [89, 93], Count successfully terminates as long as at least $(1-t)N$ users send messages to Server 2. When Count is executed multiple times, it suffices that for each execution, at least $(1-t)N$ possibly different users participate. Thus, our protocol can deal with unavailable users much more efficiently than previous protocols. Unlike these previous protocols, our protocol does not expect the secrets of participating users to add up to a predefined constant.

5.4.4 Privacy

Following Definition 5, we show that the output of the protocol preserves privacy.

Theorem 9. *Consider $\epsilon \leq 1$ and $\sigma^2 \geq 2\ln(4/\delta)/\epsilon^2$. Protocol $\mathsf{Count}(\sigma^2, t)$ guarantees (ϵ, δ)-probabilistic differential privacy of the users in the presence of up to a fraction of t unavailable or malicious users.*

Proof. There are many participants involved in the protocol: Server 1, Server 2, and users U. We now consider privacy with respect to the various participants in the protocol according to Definition 5. In their analysis we will make use of the following building blocks C, A to construct M', R as in the definition.

The users U can be divided into honest users U_h and malicious users U_m. We denote by U'_h the subset of honest users that are available. By our assumption about the honest fraction there are at least $(1-t)N$ honest and available users users, i.e. $|U'_h| \geq (1-t)N$. This assures the following claim: Consider an algorithm C that adds $|U'_h|$ times noise $\mathcal{N}(\sigma^2/((1-t)N-1))$ to $\sum_{l \in U'_h} b_l$.

Claim 8. *Algorithm* C *with* $\sigma \geq \sqrt{2\ln(4/\delta)}/\epsilon$ *preserves* (ϵ, δ)*-privacy.*

We note that since Gaussian distributions are closed under linear transformations (Fact 1) we have that adding $|U'_h|$ times noise $\mathcal{N}(\sigma^2/((1-t)N-1))$ is equivalent to adding noise $\mathcal{N}(|U'_h|\sigma^2/((1-t)N-1))$. By our honest fraction assumption, $|U'_h| \geq (1-t)N-1$ and thus this adds noise with variance greater than σ^2. We note that the L_2-sensitivity is 1. By Theorem 1 it follows that C with $\sigma \geq \sqrt{2\ln(4/\delta)}/\epsilon$ preserves (ϵ, δ)-privacy.

Let A denote the algorithm with which malicious users compute their messages m_{U_m}. There might be multiple adversaries controlling different sets of users. Algorithm A combines these.

- non-participant or honest and unavailable user: Consider an honest and unavailable user or a non-participant who is not colluding with any participant. For non-participants colluding with participants we refer to the analysis of the respective participants. We denote by V the complete view which is simply the output. We use algorithm C as algorithm M' in the privacy Definition 5.

  We use the algorithm $R(v, L_{U_m})$ that outputs $v + \sum_{m_i \in \mathsf{A}(L_{U_m})} D(m_i)$.

  Claim 9. V *and* $R(\mathsf{C}(L_{U'_h}), L_{U_m})$ *are identically distributed.*

We have that

$$\begin{aligned} V &\sim D\left(\sum_{i\in U'_h} E(b_i + \mathcal{N}(\sigma^2/((1-t)N-1))) + \sum_{m_i\in \mathsf{A}(L_{U_m})} m_i\right) \\ &\sim \sum_{i\in U'_h} \big(b_i + \mathcal{N}(\sigma^2/((1-t)N-1))\big) + \sum_{m_i\in \mathsf{A}(L_{U_m})} D(m_i) \qquad \text{By Def. 9} \\ &\sim \mathsf{C}(L_{U'_h}) + \sum_{m_i\in \mathsf{A}(L_{U_m})} D(m_i) \\ &\sim R(\mathsf{C}(L_{U'_h}), L_{U_m}) \end{aligned}$$

Thus privacy is preserved according to Definition 5 with respect to non-participants and unavailable users.

- honest and available user u: User u has as input her own bit b_u. We denote by V the complete view including the user's messages m_u and the output. Consider an algorithm C' that adds $|U'_h - 1|$ times noise $\mathcal{N}(\sigma^2/((1-t)N-1))$ to $\sum_{l\in U'_h, l\neq u} b_l$.

Claim 10. *Algorithm C' with $\sigma \geq \sqrt{2\ln(4/\delta)}/\epsilon$ preserves (ϵ,δ)-privacy.*

We note that since Gaussian distributions are closed under linear transformations (Fact 1) we have that adding $|U'_h - 1|$ times noise $\mathcal{N}(\sigma^2/((1-t)N-1))$ is equivalent to adding noise $\mathcal{N}(|U_{h'} - 1|\sigma^2/((1-t)N-1))$. By our honest fraction assumption, $|U_{h'} - 1| \geq (1-t)N - 1$ and thus this adds noise with variance at least σ^2. We note that the L_2-sensitivity is 1. By Theorem 1 it follows that C' with $\sigma \geq \sqrt{2\ln(4/\delta)}/\epsilon$ preserves (ϵ,δ)-privacy.

We use the algorithm $R(v, L_{U_m\cup\{u\}})$ that samples $X_u \sim \mathcal{N}(\sigma^2/((1-t)N-1))$ and outputs $v + \sum_{m_i\in\mathsf{A}(L_{U_m})} D(m_i) + b_u + X_u$ and $m_u = E(b_u + X_u)$.

Claim 11. *V and $R(\mathsf{C}'(L_{U'_h\setminus\{u\}}), L_{U_m\cup\{u\}})$ are identically distributed.*

Let X_i be a random variable distributed according to $\mathcal{N}(\sigma^2/((1-t)N-1))$. We have that

$$\begin{aligned}
V &\sim D\left(\sum_{i\in U'_h\setminus\{u\}} E(b_i+X_i) + E(b_u+X_u) + \sum_{m_i\in \mathsf{A}(L_{U_m})} m_i\right), E(b_u+X_u) \\
&\sim \sum_{i\in U'_h\setminus\{u\}} (b_i+X_i) + E(b_u+X_u) + \sum_{m_i\in \mathsf{A}(L_{U_m})} D(m_i), E(b_u+X_u) && \text{By Def. 9} \\
&\sim \mathsf{C}'(L_{U'_h\setminus\{u\}}) + b_u + X_u + \sum_{m_i\in \mathsf{A}(L_{U_m})} D(m_i), E(b_u+X_u) \\
&\sim R(\mathsf{C}'(L_{U'_h\setminus\{u\}}), L_{U_m\cup\{u\}})
\end{aligned}$$

Thus, privacy is preserved according to Definition 5 with respect to honest and available users.

- Malicious user $j \in U_m$: Let A denote the adversary controlling user j. This adversary might also control other malicious users $U'_m \subset U_m$. Let V denote the messages received and sent from users in U'_m during the execution of Count on input L.

  Again, we use algorithm C as algorithm M' in the privacy Definition 5.

  We denote by A' the algorithm used by adversary A controlling U'_m. Let A'' denote the algorithm with which malicious users not in U'_m compute their messages $m_{U_m\setminus U'_m}$. We use the algorithm $R(v, L_{U_m}) = v + \sum_{m_i\in\mathsf{A}'(L_{U'_m})} D(m_i) + \sum_{m_i\in\mathsf{A}''(L_{U_m\setminus U'_m})} D(m_i), \mathsf{A}'(L_{U'_m})$.

  Claim 12. *V and $R(\mathsf{C}(L_{U_h}), L_{U_m})$ are identically distributed.*

  Let X_i be a random variable distributed according to $\mathcal{N}(\sigma^2/((1-t)N-1))$.

We have that

$$
\begin{aligned}
V &\sim D\left(\sum_{i\in U'_h} E(b_i + X_i) + \sum_{m_i\in \mathsf{A}'(L_{U'_m})} m_i + \sum_{m_i\in \mathsf{A}''(L_{U_m\setminus U'_m})} m_i\right), \mathsf{A}'(L_{U'_m}) \\
&\sim \sum_{i\in U'_h} b_i + X_i + \sum_{m_i\in \mathsf{A}'(L_{U'_m})} D(m_i) + \sum_{m_i\in \mathsf{A}''(L_{U_m\setminus U'_m})} D(m_i), \mathsf{A}'(L_{U'_m}) \qquad \text{By Def. 9} \\
&\sim C(L_{U'_h}) + \sum_{m_i\in \mathsf{A}'(L_{U_m})} D(m_i) + \sum_{m_i\in \mathsf{A}''(L_{U_m\setminus U'_m})} D(m_i)), \mathsf{A}'(L_{U'_m}) \\
&\sim R(\mathsf{C}(L_{U'_h}), L_{U_m})
\end{aligned}
$$

Thus, privacy is preserved according to Definition 5 with respect to malicious users.

- <u>Server 1:</u> Server 1 sees the sum of all messages m_i and sends out the decrypted sum. We denote by V the view of Server 1 during the execution of Count on input L which is $\sum_j m_j$ of the available users and the decryption thereof. Again, we use algorithm C as algorithm M' in the privacy Definition 5.

  We use the algorithm $R(v, L_{U_m})$ that outputs $E(v) + \sum_{m_i\in \mathsf{A}(L_{U_m})} m_i, v + \sum_{m_i\in \mathsf{A}(L_{U_m})} D(m_i)$.

  Claim 13. *V and $R(\mathsf{C}(L_{U'_h}), L_{U_m})$ are identically distributed.*

  We start by analyzing the distribution of V which is $X, D(X)$ for $X \sim \sum_{i\in U'_h} E(b_i + \mathcal{N}(\sigma^2/((1-t)N-1))) + \sum_{m_i\in \mathsf{A}(L_{U_m})} m_i$. The distribution of $R(\mathsf{C}(L_{U'_h}), L_{U_m})$ is $Y, D(Y)$ where $Y \sim E(\mathsf{C}(L_{U'_h})) + \sum_{m_i\in \mathsf{A}(L_{U_m})} m_i$ by Definition 9 of the homomorphic encryption scheme. Thus, it suffices to show that X, Y are identically distributed. This is the case by definition of C. Hence, privacy is preserved according to Definition 5 with respect to Server 1.

- Server 2: Server 2 receives the messages m_i, sends out the sum of them and sees the decrypted sum. We denote this view by Count_V. Again, we use algorithm C as algorithm M' in the privacy Definition 5.

  Let p_i denote arbitrary plaintexts. We use the algorithm $R(v, L_{U_m})$ that outputs $\mathsf{A}(L_{U_m}), \langle E(p_i) | i \in U'_h \rangle, \sum_{i \in U'_h} E(p_i) + \sum_{m_i \in \mathsf{A}(L_{U_m})} m_i, v + \sum_{m_i \in \mathsf{A}(L_{U_m})} D(m_i)$.

  Claim 14. *V and $R(\mathsf{C}(L_{U'_h}), L_{U_m})$ are identically distributed.*

  We start by analyzing the distribution of V.

  We have that

$$\begin{aligned} V \sim & \mathsf{A}(L_{U_m}), \langle E(X_i + b_i) | i \in U'_h \rangle, \\ & \sum_{m_i \in \mathsf{A}(L_{U_m})} m_i + \sum_{i \in U'_h} E(X_i + b_i), \\ & D\left(\sum_{m_i \in \mathsf{A}(L_{U_m})} m_i + \sum_{i \in U'_h} E(X_i + b_i) \right) \end{aligned}$$

  Now, let us consider the distribution of $R(\mathsf{C}(L_{U'_h}), L_{U_m})$. Let $M'_{U'_h}$ be a set of random variables where M'_i is distributed according to $E(p_i)$.

$$\begin{aligned} R(\mathsf{C}(L_{U'_h}), L_{U_m}) \sim & \ \mathsf{A}(L_{U_m}), \langle E(p_i) | i \in U'_h \rangle, \\ & \sum_{m_i \in \mathsf{A}(L_{U_m})} m_i + \sum_{i \in U'_h} E(p_i), \\ & D\left(\sum_{m_i \in \mathsf{A}(L_{U_m})} m_i \right) + \mathsf{C}(L_{U'_h}) \end{aligned}$$

  Since the encryption scheme is homomorphic and correct, we have that the third components are identically distributed for R and Count:

$$\begin{aligned}
&D\left(\sum_{m_i \in \mathsf{A}(L_{U_m})} m_i + \sum_{i \in U'_h} E(X_i + b_i)\right) \\
\sim &D\left(\sum_{m_i \in \mathsf{A}(L_{U_m})} m_i\right) + \sum_{i \in U'_h} b_i + \mathcal{N}(\sigma^2/((1-t)N-1)) \\
\sim &D\left(\sum_{m_i \in \mathsf{A}(L_{U_m})} m_i\right) + \mathsf{C}(L_{U'_h}).
\end{aligned}$$

Hence, it suffices to show that the first two components given the third component are identically distributed for R and Count. In particular, it suffices to show that $\langle E(p_i) | i \in U'_h \rangle$ given the third component is identically distributed to $\langle E(X_i + b_i) | i \in U'_h \rangle$. This is the case due to our security assumption of (E, D) that states that the distribution over ciphertexts is identical for different plaintexts.

Hence, privacy is preserved according to Definition 5 with respect to Server 2.

□

5.4.5 Additional Privacy Guarantee

Our protocol also provides an additional guarantee in case either Server 2 is corrupted by an adversary (but not colluding with any user or the other server).

Theorem 10. *If Server 2 (but not Server 1) is corrupted by an adversary, we guarantee that the adversary will not be able to learn information that breaches privacy.*

The proof follows from the fact that Server 2 sends only the very last message of the protocol upon which no further action is taken. Thus, even is she is malicious, she can change her message sent but not influence other participants' messages.

This guarantee is meaningful with regard to an adversary seeking to learn private information. We do not guarantee that a malicious adversary cannot breach privacy: an adversary corrupting Server 2 could release the secret key used for the homomorphic encryption scheme, which would allow the honest-but-curious Server 1 to decrypt messages m_i obtaining b_i+r_i. This would breach user privacy.

5.4.6 Extension to Aggregation of Real Values

Count computes a noisy sum over private bits. It can also be used to sum up real numbers. Moreover, it can be used to answer a sequence of sum queries with bounded L_2-sensitivity by setting σ according to Theorem 1.

Theorem 11. *For users with real values $b_i^{(1)}, \ldots, b_i^{(d)}$ Protocol $\mathsf{Count}(\sigma^2, t)$ can be used repeatedly to compute $\sum b_i^{(1)}, \ldots, \sum b_i^{(d)}$ with noise added to protect privacy. Let s denote the L_2-sensitivity of $\sum b_i^{(1)}, \ldots, \sum b_i^{(d)}$. Consider $\epsilon \leq 1$ and $\sigma^2 \geq 2s^2 \ln(4/\delta)/\epsilon^2$. The protocol guarantees (ϵ, δ)-probabilistic differential privacy in the presence of up to a fraction of t unavailable or malicious users.*

Proof. There are many participants involved in the protocol: Server 1, Server 2, and users U. We now consider privacy with respect to the various participants in the protocol according to Definition 5. In their analysis we will make use of the following building blocks C, A to construct M', R as in the definition.

The users U can be divided into honest users U_h and malicious users U_m. We denote by $L^{(k)}$ the input values $b_i^{(k)}$ of all users i. We consider the case where the outputs are computed sequentially, although our privacy analysis extends to the case when the sums are computed concurrently. We denote by $U_h^{'(k)}$ the subset of honest users that are available in round k to compute the noisy sum of $b_i^{(k)}$. By our assumption about the honest fraction there are at least $(1-t)N$ honest and available users users, i.e. $|U_h^{'(k)}| \geq (1-t)N$. This assures the following claim: Let $X_i^{(k)}$ denote a random variable distributed according to $\mathcal{N}(\sigma^2/((1-t)N-1))$. Consider an algorithm C that adds $|U_h'|$ times noise $\mathcal{N}(\sigma^2/((1-t)N-1))$ to sum $\sum_{l \in U_h^{'(k)}} b_l^{(k)}$ for $1 \leq k \leq d$ and outputs the results, i.e. output $O_{\mathcal{C}}^{(k)} = \sum_{i \in U_h^{'(k)}} X_i^{(k)} + b_i^{(k)}$.

Claim 15. *Let s denote the L_2-sensitivity of the sums. Algorithm C with $\sigma \geq s\sqrt{2\ln(4/\delta)}/\epsilon$ preserves (ϵ, δ)-privacy.*

We note that since Gaussian distributions are closed under linear transformations (Fact 1) we have that adding $|U_h^{'(k)}|$ times noise $\mathcal{N}(\sigma^2/((1-t)N-1))$ is equivalent to adding noise $\mathcal{N}(|U_h^{'(k)}|\sigma^2/((1-t)N-1))$. By our honest fraction assumption, $|U_h^{'(k)}| \geq (1-t)N-1$ and thus this adds noise with variance greater than σ^2. By Theorem 1 and the fact that the sums L_2-sensitivity is bounded by s, it follows that C with $\sigma \geq s\sqrt{2\ln(4/\delta)}/\epsilon$ preserves (ϵ, δ)-privacy.

Let A denote the algorithm with which malicious users compute their messages m_{U_m}. Note that there might be multiple adversaries controlling different sets of users. Algorithm A combines these. The adversaries can compute messages based on the private inputs of the users they are controlling and the previously released outputs.

We now consider privacy with respect to the various participants.

- non-participant: Consider a non-participant who is not colluding with any participant. For non-participants colluding with participants we refer to the analysis of the respective participants. We denote by V the complete view which are the d outputs. We use algorithm C as algorithm M' in the privacy Definition 5.

  We use the algorithm $R(v^{(1)}L^{(1)}_{U_m}, \ldots, v^{(d)}L^{(d)}_{U_m})$ that in order to produce the kth output $O^{(k)}_R$ computes $v^{(k)} + \sum_{m_i \in \mathsf{A}(L^{(1)}_{U_m},\ldots,L^{(d)}_{U_m},O^{(1)}_R,\ldots,O^{(k-1)}_R)} D(m_i)$.

  Claim 16. $\mathsf{Count}(L^{(1)}, \ldots, L^{(d)})_V$ *and* $R(\mathsf{C}(L^{(1)}_{U_h^{'(1)}}, \ldots, L^{(d)}_{U_h^{'(d)}}), L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m})$ *are identically distributed.*

  We have that for $\mathsf{Count}(L^{(1)}, \ldots, L^{(d)})_V$ the kth output $O^{(k)}_{\mathsf{Count}}$ is distributed as follows

$$\begin{aligned} O^{(k)}_{\mathsf{Count}} &\sim D\left(\sum_{i \in U_h^{'(d)}} E(b_i^{(k)} + X_i^{(k)}) + \sum_{\substack{m_i \in \mathsf{A}(L^{(1)}_{U_m},\ldots,L^{(d)}_{U_m}, \\ O^{(1)},\ldots,O^{(k-1)})}} m_i \right) \\ &\sim \sum_{i \in U_h^{'(k)}} \left(b_i^{(k)} + X_i^{(k)} \right) + \sum_{\substack{m_i \in \mathsf{A}(L^{(1)}_{U_m},\ldots,L^{(d)}_{U_m}, \\ O^{(1)},\ldots,O^{(k-1)})}} D(m_i)) \qquad \text{By Def. 9} \end{aligned}$$

  Similarly, for $R(\mathsf{C}(L^{(1)}_{U_h^{'(1)}}, \ldots, L^{(d)}_{U_h^{'(d)}}), L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m})$ the kth output $O^{(k)}_R$ is distributed as follows

$$O^{(k)}_R \sim O^{(k)}_{\mathcal{C}} + \sum_{m_i \in \mathsf{A}(L^{(1)}_{U_m},\ldots,L^{(d)}_{U_m},O^{(1)}_R,\ldots,O^{(k-1)}_R)} D(m_i)$$

By definition of $\mathcal{C}$ we have that

$$O_{\mathcal{C}}^{(k)} \sim \sum_{i \in U_h^{'(k)}} \left(b_i^{(k)} + \mathcal{N}(\sigma^2/((1-t)N-1)) \right).$$

From induction on k it follows that $O_{\mathsf{Count}}^{(k)} \sim O_R^{(k)}$.

Thus privacy is preserved according to Definition 5 with respect to non-participants.

- honest user u: User u has as input her own values $b_u^{(1)}, \dots, b_u^{(d)}$. We denote by V the complete view including the user's messages $m_u^{(k)}$ and the outputs $O_{\mathsf{Count}}^{(k)}$. Let the bit $a_u^{(k)}$ denote whether the user is available in round k.

  Consider an algorithm C that adds $|U_h^{'(k)} - a_u^{(k)}|$ times noise $\mathcal{N}(\sigma^2/((1-t)N-1))$ to $\sum_{l \in U_h^{'(k)}, l \neq u} b_l^{(k)}$ for $1 \leq k \leq d$.

  Claim 17. *Algorithm* C *with* $\sigma \geq s\sqrt{2\ln(4/\delta)}/\epsilon$ *preserves* (ϵ, δ)*-privacy.*

  We note that since Gaussian distributions are closed under linear transformations (Fact 1) we have that adding $|U_h^{'(k)} - a_u^{(k)}|$ times noise $\mathcal{N}(\sigma^2/((1-t)N-1))$ is equivalent to adding noise $\mathcal{N}(|U_h^{'(k)} - a_u^{(k)}|\sigma^2/((1-t)N-1))$. By our honest fraction assumption, $|U_h^{'(k)} - a_u^{(k)}| \geq (1-t)N-1$ and thus this adds noise with variance at least σ^2. We note that the L_2-sensitivity is bounded by s. By Theorem 1 it follows that C with $\sigma \geq s\sqrt{2\ln(4/\delta)}/\epsilon$ preserves (ϵ, δ)-privacy.

  We use algorithm C as algorithm M' in the privacy Definition 5.

  We use the algorithm $R(v^{(1)}L_{U_m}^{(1)}, \dots, v^{(d)}L_{U_m}^{(d)})$ that, in order to produce the kth output $O_R^{(k)}$, samples $X_u^{(k)} \sim \mathcal{N}(\sigma^2/((1-t)N-1))$ and computes

$$v^{(k)} + a_u^{(k)}(X_u^{(k)} + b_u^{(k)}) + \sum_{m_i \in \mathsf{A}(L_{U_m}^{(1)}, \dots, L_{U_m}^{(d)}, O_R^{(1)}, \dots, O_R^{(k-1)})} D(m_i)$$

and the message

$$E_u^{(k)} = \begin{cases} E(b_u^{(k)} + X_u^{(k)}) & \text{if } a_u^{(k)} = 1 \\ \perp & \text{otherwise} \end{cases}$$

For notational convenience we define $x + \perp = x$ and $E(\perp) = D(\perp) = \perp$.

Claim 18. *$\mathsf{Count}(L^{(1)}, \ldots, L^{(d)})_V$ and $R(\mathsf{C}(L^{(1)}_{U_h'^{(1)}}, \ldots, L^{(d)}_{U_h'^{(d)}}), L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m})$ are identically distributed.*

We have that for $\mathsf{Count}(L^{(1)}, \ldots, L^{(d)})_V$ the kth output $O^{(k)}_{\mathsf{Count}}$ and the kth message by u, $m_u^{(k)}$, are distributed as follows

$$\begin{aligned} &O^{(k)}_{\mathsf{Count}}, m_u^{(k)} \\ &\sim D\left(\sum_{i \in U_h'^{(k)} \setminus \{u\}} E(b_i^{(k)} + X_i^{(k)}) + E_u^{(k)} + \sum_{\substack{m_i \in \mathsf{A}(L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m}, \\ O^{(1)}, \ldots, O^{(k-1)})}} m_i \right), E_u^{(k)} \\ &\sim \sum_{i \in U_h'^{(k)} \setminus \{u\}} b_i^{(k)} + X_i^{(k)} + a_u^{(k)}(b_u^{(k)} + X_u^{(k)}) + \sum_{\substack{m_i \in \mathsf{A}(L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m}, \\ O^{(1)}, \ldots, O^{(k-1)})}} D(m_i), E_u^{(k)} \quad \text{By Def. 9} \end{aligned}$$

Similarly, for $R(\mathsf{C}(L^{(1)}_{U_h'^{(1)}}, \ldots, L^{(d)}_{U_h'^{(d)}}), L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m})$ the kth output $O_R^{(k)}$ and the kth message by u, $m_u^{(k)}$, are distributed as follows

$$O_R^{(k)}, m_u^{(k)} \sim O_{\mathcal{C}}^{(k)} + a_u^{(k)}(X_u^{(k)} + b_u^{(k)}) + \sum_{m_i \in \mathsf{A}(L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m}, O_R^{(1)}, \ldots, O_R^{(k-1)})} D(m_i), E_u^{(k)}$$

By definition of $\mathcal{C}$ we have that

$$O_{\mathcal{C}}^{(k)} \sim \sum_{i \in U_h'^{(k)} \setminus \{u\}} \left(b_i^{(k)} + \mathcal{N}(\sigma^2/((1-t)N - 1)) \right).$$

From induction on k it follows that $O^{(k)}_{\mathsf{Count}} \sim O_R^{(k)}$.

Thus, privacy is preserved according to Definition 5 with respect to honest users.

- Malicious user $j \in U_m$: Let A denote the adversary controlling user j. This adversary might also control other malicious users $U'_m \subset U_m$ of up to tN users. We denote by V the view of these users with the outputs of the protocol and the messages sent by them.

  Again, we use algorithm C as algorithm M' in the privacy Definition 5.

  We denote by A' the algorithm used by the adversary controlling U'_m. Let A'' denote the algorithm with which malicious users not in U'_m compute their messages $m_{U_m \setminus U'_m}$. We use the algorithm $R(v^{(1)} L^{(1)}_{U_m}, \ldots, v^{(d)} L^{(d)}_{U_m})$ that in order to produce the kth output $O^{(k)}_R$ and the messages $m^{(k)}_{U'_m}$ computes $v^{(k)} + \sum_{m_i \in \mathsf{A}''(L^{(1)}_{U_m \setminus U'_m}, \ldots, L^{(d)}_{U_m \setminus U'_m}, O^{(1)}_R, \ldots, O^{(k-1)}_R)} D(m_i) + \sum_{m_i \in \mathsf{A}'(L^{(1)}_{U'_m}, \ldots, L^{(d)}_{U'_m}, O^{(1)}_R, \ldots, O^{(k-1)}_R)} D(m_i)$, $\mathsf{A}'(L^{(1)}_{U'_m}, \ldots, L^{(d)}_{U'_m}, O^{(1)}_R, \ldots, O^{(k-1)}_R)$.

  Claim 19. *$\mathsf{Count}(L^{(1)}, \ldots, L^{(d)})_V$ and $R(\mathsf{C}(L^{(1)}_{U'^{(1)}_h}, \ldots, L^{(d)}_{U'^{(d)}_h}), L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m})$ are identically distributed.*

  We have that for $\mathsf{Count}(L^{(1)}, \ldots, L^{(d)})_V$ the kth output $O^{(k)}_{\mathsf{Count}}$ and the messages $m^{(k)}_{U'_m}$ are distributed as follows

$$\begin{aligned}
&O^{(k)}_{\mathsf{Count}}, m_{U'_m} \\
\sim &D\left(\sum_{i \in U'^{(k)}_h} E(b^{(k)}_i + X^{(k)}_i) + \sum_{\substack{m_i \in \mathsf{A}''(L^{(1)}_{U_m \setminus U'_m}, \ldots, \\ L^{(d)}_{U_m \setminus U'_m}, O^{(1)}, \ldots, O^{(k-1)})}} m_i + \sum_{\substack{m_i \in \mathsf{A}'(L^{(1)}_{U'_m}, \ldots, \\ L^{(d)}_{U'_m}, O^{(1)}, \ldots, O^{(k-1)})}} m_i \right), \\
&\mathsf{A}'(L^{(1)}_{U'_m}, \ldots, L^{(d)}_{U'_m}, O^{(1)}, \ldots, O^{(k-1)}) \\
\sim &\sum_{i \in U'^{(k)}_h} b^{(k)}_i + X^{(k)}_i + \sum_{\substack{m_i \in \mathsf{A}''(L^{(1)}_{U_m \setminus U'_m}, \ldots, \\ L^{(d)}_{U_m \setminus U'_m}, O^{(1)}, \ldots, O^{(k-1)})}} D(m_i) + \sum_{\substack{m_i \in \mathsf{A}'(L^{(1)}_{U'_m}, \ldots, \\ L^{(d)}_{U'_m}, O^{(1)}, \ldots, O^{(k-1)})}} D(m_i))), \\
&\mathsf{A}'(L^{(1)}_{U'_m}, \ldots, L^{(d)}_{U'_m}, O^{(1)}, \ldots, O^{(k-1)}) \qquad \text{By Def. 9}
\end{aligned}$$

Similarly, for $R(\mathsf{C}(L^{(1)}_{U'^{(1)}_h},\ldots,L^{(d)}_{U'^{(d)}_h}),L^{(1)}_{U_m},\ldots,L^{(d)}_{U_m})$ the k^{th} output $O^{(k)}_R$ and the messages $m^{(k)}_{U'_m}$ are distributed as follows

$$\begin{aligned}&O^{(k)}_R, m_{U'_m}\\ &\sim O^{(k)}_{\mathcal{C}} + \sum_{\substack{m_i\in\mathsf{A}''(L^{(1)}_{U_m\setminus U'_m},\ldots,\\ L^{(d)}_{U_m\setminus U'_m},O^{(1)}_R,\ldots,O^{(k-1)}_R)}} D(m_i) + \sum_{\substack{m_i\in\mathsf{A}'(L^{(1)}_{U'_m},\ldots,\\ L^{(d)}_{U'_m},O^{(1)}_R,\ldots,O^{(k-1)}_R)}} D(m_i), \mathsf{A}'(L^{(1)}_{U'_m},\ldots,L^{(d)}_{U'_m},O^{(1)}_R,\ldots,O^{(k-1)}_R)\end{aligned}$$

By definition of $\mathcal{C}$ we have that

$$O^{(k)}_{\mathcal{C}} \sim \sum_{i\in U'^{(k)}_h}\left(b^{(k)}_i + \mathcal{N}(\sigma^2/((1-t)N-1))\right).$$

From induction on k it follows that $O^{(k)}_{\mathsf{Count}} \sim O^{(k)}_R$.

Thus, privacy is preserved according to Definition 5 with respect to malicious users.

- Server 1: Server 1 sees the sum of all messages $m^{(k)}_i$ and sends out the decrypted sum. We denote by V the complete view which is $\sum_j m^{(k)}_j$ of the available participants and the decryption thereof. Again, we use algorithm C as algorithm M' in the privacy Definition 5.

  We use the algorithm $R(v^{(1)}L^{(1)}_{U_m},\ldots,v^{(d)}L^{(d)}_{U_m})$ that in order to produce the k^{th} view computes

$$\begin{aligned}S^{(k)}_R &= E(v^{(k)}) + \sum_{m_i\in\mathsf{A}(L^{(1)}_{U_m},\ldots,L^{(d)}_{U_m},O^{(1)}_R,\ldots,O^{(k-1)}_R)} m_i,\\ O^{(k)}_R &= v^{(k)} + \sum_{m_i\in\mathsf{A}(L^{(1)}_{U_m},\ldots,L^{(d)}_{U_m},O^{(1)}_R,\ldots,O^{(k-1)}_R)} D(m_i)\end{aligned}$$

Claim 20. *$\mathsf{Count}(L^{(1)},\ldots,L^{(d)})_V$ and $R(\mathsf{C}(L^{(1)}_{U'^{(1)}_h},\ldots,L^{(d)}_{U'^{(d)}_h}),L^{(1)}_{U_m},\ldots,L^{(d)}_{U_m})$ are identically distributed.*

We start by analyzing the distribution of $\mathsf{Count}(L)_V$ in round k which is $Y^{(k)}, D(Y^{(k)})$ by Definition 9 for

$$Y^{(k)} \sim \sum_{i \in U_h'^{(k)}} E(b_i^{(k)} + \mathcal{N}(\sigma^2/((1-t)N-1))) + \sum_{m_i \in \mathsf{A}(L_{U_m}^{(1)}, \ldots, L_{U_m}^{(d)}, D(Y^{(1)}), \ldots, D(Y^{(k-1)})} m_i.$$

The distribution of $R(v^{(1)} L_{U_m}^{(1)}, \ldots, v^{(d)} L_{U_m}^{(d)})$ is $Z^{(k)}, D(Z^{(k)})$ where

$$Z^{(k)} \sim E(O_{\mathcal{C}}^{(k)}) + \sum_{m_i \in \mathsf{A}(L_{U_m}^{(1)}, \ldots, L_{U_m}^{(d)}, D(Z^{(1)}), \ldots, D(Z^{(k-1)}))} m_i$$

by Definition 9. Thus, it suffices to show that $Y^{(k)}, Z^{(k)}$ are identically distributed. By definition of $\mathcal{C}$ we have that $O_{\mathcal{C}}^{(k)} \sim \sum_{i \in U_h'^{(k)}} \left(b_i^{(k)} + \mathcal{N}(\sigma^2/((1-t)N-1)) \right)$. From induction on k the claim follows. Hence, privacy is preserved according to Definition 5 with respect to Server 1.

- Server 2: Server 2 receives the messages, sends out the sum of them and sees the decrypted sum. We denote this view by V. Again, we use algorithm C as algorithm M'.

  Let $p_i^{(k)}$ denote arbitrary plaintexts. We use algorithm $R(v^{(1)} L_{U_m}^{(1)}, \ldots, v^{(d)} L_{U_m}^{(d)})$ to produce the k^{th} messages $M_R^{(k)}$, sum over these messages $S_R^{(k)}$ as well as the output $O_R^{(k)}$ as

$$\begin{aligned} M_R^{(k)} &\sim \mathsf{A}(L_{U_m}^{(1)}, \ldots, L_{U_m}^{(d)}, O_R^{(1)}, \ldots, O_R^{(k-1)}), \langle E(p_i^{(k)}) | i \in U_h'^{(k)} \rangle \\ S_R^{(k)} &\sim \sum_{m_i \in \mathsf{A}(L_{U_m}^{(1)}, \ldots, L_{U_m}^{(d)}, O_R^{(1)}, \ldots, O_R^{(k-1)})} m_i + E(v^{(k)}) \\ O_R^{(k)} &\sim D\left(S_R^{(k)}\right) \end{aligned}$$

Claim 21. *$\mathsf{Count}(L^{(1)}, \ldots, L^{(d)})_V$ and $R(\mathsf{C}(L_{U_h'^{(1)}}^{(1)}, \ldots, L_{U_h'^{(d)}}^{(d)}), L_{U_m}^{(1)}, \ldots, L_{U_m}^{(d)})$ are identically distributed.*

We start by analyzing the distribution of $\mathsf{Count}(L^{(1)}, \ldots, L^{(d)})_V$ in the k^{th} round. We have that the view of the k^{th} round is

$$\begin{aligned}
M^{(k)}_{\mathsf{Count}} &\sim \mathsf{A}(L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m}, O^{(1)}_{\mathsf{Count}}, \ldots, O^{(k-1)}_{\mathsf{Count}}), \langle E(b^{(k)}_i + X^{(k)}_i) | i \in U'^{(k)}_h \rangle, \\
S^{(k)}_{\mathsf{Count}} &\sim \sum_{m_i \in \mathsf{A}(L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m}, O^{(1)}_{\mathsf{Count}}, \ldots, O^{(k-1)}_{\mathsf{Count}})} m_i + \sum_{i \in U'^{(k)}_h} E(b^{(k)}_i + X^{(k)}_i), \\
O^{(k)}_{\mathsf{Count}} &\sim D\left(S^{(k)}_{\mathsf{Count}}\right)
\end{aligned}$$

Since the encryption scheme is homomorphic and correct, we can prove by induction on k that the third components are identically distributed for R and Count:

$$\begin{aligned}
O^{(k)}_{\mathsf{Count}} &\sim \sum_{m_i \in \mathsf{A}(L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m}, O^{(1)}_{\mathsf{Count}}, \ldots, O^{(k-1)}_{\mathsf{Count}})} D\left(m_i\right) + \sum_{i \in U'^{(k)}_h} b^{(k)}_i + X^{(k)}_i \\
&\sim \sum_{m_i \in \mathsf{A}(L^{(1)}_{U_m}, \ldots, L^{(d)}_{U_m}, O^{(1)}_{R}, \ldots, O^{(k-1)}_{R})} D(m_i) + O^{(k)}_{\mathcal{C}} \\
&\sim O^{(k)}_{R}.
\end{aligned}$$

Hence, it suffices to show that the first two components given the third component are identically distributed for R and Count. For the second component this is the case by Definition 9. For the first component, it follows from our security assumption of (E, D) that states that the distribution over ciphertexts is identical for different plaintexts.

The claim then follows by induction on k. Hence, privacy is preserved according to Definition 5 with respect to Server 2.

$\square$

5.5 Employing Count to compute CTRs

The optimization framework for delivering targeted advertisements in Chapter 4 uses various statistics; in this chapter we describe how to obtain those. Specifically, we develop protocols for computing the probability distribution over contexts, $\Pr[c]$, and the context-dependent click-through rates, $\mathsf{CTR}(a|c)$. $\Pr[c]$ can be estimated as the number of times a user reported to be in context c divided by the total number of reported contexts. For an ad a and a context c, $\mathsf{CTR}(a|c)$ can be estimated as the number of times a user in context c reported to have clicked on a divided by the number of times a user in context c reported to have viewed a. These estimations are based on count queries; hence we focus on privacy-preserving computation of count queries in the rest of the chapter.

Note that the above statistics can be estimated well for contexts with a lot of user data (e.g., clicks). However, we consider context attributes beyond location (unlike location-based services such as [5, 17, 18, 33, 35, 56, 81, 39, 104, 106]) and one big challenge is that sufficient click data may not be available for a large number of contexts. For such rare or new contexts, the estimates can be noisy or not even be defined. One option would be to simply not show any ads to a user in a rare context. However, this would seriously harm utility since there is typically a long tail of rare contexts (as is the case in Web search, movie, music and Web browsing [38]). Instead, we suggest estimating $\Pr[c]$ and $\mathsf{CTR}(a|c)$ for a rare context c based on contexts similar to c for which we have enough click data. Coming back to our example from Section 4.2, if we do not have enough click data for users who were skating in Central Park (c), we might use clicks from users in close-by locations who were doing some sort of physical activity ($\tilde{c}$) to estimate the statistics for the context c. This helps us increase coverage

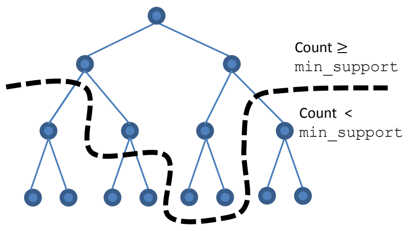


Figure 5.1: Hierarchy H over contexts.

of targeted ads, albeit at the possible cost of lower quality ads. We can trade-off coverage and relevance by adding a constraint on CTR for displayed ads as discussed in Sections 4.2.3 and 4.3.2.

5.5.1 Possible Solutions

One possible way to employ Count to obtain privacy-preserving statistics for various contexts is to compute noisy counts for all possible contexts and all possible ads. Another alternative approach, with better utility, would be to use multi-dimensional histograms [103, 44]. However, all these approaches have a running time at least linear in the number of possible contexts, rendering them infeasible. Moreover, a large part of the computation is wasteful, since, as mentioned before, statistics computed for rare contexts are almost meaningless.

To address this problem, we opt for a simple top-down approach that can efficiently deal with sparse data by identifying and pruning the computations for rare contexts. The solution requires using a context hierarchy that specifies similarity of contexts. Such a top-down algorithm has been used recently to find frequent signatures by gradually expanding the prefix of signatures with high noisy counts [79]. We adapt it to compute CTRs over a context hierarchy.

5.5.2 Context Hierarchy

To define similarity over contexts, we reuse the hierarchies over which users generalize their context attributes. The context hierarchy is built by merging various attribute hierarchies; a concrete example can be found in Section 4.4.1. This hierarchy tells us, for each generalized context, which attribute to generalize next.[2] Given some rare context, we can generalize it until we have a sufficient number of clicks for the generalized context. With these clicks we estimate the CTRs. The parameter `min_support` specifies how many clicks are sufficient for robust estimates. Figure 5.1 shows a hierarchy H over contexts with leaf nodes being exact contexts and intermediate nodes being generalized contexts. It shows a cut-off through the hierarchy so that all (generalized) contexts above the cut-off have at least `min_support` many clicks in the training data for descendant contexts. The contexts below the threshold need to be generalized before estimating on their CTRs.[3]

5.5.3 A Top-Down Algorithm

To compute privacy-preserving CTRs for the generalized contexts in the hierarchy H, algorithm TopDown starts at the root and moves down the hierarchy. For each traversed node v and for each ad a, it estimates $\mathsf{CTR}(a|v)$ by calling Count to compute how often users in a descendant context of v have clicked (or only viewed) a. The results of this computation is referred to as

[2]It is recommend but not required that users generalize the contexts they send to the server to a node in the hierarchy.

[3]Note, that there are other ways to define similarity, for example using the lattice structure imposed by the attributes' hierarchies. Our experimental results show only a minor effect on quality when using a fixed combined hierarchy as opposed to a lattice structure.

$\texttt{clicks}_{a,v}$ ($\texttt{no\_clicks}_{a,v}$, resp.). The estimated click-through-rate is then simply $\widehat{\mathsf{CTR}}(a|v) = \frac{\texttt{clicks}_a}{\texttt{clicks}_{a,v}+\texttt{no\_clicks}_{a,v}}$. TopDown also computes the total number of times a descendant of v appears in the ad log and adds noise to this count. If the count is above min_support then the algorithm recurses on v's children, otherwise all descendants are pruned. We note that the accuracy of Estimates can be further improved by using the post-processing techniques of Hay et al. to make sure the counts of all children add up to the parent's count [45]. To bound the sensitivity and to guarantee differential privacy, we limit the number of entries per user in the ad log. Estimates deletes from the ad log all but m random entries per user and then calls TopDown.

We now analyze the privacy and efficiency of our algorithm.

Proposition 8 (Efficiency). *Let b denote the maximum number of ads bidding on the same context. We denote by height(H) the height of the hierarchy and by branch(H) the maximum number of children for a node in H. Consider a version of* Estimates *using an exact counting procedure. The number of counts is bounded by $(b + \mathit{branch}(H)) \cdot \mathit{height}(H) \cdot N \cdot m/\texttt{min\_support}$.*

For the original version of Estimates *the number of counts can be higher when a node's noisy count is at least* min_support *while its true count is less than at least* min_support. *However, the probability of this is bounded by*

$$\frac{e^{-(\texttt{min\_support}-c)^2((1-t)N-1)^2/(2\sigma^2)}\sigma}{((1-t)N-1)(\texttt{min\_support}-c)\sqrt{2\pi}}.$$

Proof. Consider a version of Estimates using an exact counting procedure. In that case there are at most $N \cdot m/\texttt{min\_support}$ many nodes at each level with a count of at least min_support. Each of these nodes can issue counts for all ads bidding on this node (at most b) and all children (at most branch(H)). Overall the

```
Estimates(context-driven ad log, noise scale $\lambda$, threshold min_support, contribution bound $m$, hierarchy $H$)
for each user do
    Delete all but $m$ views or clicks on ads and their contexts of this user from the ad log.
end for
return TopDown(ad log, root($H$), $\lambda$, min_support)
```

Algorithm 7: Privacy-preserving Estimates.

number of counts is bounded by $(b+\text{branch}(H))\cdot\text{height}(H)\cdot N\cdot m/\texttt{min\_support}$.

Now, with the noisy version a count of a node can falsely appear to be at least `min_support` when it really is c which is less than `min_support`. However, by Fact 1 this probability is equal to the probability that a standard Gaussian variable with zero mean and 1 variance exceeds the value $(\texttt{min\_support}-c)((1-t)N-1)/\sigma$. This probability can be bounded using Fact 2 by

$$\frac{e^{-(\texttt{min\_support}-c)^2((1-t)N-1)^2/(2\sigma^2)}\sigma}{((1-t)N-1)(\texttt{min\_support}-c)\sqrt{2\pi}}.$$

□

The proposition states that Estimates can be much more efficient than the other possible solutions discussed in Section 5.5.1 whose running time is linear in the number of contexts. The number of contexts can be up to $\text{branch}(H)^{\text{height}(H)}$. If this number is much larger than the number of users then Estimates can be more efficient.

To improve the efficiency when using Count in Estimates, we can batch all the messages in one level in the hierarchy.

In Estimates we employ Count to obtain noisy estimates of $\texttt{clicks}_{a,v}$, $\texttt{no\_clicks}_{a,v}$, and $\texttt{count}_v$.

```
TopDown(context-driven ad log, node $v$ in the hierarchy, noise scale $\lambda$, threshold $\texttt{min\_support}$)
$\texttt{count}_v = \mathsf{Count}$ (# of appearances of node $v$ appears)
release $\texttt{count}_v$
if $\texttt{count}_v > \texttt{min\_support}$ then
    $\mathcal{A}' =$ set of ads with bids on context of $v$
    for $a \in \mathcal{A}'$ do
        $\texttt{clicks}_{a,v} = \mathsf{Count}$ (# of clicks on $a$ in $v$ in ad log)
        $\texttt{no\_clicks}_{a,v} = \mathsf{Count}$ (# of views of $a$ w/o clicks in $v$)
        release $\widehat{\mathsf{CTR}}(a|v) = \frac{\texttt{clicks}_{a,v}}{\texttt{clicks}_{a,v}+\texttt{no\_clicks}_{a,v}}$
    end for
    for each child $w$ of $v$ do return TopDown(ad log, $w$ , $\lambda$, $\texttt{min\_support}$)
    end for
end if
```

Algorithm 8: Top-Down computation of noisy statistics.

Corollary 2 (Privacy). *Consider any $\epsilon \leq 1$. Let σ^2 be at least $6 height(H) m^2 \log(4/\delta)/\epsilon^2$. When* Estimates *employs* $\mathsf{Count}(t, \sigma^2)$ *as a subroutine for counting* $\texttt{clicks}_{a,v}$, $\texttt{no\_clicks}_{a,v}$, $\texttt{count}_v$, *it guarantees (ϵ, δ)-probabilistic differential privacy. A fraction of t unavailable or malicious users during each call of* $\mathsf{Count}(t, \sigma^2)$ *can be tolerated.*

Proof. Consider two neighboring click logs L, L' where L' is obtained from L by adding or deleting the data of a single user. Consider the hierarchy H consisting of height(H) levels where level i contains 2^i many nodes. We denote by $c_{l,j}$ ($c'_{l,j}$, respectively) the count of the j^{th} node at level l in L (L', respectively). We denote by $c_{l,j,a,1}$ ($c'_{l,j,a,1}$) the count of the clicks on a in the j^{th} node at level l and by $c_{l,j,a,0}$ ($c'_{l,j,a,0}$) the count of the views of a in the j^{th} node at level l that did not result in clicks in L (L', respectively).

Within a level of the hierarchy the L_2-sensitivity of each count is at most m.

Overall the square of the L_2-sensitivity is at most

$$\sum_{l=0}^{\text{height}(H)-1} \sum_{j=1}^{2^l} (c_{l,j} - c'_{l,j})^2 + \sum_a (c_{l,j,a,1} - c'_{l,j,a,1})^2 + (c_{l,j,a,0} - c'_{l,j,a,0})^2 \tag{5.1}$$

$$\leq \sum_{l=0}^{\text{height}(H)-1} 3 \cdot m^2 \tag{5.2}$$

$$= 3\text{height}(H)m^2 \tag{5.3}$$

Thus, the L_2-sensitivity is bounded by $\sqrt{(3\text{height}(H)}m$. From Theorem 11 it follows that choosing $\sigma^2 \geq 3\text{height}(H)m^2 2\log(4/\delta)/\epsilon^2$ guarantees (ϵ, δ) probabilistic differential privacy. □

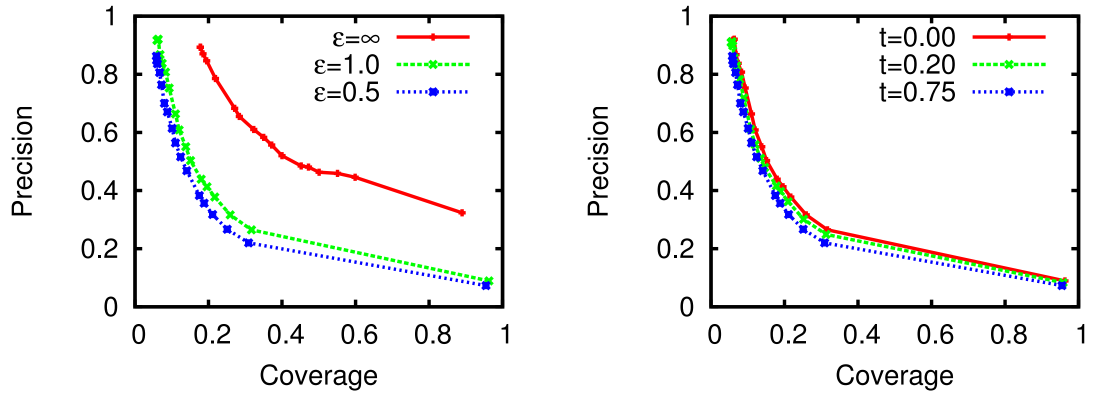
5.6 Experiments

5.6.1 Experimental Setup

We use the same dataset of the location-aware searches of 116,432 users over the course of a month in Microsoft Bing for mobile as in Section 4.4.1. Recall that the user context is comprised of location, user interest, and the user query.

Metrics. To evaluate the accuracy of our Count protocol we apply it to extract CTRs that we use in turn to target advertisements. We compare the quality of the targeted advertisements in terms of precision and coverage with that of the targeted advertisements computed based on the non-privacy-preserving CTRs. We report average precision and coverage for 1,000 random contexts from the testing data; the averages become fairly stable after 1,000 predictions.

Parameters. Unless otherwise stated, we use the following default configura-



(a) Effect of noise ϵ (b) Effect of robustness t

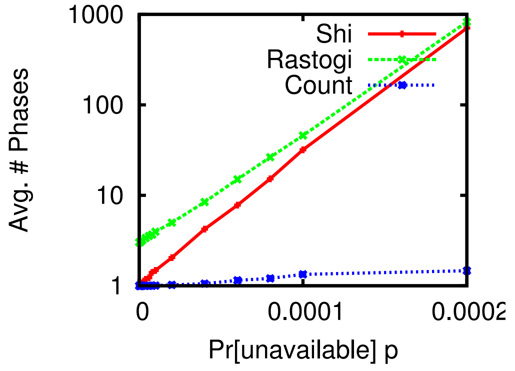
Figure 5.2: Differentially-private estimates.

tion. For limited information disclosure, we use $(4, 2, 2)$ generalization. We set the upper bound on communication complexity, k, to be 10, the threshold on click-through rate to be 0.3, and the threshold on support to be 2. We fixed $\delta = 0.02$ and the maximum number of contributions per user, $m = 4$. Moreover, we found it beneficial to limit how far TopDown goes down in the hierarchy. Such a limit reduces the amount of noise added to each count. This is important for training data as small as this. Therefore, we chose an aggressive limit of 1.

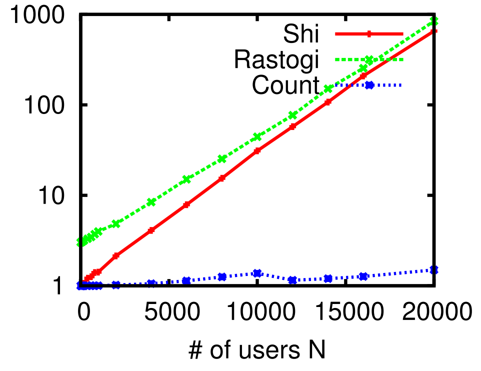
5.6.2 Privacy-Preserving CTRs

Efficiency. When we run Estimates on the log, a user has to send roughly 1MB on average. Many of the count queries can be batched. On average, a user participates in two batches for all CTR computations. We feel this communication cost is acceptable.

Accuracy. Figure 5.2(a) shows how precision and coverage of our framework degrades when we increase the differential privacy guarantee (by decreasing



(a) $N = 10{,}000$



(b) $p = 0.0001$

Figure 5.3: Varying user population.

ϵ). As a point of comparison, the figure also draws the precision and coverage curve when using the exact, non-private statistics ($\epsilon = \infty$). We can see that to achieve a precision of 0.6, the coverage of our framework using non-private statistics is much higher than the coverage of our framework using the ϵ-differentially private statistics (i.e., 0.3 vs 0.1). This is the price we have to pay for a privacy guarantee. The exact value of the privacy parameter ϵ (1 vs. 0.5) has a minor effect on the utility. We expect the cost of privacy to decrease with a larger user population. Moreover, we can avoid such negative impact on utility by paying the price of privacy in terms of communication overhead k—as shown in Figure 4.6, the utility can be improved by using a higher value of k.

Robustness. Figure 5.2(b) shows the effect of varying t (fraction of malicious/unavailable users) on precision and coverage for $\epsilon = 1.0$. We see that the parameter t has only a mild effect. Even when 75% of the users could be unavailable or malicious ($t = 0.75$) the utility is almost the same as when all users are available and honest.

To compare the robustness of our Count protocol with existing works, we model users' unavailability as a simple random process. Suppose a *phase* denotes the time it takes for the server to send a message to all users or for all users to send messages to the server. Let p denote the probability that a user is unavailable to participate in a given phase. We compare various protocols in terms of the average number of phases required to complete a query, as it indicates the latency and communication complexity of a protocol.

Figure 5.3 illustrates the effects of unavailability on communication complexity. We compare our algorithm with two existing protocols: RASTOGI [89] and SHI [93]. We run 1000 queries and report the average number of phases per query for different protocols. As shown, all the protocols cost close to their optimal number of phases when the probability of being unavailable (p) and the number of users (N) are small. However, *unlike our protocol, the costs for* SHI *and* RASTOGI *increase exponentially with N and p* (note the log scale of the graphs). For $p \geq 0.0001$ (corresponding to less than only 10 seconds a day) in (a) or $N \geq 1000$ (much fewer than users of popular online services) in (b) the protocols become impractical. This shows that unlike our protocol, SHI and RASTOGI are impractical for online services with dynamic users.

CHAPTER 6

PRIVATELY RELEASING COUNT STATISTICS FROM SEARCH LOGS

6.1 Introduction

Civilization is the progress toward a society of privacy. The savage's whole existence is public, ruled by the laws of her 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.

The only release of a search log happened in 2007 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 [7], 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 [53] and the replacement of keywords in search queries by random numbers [68].

In this chapter, 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* [90] in search logs [1, 82, 47, 50] are insufficient in the light of attackers who can actively influence the search log. We then turn to *differential privacy* [25], 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. [65] and us with the goal to achieve relaxations of differential privacy. Korolova et al. showed how to set the parameters of ZEALOUS to guaran-

[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]**ZEA**rch **LO**g p**U**bli**S**ing

tee (ϵ, δ)-indistinguishability [23], and we here offer a new analysis that shows how to set the parameters of ZEALOUS to guarantee *(ϵ, δ)-probabilistic differential privacy* [74] (Section 6.4.2), a much stronger privacy guarantee as our analytical comparison shows.

Our chapter 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 itemsets, as ZEALOUS protects privacy against much stronger attackers than those considered in existing work on privacy-preserving publishing of frequent items/itemsets [73].

The remainder of this chapter is organized as follows. We start with some background in Section 6.2. Our negative results are presented in Section 6.3. We then describe Algorithm ZEALOUS and its analysis in Section 6.4. We compare indistinguishability with probabilistic differential privacy in Section 6.5. Section 6.6 shows the results of an extensive study of how to set parameters in ZEALOUS, and Section 6.7 contains a thorough evaluation of ZEALOUS in comparison with previous work. We conclude with a discussion of related work and other applications.

6.2 Preliminaries

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

6.2.1 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 chapter. 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.

6.2.2 Disclosure Limitations for Publishing Search Logs

In this chapter we work with k-anonymity (see Section 2.1.2) and differential privacy and its relaxations (see Section 2.2.1).

6.2.3 Utility Measures

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

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 10 (($\mathcal{A}, S$)-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 11 (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.

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 et al. [10] 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 chapter 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 [4]. 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 [54]. Algorithms for query substitution examine query pairs to learn how users re-phrase queries. We use an algorithm developed by Jones et al. [54].

6.3 Negative Results

In this section, we discuss the deficiency of two existing models of disclosure limitations for search log publication. Section 6.3.1 focuses on k-anonymity, and Section 6.3.2 investigates differential privacy.

6.3.1 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. While it offers great utility even beyond releasing frequent items its disclosure guarantee might not be satisfactory. 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.

6.3.2 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 6.2.3 using our notion of accuracy). Our analysis is based on the following lemma.

Lemma 13. 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 13 follows directly from the definition of ϵ-differential privacy. Based on Lemma 13, 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 12. Consider an accuracy constant c, a threshold τ, a slack ξ and a very large domain $\mathcal{D}$ of size $\geq Um\left(\frac{2e^{2\epsilon(\tau+\xi)/m}}{c(\tau+\xi)} + \frac{1}{\tau-\xi+1}\right)$, 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 11) 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 11, we have that

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

By Lemma 13 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 $\left(|\mathcal{D}| - \frac{Um}{\tau-\xi+1}\right) c/(e^{2\epsilon(\tau+\xi)/m})$. For large domains of size at least $Um\left(\frac{2e^{2\epsilon(\tau+\xi)/m}}{c(\tau+\xi)} + \frac{1}{\tau-\xi+1}\right)$ the inaccuracy is at least $\frac{2Um}{\tau+\xi}$ which is greater than the inaccuracy of the baseline. $\square$

To illustrate Theorem 12, let us consider a search log S where each query contains at most 3 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 12, 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, no differentially private algorithm can be accurate for both very-frequent and very-infrequent query pairs.

6.4 Achieving Privacy

In this section, we introduce a search log publishing algorithm called ZEALOUS that has been independently developed by Korolova et al. [65] and us. 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. Figure 6.1 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 Lap(λ)[4], and add η_k to the count c_k, resulting in a

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

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

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.

To understand the purpose of the various steps one has to keep in mind the privacy guarantee we would like to achieve. Step 1., 2. and 4. of the algorithm are fairly standard. It is known that adding Laplacian noise to histogram counts achieves ϵ-differential privacy [25]. 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. Section 6.4.1 and Section 6.4.2 discuss the privacy guarantees of ZEALOUS with respect to (ϵ, δ)-indistinguishability and (ϵ, δ)-probabilistic differential privacy, respectively. Section 6.4.3 presents a quantitative analysis of the privacy protection offered by ZEALOUS. Sections 6.4.4 and 6.4.5 analyze the utility guarantees of ZEALOUS.

6.4.1 Indistinguishability Analysis

Theorem 13 states how the parameters of ZEALOUS can be set to obtain a sanitized histogram that provides (ϵ', δ')-indistinguishability.

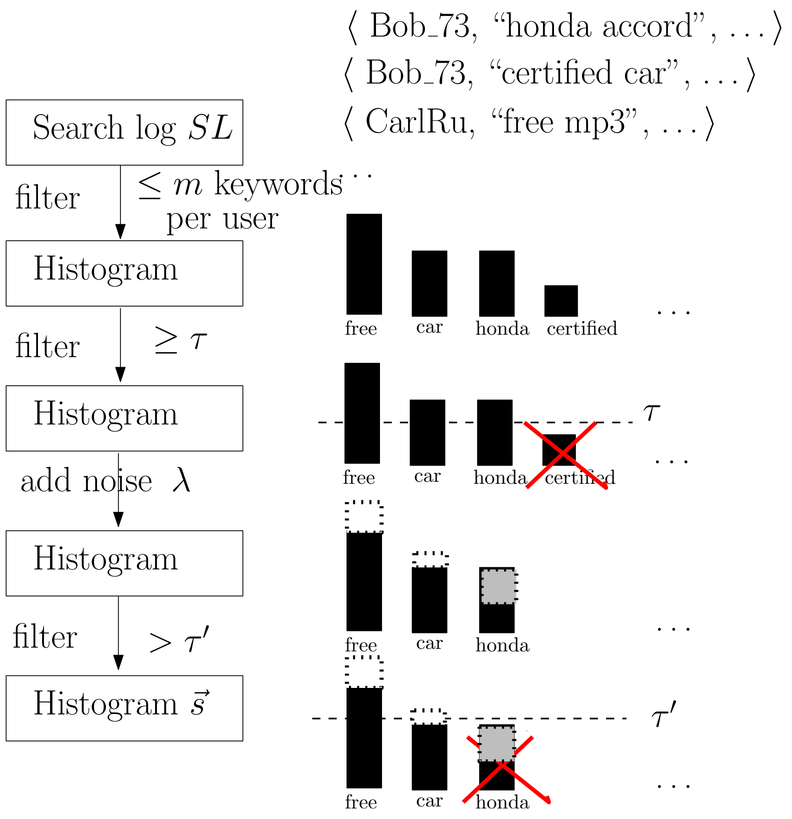


Figure 6.1: Privacy–Preserving Algorithm.

Theorem 13. *[65]* Given a search log S and positive numbers m, τ, τ', and λ, ZEALOUS achieves (ϵ', δ')-indistinguishability, if

$$\lambda \geq 2m/\epsilon', \text{and} \tag{6.1}$$

$$\tau = 1, \text{and} \tag{6.2}$$

$$\tau' \geq m\left(1 - \frac{\log(\frac{2\delta'}{m})}{\epsilon'}\right). \tag{6.3}$$

To publish not only frequent queries but also their clicks, Korolova et al. [65] 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.

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

$$\lambda \geq 2m/\epsilon, \text{ and} \tag{6.4}$$

$$\tau' - \tau \geq \max\left(-\lambda \ln\left(2 - 2e^{-\frac{1}{\lambda}}\right), -\lambda \ln\left(\frac{2\delta}{U \cdot m/\tau}\right)\right), \tag{6.5}$$

where U denotes the number of users in S.

Proof. Let H be the keyword histogram constructed by ZEALOUS in Step 2 when applied to S and K be the set of keywords in H whose count equals τ. Let Ω be the set of keyword histograms, that do not contain any keyword in K. For notational simplicity, let us denote ZEALOUS as a function Z. We will prove the theorem by showing that, given Equations (6.4) and (6.5),

$$Pr[Z(S) \notin \Omega] \leq \delta, \tag{6.6}$$

and for any keyword histogram $\omega \in \Omega$ and for any neighboring search log S' of S,

$$e^{-\epsilon} \cdot Pr[Z(S')=\omega] \leq Pr[Z(S)=\omega] \leq e^{\epsilon} \cdot Pr[Z(S')=\omega]. \tag{6.7}$$

We will first prove that Equation (6.6) holds. Assume that the i-th keyword in K has a count $\tilde{c}_i$ in $Z(S)$ for $i \in [1, |K|]$. Then,

$$\begin{aligned}
&Pr[Z(S) \notin \Omega] \\
&= Pr\Big[\exists i \in [1, |K|], \tilde{c}_i > \tau'\Big] \\
&= 1 - Pr\Big[\forall i \in [1, |K|], \tilde{c}_i \leq \tau'\Big] \\
&= 1 - \prod_{i \in [1, |K|]} \left(\int_{-\infty}^{\tau' - \tau} \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}} dx \right) \\
&\quad \text{(the noise added to } \tilde{c}_i \text{ has to be} \geq \tau' - \tau) \\
&= 1 - \left(1 - \frac{1}{2} \cdot e^{-\frac{\tau' - \tau}{\lambda}}\right)^{|K|} \\
&\leq \frac{|K|}{2} \cdot e^{-\frac{\tau' - \tau}{\lambda}} \\
&\leq \frac{U \cdot m}{2\tau} \cdot e^{-\frac{\tau' - \tau}{\lambda}} \qquad \text{(because } |K| \leq U \cdot m/\tau) \\
&\leq \frac{U \cdot m}{2\tau} \cdot e^{-\frac{-\lambda \ln\left(\frac{2\delta}{U \cdot m/\tau}\right)}{\lambda}} \qquad \text{(by Equation 6.5)} \\
&= \delta. \qquad (6.8)
\end{aligned}$$

Next, we will show that Equation (6.7) also holds. Let S' be any neighboring search log of S. Let ω be any possible output of ZEALOUS given S, such that $\omega \in \Omega$. To establish Equation (6.7), it suffices to prove that

$$\frac{Pr[Z(S) = \omega]}{Pr[Z(S') = \omega]} \leq e^{\epsilon}, \text{ and} \qquad (6.9)$$

$$\frac{Pr[Z(S') = \omega]}{Pr[Z(S) = \omega]} \leq e^{\epsilon}. \qquad (6.10)$$

We will derive Equation (6.9). The proof of (6.10) is analogous.

Let H' be the keyword histogram constructed by ZEALOUS in Step 2 when applied to S'. Let Δ be the set of keywords that have different counts in H and

H'. Since S and S' differ in the search history of a single user, and each user contributes at most m keywords, we have $|\Delta| \leq 2m$. Let k_i ($i \in [1, |\Delta|]$) be the i-th keyword in Δ, and d_i, d'_i, and d^*_i be the counts of k_i in H, H', and ω, respectively. Since a user adds at most one to the count of a keyword (see Step 2.), we have $d_i - d'_i = 1$ for any $i \in [1, |\Delta|]$. To simplify notation, let E_i, E'_i, and $E_i{}^*, E'^*_i$ denote the event that k_i has counts d_i, d'_i, d^*_i in H, H', and $Z(S), Z(S')$, respectively. Therefore,

$$\frac{Pr[Z(S) = \omega]}{Pr[Z(S') = \omega]} = \prod_{i \in [1, |\Delta|]} \frac{Pr[E_i{}^* \mid E_i]}{Pr[E'^*_i \mid E'_i]}.$$

In what follows, we will show that $\frac{Pr[E_i{}^*|E_i]}{Pr[E'^*_i|E'_i]} \leq e^{1/\lambda}$ for any $i \in [1, |\Delta|]$. We differentiate three cases: (i) $d_i \geq \tau$, $d^*_i \geq \tau$, (ii) $d_i < \tau$ and (iii) $d_i = \tau$ and $d^*_i = \tau - 1$.

Consider case (i) when d_i and d^*_i are at least τ. Then, if $d^*_i > 0$, we have

$$\begin{aligned} &\frac{Pr[E_i{}^* \mid E_i]}{Pr[E'^*_i \mid E'_i]} \\ &= \frac{\frac{1}{2\lambda} e^{-|d^*_i - d_i|/\lambda}}{\frac{1}{2\lambda} e^{-|d^*_i - d'_i|/\lambda}} \\ &= e^{(|d^*_i - d'_i| - |d^*_i - d_i|)/\lambda} \\ &\leq e^{|d_i - d'_i|/\lambda} \\ &= e^{\frac{1}{\lambda}}. \qquad \text{(because } |d_i - d'_i| = 1 \text{ for any } i\text{)} \end{aligned}$$

On the other hand, if $d^*_i = 0$,

$$\frac{Pr[E_i{}^* \mid E_i]}{Pr[E'^*_i \mid E'_i]} = \frac{\int_{-\infty}^{\tau' - d_i} \frac{1}{2\lambda} e^{-|x|/\lambda} dx}{\int_{-\infty}^{\tau' - d'_i} \frac{1}{2\lambda} e^{-|x|/\lambda} dx} \leq e^{\frac{1}{\lambda}}.$$

Now consider case (ii) when d_i is less than τ. Since $\omega \in \Omega$, and ZEALOUS eliminates all counts in H that are smaller than τ, we have $d^*_i = 0$, and $Pr[E^*_i \mid$

$E_i] = 1$. On the other hand,

$$Pr[E_i'^* \mid E_i'] = \begin{cases} 1, & \text{if } d_i' \leq \tau \\ 1 - \frac{1}{2}e^{-|\tau' - d_i'|/\lambda}, & \text{otherwise} \end{cases}$$

Therefore,

$$\begin{aligned} & \frac{Pr[E_i{}^* \mid E_i]}{Pr[E_i'^* \mid E_i']} \\ \leq & \quad \frac{1}{1 - \frac{1}{2}e^{-|\tau' - d_i'|/\lambda}} \\ \leq & \quad \frac{1}{1 - \frac{1}{2}e^{-(\tau' - \tau)/\lambda}} \\ \leq & \quad \frac{1}{1 - \frac{1}{2}e^{\ln\left(2 - 2e^{-\frac{1}{\lambda}}\right)}} \qquad \text{(by Equation 6.7)} \\ = & \quad e^{\frac{1}{\lambda}}. \end{aligned}$$

Consider now case (iii) when $d_i = \tau$ and $d_i^* = \tau - 1$. Since $\omega \in \Omega$ we have $d_i^* = 0$. Moreover, since ZEALOUS eliminates all counts in H that are smaller than τ, it follows that $Pr[E_i^* \mid E_i'] = 1$. Therefore,

$$\frac{Pr[E_i{}^* \mid E_i]}{Pr[E_i'^* \mid E_i']} = Pr[E_i{}^* \mid E_i] \leq e^{\frac{1}{\lambda}}.$$

In summary, $\frac{Pr[E_i{}^* | E_i]}{Pr[E_i'^* | E_i']} \leq e^{1/\lambda}$. Since $|\Delta| \leq 2m$, we have

$$\begin{aligned} & \frac{Pr[Z(S) = \omega]}{Pr[Z(S') = \omega]} \\ = & \quad \prod_{i \in [1, |\Delta|]} \frac{Pr[E_i{}^* \mid E_i]}{Pr[E_i'^* \mid E_i']} \\ \leq & \quad \prod_{i \in [1, |\Delta|]} e^{1/\lambda} \\ = & \quad e^{|\Delta|/\lambda} \\ \leq & \quad e^{\epsilon} \qquad \text{(by Equation 6.6 and } |\Delta| \leq 2m\text{)}. \end{aligned}$$

This concludes the proof of the theorem. □

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

Table 6.1: (ϵ', δ')-indistinguishability vs. (ϵ, δ)-probabilistic differential privacy. $U = 500k$, $m = 5$.

6.4.3 Quantitative Comparison of Prob. Diff. Privacy and Indistinguishability for ZEALOUS

In Table 6.1, 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 6.1 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 6.5.

6.4.4 Utility Analysis

Next, we analyze the utility guarantee of ZEALOUS in terms of its accuracy (as defined in Section 6.2.3).

Theorem 15. Given parameters $\tau = \tau^* - \xi, \tau' = \tau^* + \xi$, noise scale λ, and a search log S, the inaccuracy of ZEALOUS with slack ξ equals

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

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 Lap(λ)-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. $\square$

6.4.5 Separation Result

Combining the analysis in Sections 6.3.2 and 6.4.4, we obtain the following separation result between ϵ-differential privacy and (ϵ, δ)- probabilistic differential privacy.

Theorem 16 (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 non-zero probability (independent of the input database), its inaccuracy for large item domains is larger than an algorithm that always outputs an empty set.

6.5 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 [23] or [85].

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

Proposition 9. If an algorithm $\mathcal{A}$ is (ϵ, δ)-probabilistic differentially private then it is also (ϵ, δ)-indistinguishable.

Proof. Assume that, for all search logs S, we can divide the output space Ω into to two sets Ω_1, Ω_2, such that

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

for all search logs S' differing from S only in the search history of a single user and for all $O \in \Omega_1$:

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

Consider any subset $\mathcal{O}$ of the output space Ω of $\mathcal{A}$. Let $\mathcal{O}_1 = \mathcal{O} \cap \Omega_1$ and $\mathcal{O}_2 = \mathcal{O} \cap \Omega_2$. We have

$$\begin{aligned}
&\Pr[\mathcal{A}(S) \in \mathcal{O}] \\
&= \int_{O\in\mathcal{O}_2} \Pr[\mathcal{A}(S) = O]dO + \int_{O\in\mathcal{O}_1} \Pr[\mathcal{A}(S) = O]dO \\
&\leq \int_{O\in\Omega_2} \Pr[\mathcal{A}(S) = O]dO + e^{\epsilon}\int_{O\in\Omega_1} \Pr[\mathcal{A}(S') = O]dO \\
&\leq \delta + e^{\epsilon}\int_{O\in\Omega_1} \Pr[\mathcal{A}(S') = O]dO \\
&\leq \delta + e^{\epsilon} \cdot \Pr[\mathcal{A}(S') \in \Omega_1].
\end{aligned}$$

□

The converse of Proposition 9 does not hold. In particular, there exists an 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 6. *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 10. 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 $\mathcal{D}^U$.

Proof. We have to show that for all search logs S, S' differing in one user history and for all sets $\mathcal{O}$:

$$\Pr[\hat{\mathcal{A}}(S) \in \mathcal{O}] \leq \Pr[\hat{\mathcal{A}}(S') \in \mathcal{O}] + 1/(|\mathcal{D}| - 1).$$

Since Algorithm $\hat{\mathcal{A}}$ neglects all but the first input this is true for neighboring search logs not differing in the first user's input. We are left with the case of two neighboring search logs S, S' differing in the search history of the first user. Let us analyze the output distributions of Algorithm 1 under these two inputs S and S'. For all search histories except the search histories of the first user in S, S' the output probability is $1/(|\mathcal{D}| - 1)$ for either input. Only for the two search histories of the first user S_1, S'_1 the output probabilities differ: Algorithm 1 never outputs S_1 given S, but it outputs this search history with probability $1/(|\mathcal{D}| - 1)$ given S'. Symmetrically, Algorithm $\hat{\mathcal{A}}$ never outputs S'_1 given S', but it outputs this search history with probability $1/(|\mathcal{D}|-1)$ given S. Thus, we have for all sets $\mathcal{O}$

$$\Pr[\hat{\mathcal{A}}(S) \in \mathcal{O}] = \sum_{d \in \mathcal{O} \cap (\mathcal{D}-S_1)} 1/(|\mathcal{D}|-1) \tag{6.11}$$

$$\leq 1/(|\mathcal{D}|-1) + \sum_{d \in \mathcal{O} \cap (\mathcal{D}-S_2)} 1/(|\mathcal{D}|-1) \tag{6.12}$$

$$= \Pr[\hat{\mathcal{A}}(S) \in \mathcal{O}] + 1/(|\mathcal{D}|-1) \tag{6.13}$$

□

The next proposition shows that every single output of the algorithm constitutes a privacy breach.

Proposition 11. 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.

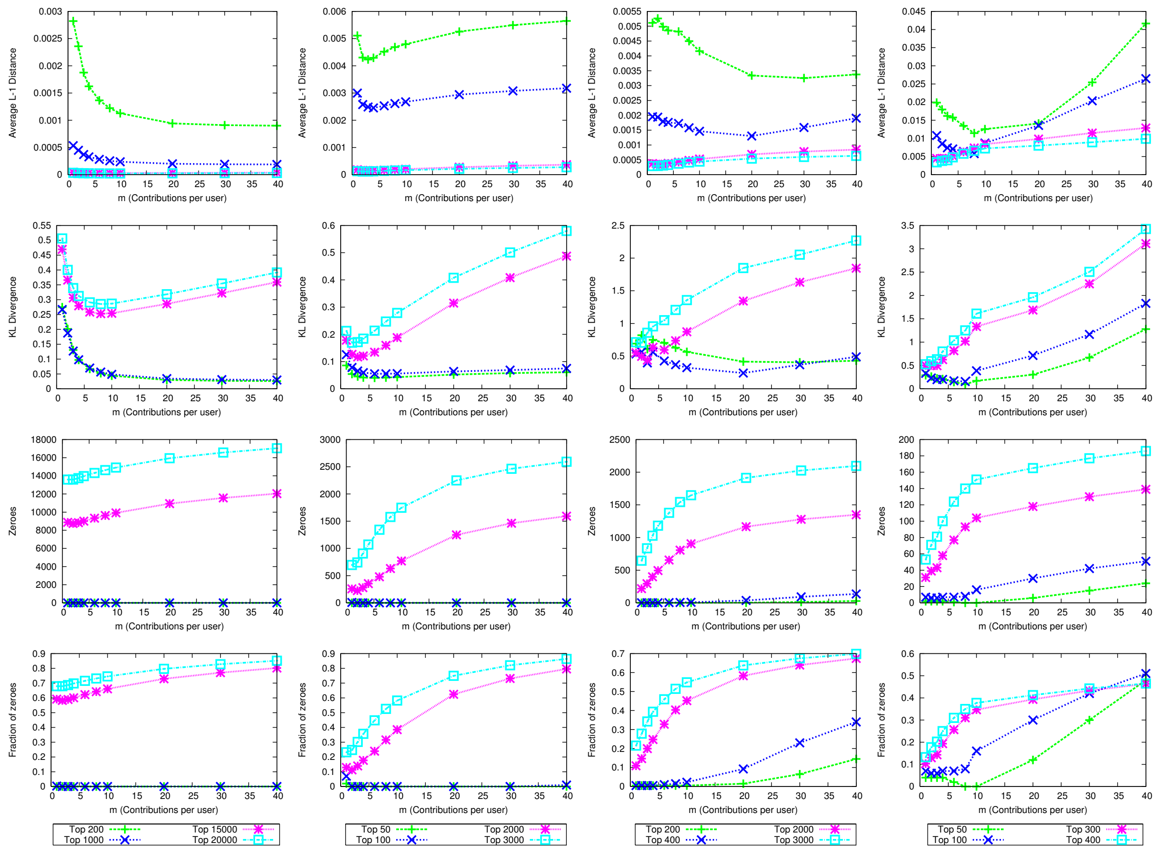
Proof. Fix an input S and an output O that is different from the search history of the first user. Consider the input S' differing from S only in the first user history, where $S'_1 = O$. Here,

$$1/(|\mathcal{D}|-1) = \Pr[\mathcal{A}(S) = O] \nleq e^{\epsilon} \Pr[\mathcal{A}(S') = O] = 0.$$

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

Corollary 3. 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 3, 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.

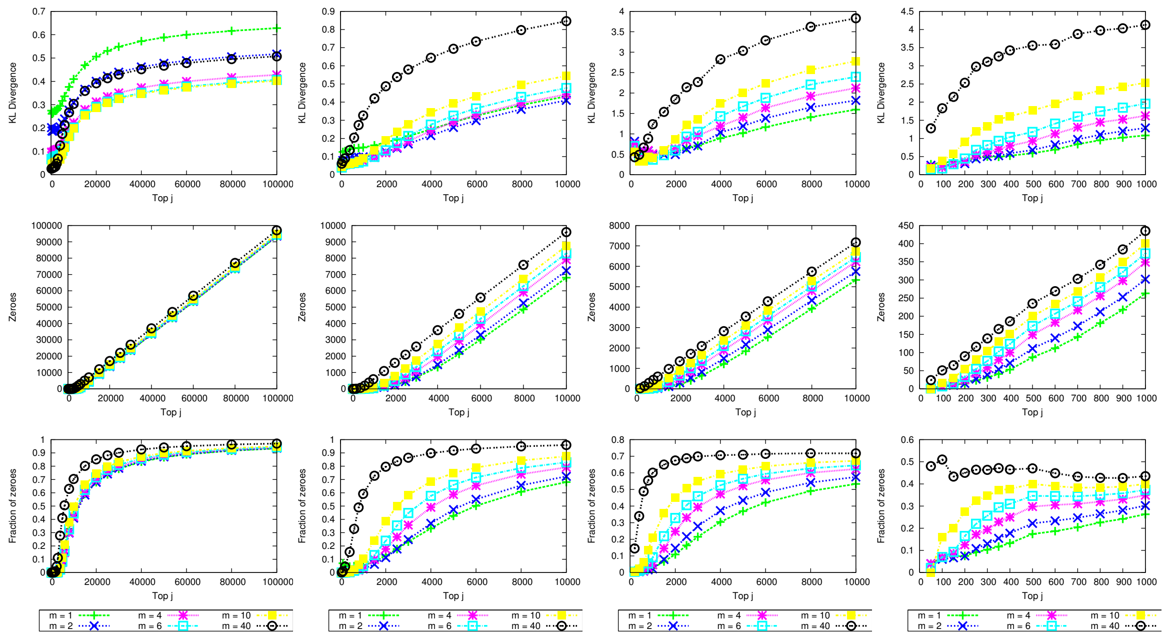


(a) Keywords (b) Queries (c) Clicks (d) Query Pairs

Figure 6.2: Preservation of the counts of the top-j most frequent items by ZEALOUS under varying m. The domain of items are keywords, queries, clicks, and query pairs. Preservation is measured as the average L1-distance and KL-divergence of the released counts and their true counts and the number and fraction of unpublished top-j most frequent items are shown.

6.6 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



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

Figure 6.3: Same as Figure 6.2 except that the x-axis is now varying j for top-j and not m for the number of contributions.

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.

τ	1	3	4	5	7	9
τ'	81.1	78.7	78.6	78.7	79.3	80.3

Table 6.2: τ' as a function of τ for $m = 2$, $\epsilon = 1$, $\delta = 0.01$

6.6.1 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 6.2 presents the value of τ' for different values of τ for $m = 2$ and $\epsilon = 1$. Table 6.2 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 12. 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 Equation (6.5)) to determine its minimum.

6.6.2 Choosing the Number of Contributions m

Proposition 12 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

(A) Number of distinct items released under ZEALOUS.

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) Number of total items $\times 10^3$ released under ZEALOUS.

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

Table 6.3: Releases of ZEALOUS

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 6.3 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% while at the same time the number of distinct query pairs decreases by 96% as we increase m from 1 to 40. This trend has two reasons. First, from Theorem 14 and Proposition 12 we see that threshold τ' increases super-linearly in m. Second, as m increases the number of keywords contributed by the users increases only sub-linearly in m; fewer users are able to supply m

keywords	queries	clicks	query pairs
56	20	14	7

Table 6.4: Avg number of items per user in the original search log

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, 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 6.4).

To understand how m affects precision, we measure the total sum of the counts in the sanitized histogram as we increase m in Table 6.3. 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 6.4). 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 Figures 6.2 and 6.3. In Figure 6.2 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 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.

Figure 6.3 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.

6.7 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 13 and 14 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.

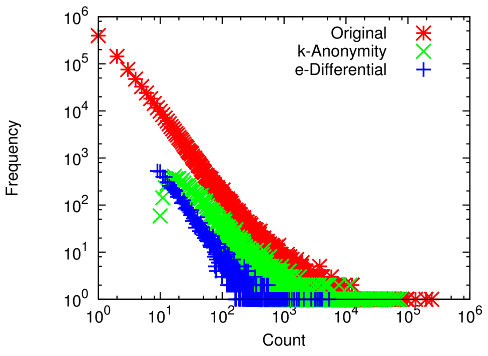
Evaluation Metrics.

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 utility evaluation with statistics in Section 6.7.1 and then our evaluation with real applications in Sections 6.7.2 and 6.7.3.

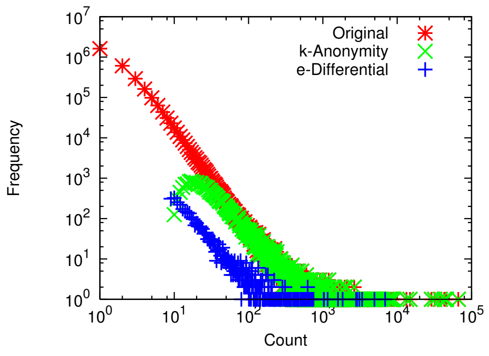
6.7.1 General Statistics

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.

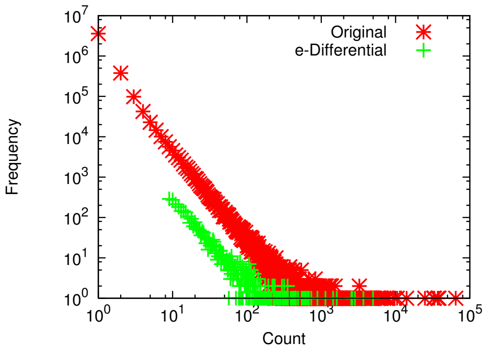
In our first experiment we compare the distribution of the counts in the his-



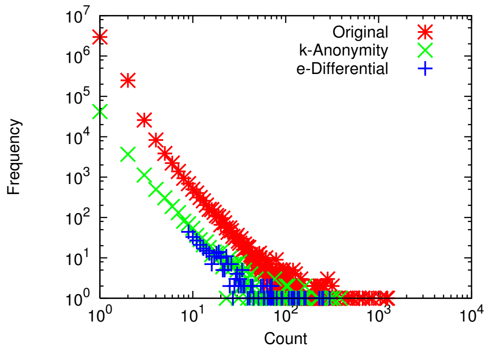
(a) Keyword Counts



(b) Query Counts



(c) Click Counts

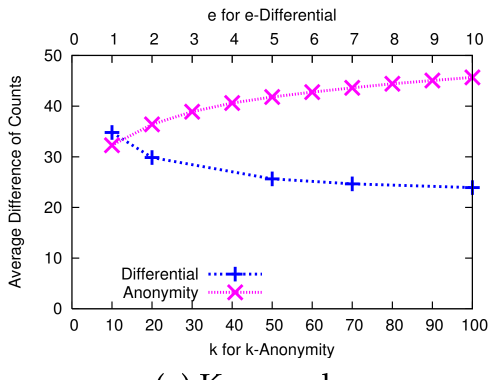


(d) Query Pair Counts

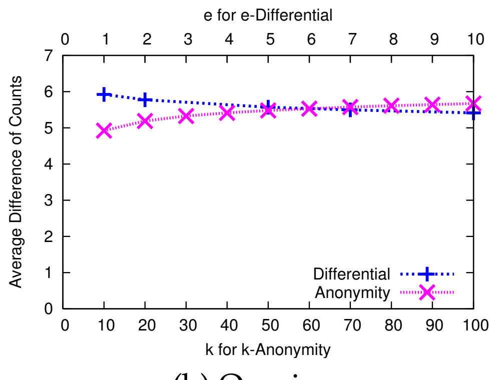
Figure 6.4: Distributions of counts in the histograms over keywords, queries, clicks, and query pairs in the original search log and its sanitized versions created by 10-anonymity and 5-probabilistic differential privacy (with $m = 1$).

tograms. 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 $k \approx \tau'$.

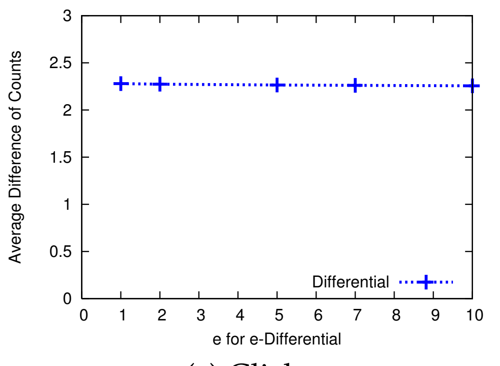
Figure 6.4 shows the distribution of the counts in the histograms on a log-log scale. Recall that the k-query anonymous search log does not contain any click data, and thus it does not appear in Figure 6.4(c). We see that the power-law shape of the distribution is well preserved. However, the total frequencies are lower for the sanitized search logs than the frequencies in the original histogram



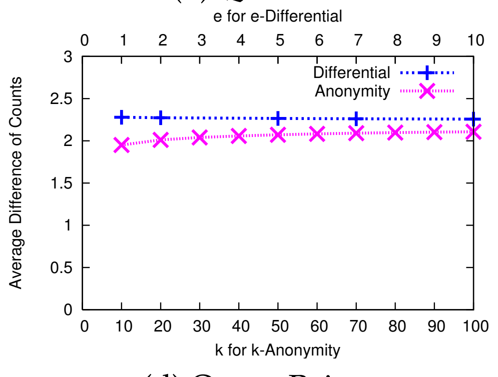
(a) Keywords



(b) Queries



(c) Clicks



(d) Query Pairs

Figure 6.5: Average difference between counts of items in the original histogram and the ϵ-probabilistic differential privacy-preserving histogram, and the k-anonymous histogram for varying parameters ϵ (with $m = 1$) and k.

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. Figure 6.5 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 non-zero 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. Figure 6.5 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.

6.7.2 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 Section6.2.3). In our

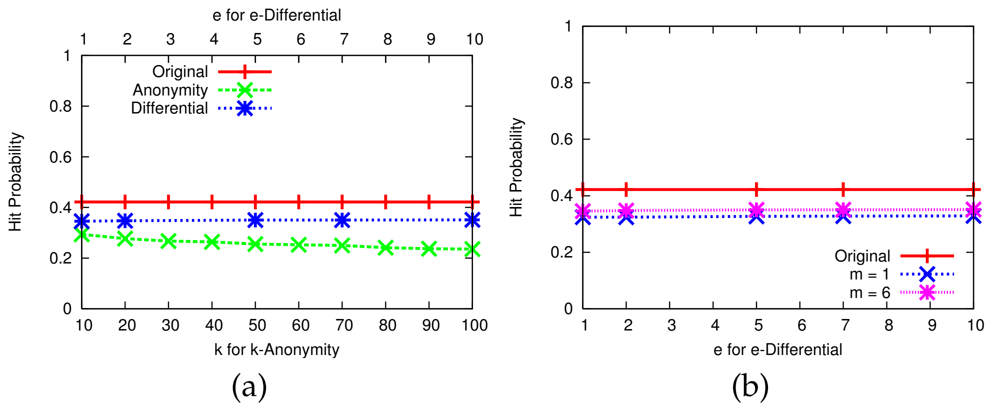


Figure 6.6: Hit probabilities of the inverted index construction based on the original search log, the k-anonymous search log, and the ZEALOUS histogram under varying parameters k and ϵ (and contributions m in (b)).

experiments, we use an improved version of the algorithm developed by Baeza–Yates to decide which posting lists should be kept in memory [4]. 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 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.

Figure 6.6(a) 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 Figure 6.6(b). 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.

6.7.3 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 [54]. 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, MAP (mean average precision) and NDG (normalized discounted cumulative gain) 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:

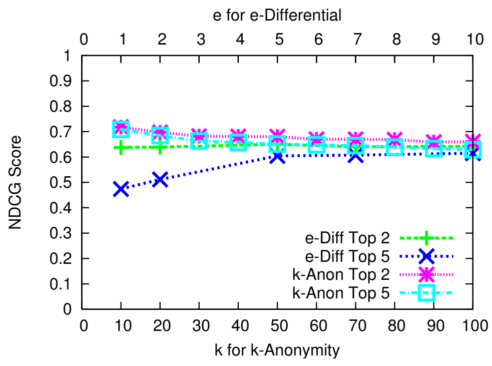
$$\text{MAP}(q) = \sum_{i=0}^{j-1} \frac{i+1}{\text{rank of } q_i \text{ in } [q'_0, \ldots, q'_{j-1}] + 1},$$

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

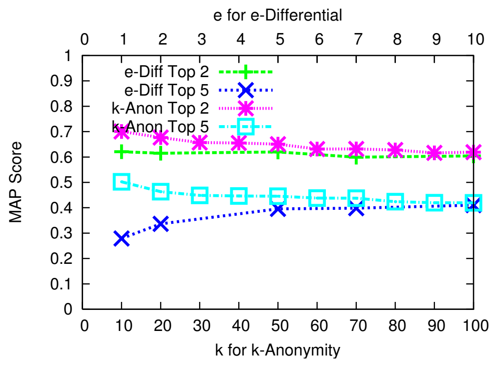
Our last metric called NDCG measures how the relevant substitutions are placed in the ranking list. It does not only compare the ranks of a substitution in the two rankings, but is also penalizes highly relevant substitutions according to $[q_0, \ldots, q_{j-1}]$ that have a very low rank in $[q'_0, \ldots, q'_{j-1}]$. Moreover, it takes the length of the actual lists into consideration. We refer the reader to the paper by Chakrabarti et al. [15] 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.

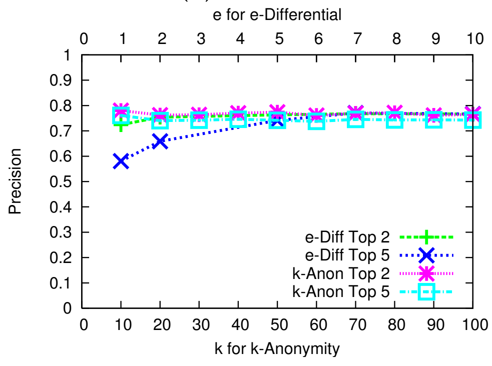
In Figure 6.7 we vary k and ϵ for $m = 1$ and we draw the utility curves for top-j for $j = 2$ and $j = 5$. We observe that varying ϵ and k has hardly any influence on performance. On all precision measures, ZEALOUS provides utility comparable to k-query-anonymity. However, the coverage provided by ZEALOUS 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.



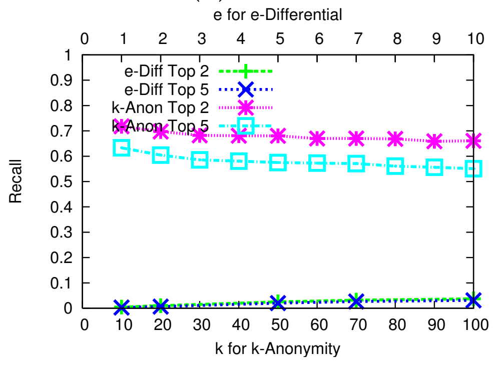
(a) NDCG



(b) MAP



(c) Precision



(d) Recall

Figure 6.7: Quality of the query substitutions of the ϵ-probabilistic differential privacy-preserving histogram, and the k-anonymous histogram for varying parameters ϵ (with $m = 1$) and k. The quality is measured by comparing the top-2 and top-5 suggested query substitutions to the ground truth recording the NDCG, MAP, precision, and recall.

As a last experiment, we study the effect of increasing m for query substitutions. Figure 6.8 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.

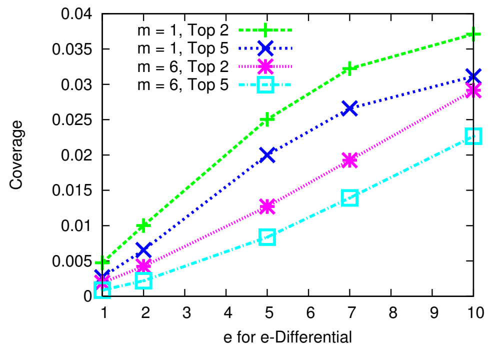


Figure 6.8: Coverage of the query substitutions of the privacy-preserving histogram for $m = 1$ and $m = 6$.

6.8 Related Work

Related work on anonymity in search logs [1, 47, 82, 50] is discussed in Section 6.3.1.

More recently, there has been work on privacy in search logs. Korolova et al. [65] proposes the same basic algorithm that we propose and review in Section 6.4.[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 6.5). 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. [65] set $\tau = 1$ (which is not the optimal choice in many cases). Our experiments augment and extend the experiments of Korolova et al. [65]. We illustrate the tradeoff of setting the number of contributions m for various domains and statistics including L1-distance and KL divergence which extends [65] greatly. Our application oriented evaluation considers different

[5]In order to improve utility of the algorithm as stated in [65], 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 3 frequent keywords and the clicks of their top ranked documents.

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.

6.9 Beyond Search Logs

While the main focus of this chapter 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 [43]. 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 / itemsets / consecutive itemsets. Existing work on publishing frequent itemsets 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. [73].

6.10 Conclusions

This chapter 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 itemsets.

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

BIBLIOGRAPHY

[1] Eytan Adar. User 4xxxxx9: Anonymizing query logs. In *WWW Workshop on Query Log Analysis*, 2007.

[2] Gagan Aggarwal, Mayank Bawa, Prasanna Ganesan, Hector Garcia-Molina, Krishnaram Kenthapadi, Rajeev Motwani, Utkarsh Srivastava, Dilys Thomas, and Ying Xu 0002. Two can keep a secret: A distributed architecture for secure database services. In *CIDR*, 2005.

[3] Arvind Arasu, Michaela Götz, and Raghav Kaushik. On active learning of record matching packages. In *SIGMOD*, 2010.

[4] Roberto Baeza-Yates. Web usage mining in search engines. *Web Mining: Applications and Techniques*, 2004.

[5] Bhuvan Bamba, Ling Liu, Peter Pesti, and Ting Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *WWW*, 2008.

[6] 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. In *PODS*, 2007.

[7] Michael Barbaro and Tom 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.

[8] Gerald Bieber, Jörg Voskamp, and Bodo Urban. Activity recognition for everyday life on mobile phones. In *HCI*, 2009.

[9] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[10] Justin Brickell and Vitaly Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *KDD*, 2008.

[11] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[12] Thorben Burghardt, Klemens Böhm, Achim Guttmann, and Chris Clifton. Anonymous search histories featuring personalized advertisement - balancing privacy with economic interests. *Transactions on Data Privacy*, 4(1):31–50, 2011.

[13] Georg Buscher, Susan T. Dumais, and Edward Cutrell. The good, the bad, and the random: an eye-tracking study of ad quality in web search. In *SIGIR*, 2010.

[14] Soumen Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *WWW*, 2007.

[15] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. Structured learning for non-smooth ranking losses. In *KDD*, pages 88–96, 2008.

[16] Bee-Chung Chen, Daniel Kifer, Kristen LeFevre, and Ashwin Machanavajjhala. Privacy-preserving data publishing. *Foundations and Trends in Databases*, 2(1-2):1–167, 2009.

[17] Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. Preserving user location privacy in mobile data management infrastructures. In *Privacy Enhancing Technologies*, 2006.

[18] Chi-Yin Chow and Mohamed F. Mokbel. Enabling private continuous queries for revealed user locations. In *SSTD*, 2007.

[19] Chi-Yin Chow, Mohamed F. Mokbel, Jie Bao, and Xuan Liu. Query-aware location anonymization for road networks. *GeoInformatica*, 15(3):571–607, 2011.

[20] Chi-Yin Chow, Mohamed F. Mokbel, and Xuan Liu. Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments. *GeoInformatica*, 15(2):351–380, 2011.

[21] Chi-Yin Chow, Mohamed F. Mokbel, Joe Naps, and Suman Nath. Approximate evaluation of range nearest neighbor queries with quality guarantee. In *SSTD*, 2009.

[22] Maria Luisa Damiani, Elisa Bertino, and Claudio Silvestri. The probe framework for the personalized cloaking of private locations. *Transactions on Data Privacy*, 3(2):123–148, 2010.

[23] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, volume 4004, 2006.

[24] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[25] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[26] N. Eagle, A. Pentland, and D. Lazer. Inferring social network structure using mobile phone data. *Proceedings of the National Academy of Sciences (PNAS)*, 106:15274–15278, 2009.

[27] William Enck, Peter Gilbert, Byung gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, 2010.

[28] Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, 2003.

[29] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Commun. ACM*, 39:77–85, 1996.

[30] Uriel Feige. A threshold of ln n for approximating set cover. *JOURNAL OF THE ACM*, 45:314–318, 1998.

[31] Matthew Fredrikson and Benjamin Livshits. Repriv: Re-envisioning in-browser privacy. Technical Report MSR-TR-2010-116, Microsoft Research, August 2010.

[32] William Gasarch. A survey on private information retrieval. *Bulletin of the EATCS*, 82:72–107, 2004.

[33] Bugra Gedik and Ling Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS*, 2005.

[34] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. MIT Press, 2007.

[35] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *GIS*, 2009.

[36] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD*, 2008.

[37] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Mobihide: a mobile peer-to-peer system for anonymous location-based queries. In *SSTD*, 2007.

[38] Sharad Goel, Andrei Z. Broder, Evgeniy Gabrilovich, and Bo Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. In *WSDM*, 2010.

[39] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*, 2003.

[40] Marco Gruteser and Xuan Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security and Privacy*, 2:28–34, 2004.

[41] Saikat Guha, Bin Cheng, and Paul Francis. Challenges in Measuring Online Advertising Systems. In *Proceedings of Internet Measurement Conference (IMC)*, Melbourne, Australia, Nov 2010.

[42] Saikat Guha, Alexey Reznichenko, Kevin Tang, Hamed Haddadi, and Paul Francis. Serving Ads from localhost for Performance, Privacy, and Profit. In *HotNets*, 2009.

[43] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 1st edition, September 2000.

[44] Moritz Hardt and Guy Rothblum. A multiplicative weights mechanism for interactive privacy-preserving data analysis. In *FOCS*, 2010.

[45] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.

[46] Yeye He, Siddharth Barman, Di Wang, and Jeffrey F. Naughton. On the

complexity of privacy-preserving complex event processing. In *PODS*, 2011.

[47] Yeye He and Jeffrey F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1):934–945, 2009.

[48] E. Kim S. Helal and D. Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1):48–53, 2010.

[49] Dorit Hochbaum and Anu Pathria. Analysis of the Greedy Approach in Problems of Maximum k-Coverage. *Naval Research Logistics*, 45(6):615–627, 1998.

[50] Yuan Hong, Xiaoyun He, Jaideep Vaidya, Nabil Adam, and Vijayalakshmi Atluri. Effective anonymization of query logs. In *CIKM*, 2009.

[51] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *WWW*, 2003.

[52] Christian S. Jensen, Hua Lu, and Man Lung Yiu. *Location Privacy Techniques in Client-Server Architectures*, pages 31–58. Springer-Verlag, 2009.

[53] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. "I know what you did last summer": query logs and user privacy. In *CIKM*, 2007.

[54] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *WWW*, 2006.

[55] Ari Juels. Targeted advertising ... and privacy too. In *CT-RSA*, 2001.

[56] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Trans. on Knowl. and Data Eng.*, 19:1719–1733, 2007.

[57] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.

[58] Patrick Gage Kelley, Michael Benisch, Lorrie Faith Cranor, and Norman M. Sadeh. When are users comfortable sharing locations with advertisers? In *CHI*, 2011.

[59] Ali Khoshgozaran and Cyrus Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *Proceedings of the 10th international conference on Advances in spatial and temporal databases*, SSTD'07, 2007.

[60] Ali Khoshgozaran and Cyrus Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *SSTD*, 2007.

[61] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. An anonymous communication technique using dummies for location-based services. *International Conference on Pervasive Services*, 0:88–97, 2005.

[62] Daniel Kifer. Attacks on privacy and definetti's theorem. In *SIGMOD*, 2009.

[63] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, 2011.

[64] Aleksandra Korolova. Privacy violations using microtargeted ads: A case study. In *Workshop on Privacy Aspects of Data Mining (PADM)*, 2010.

[65] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *WWW*, 2009.

[66] Andreas Krause and Eric Horvitz. A utility-theoretic approach to privacy and personalization. In *AAAI*, 2008.

[67] Alexander Kuenzer, Christopher Schlick, Frank Ohmann, Ludger Schmidt, and Holger Luczak. An empirical study of dynamic bayesian networks for user modeling. In *UM*, 2001.

[68] Ravi Kumar, Jasmine Novak, Bo Pang, and Andrew Tomkins. On anonymizing query logs via token-based hashing. In *WWW*, 2007.

[69] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *FOCS*, 1997.

[70] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, 2006.

[71] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *ICDE*, 2007.

[72] Tiancheng Li, Ninghui Li, and Jian Zhang. Modeling and integrating background knowledge in data anonymization. In *ICDE*, 2009.

[73] Yongcheng Luo, Yan Zhao, and Jiajin Le. A survey on the privacy preserving algorithm of association rule mining. *Electronic Commerce and Security, International Symposium*, 1:241–245, 2009.

[74] Ashwin Machanavajjhala, Daniel Kifer, John M. Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.

[75] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *TKDD*, 1(1), 2007.

[76] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[77] Andrea Mannini and Angelo Maria Sabatini. Accelerometry-based classification of human activities using markov modeling. *Computational Intelligence and Neuroscience*, 2011.

[78] David J. Martin, Daniel Kifer, Ashwin Machanavajjhala, Johannes Gehrke, and Joseph Y. Halpern. Worst case background knowledge for privacy preserving data publishing. In *ICDE*, 2007.

[79] Frank McSherry and Ratul Mahajan. Differentially-private network trace analysis. In *SIGCOMM*, 2010.

[80] Emiliano Miluzzo, Cory T. Cornelius, Ashwin Ramaswamy, Tanzeem Choudhury, Zhigang Liu, and Andrew T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *MobiSys*, 2010.

[81] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: A privacy-aware location-based database server. In *ICDE*, 2007.

[82] Rajeev Motwani and Shubha Nabar. Anonymizing unstructured data. *arXiv*, 2008.

[83] Shubha U. Nabar, Bhaskara Marthi, Krishnaram Kenthapadi, Nina Mishra, and Rajeev Motwani. Towards robustness in query auditing. In *VLDB*, 2006.

[84] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.

[85] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.

[86] Abhinav Parate and Gerome Miklau. A framework for safely publishing communication traces. In *CIKM*, 2009.

[87] Linda Pareschi, Daniele Riboni, Alessandra Agostini, and Claudio Bettini. Composition and generalization of context data for privacy preservation. In *PerComm*, 2008.

[88] V. Rastogi, D. Suciu, and S. Hong. The boundary between privacy and utility in data publishing. Technical report, University of Washington, 2007.

[89] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, 2010.

[90] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, 2001.

[91] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, CMU, SRI, 1998.

[92] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *CIKM*, 2005.

[93] Elaine Shi, T-H. Hubert Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.

[94] Eran Toch, Justin Cranshaw, Paul Hankes Drielsma, Janice Y. Tsai, Patrick Gage Kelley, James Springfield, Lorrie Cranor, Jason Hong, and

Norman Sadeh. Empirical models of privacy in location sharing. In *Ubicomp*, 2010.

[95] Vincent Toubiana, Helen Nissenbaum, Arvind Narayanan, Solon Barocas, and Dan Boneh. Adnostic: Privacy preserving targeted advertising. In *NDSS*, 2010.

[96] Janice Tsai, Patrick G Kelley, Lorrie F Cranor, and Norman Sadeh. Location sharing technologies: Privacy risks and controls. *I/S: A Journal of Law and Policy for the Information Societ*, 6(2), 2010.

[97] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Jia Liu, Ke Wang, and Yabo Xu. Global privacy guarantee in serial data publishing. In *ICDE*, 2010.

[98] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *VLDB*, 2007.

[99] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, Yabo Xu, Jian Pei, and Philip S. Yu. Probabilistic inference protection on anonymized data. In *ICDM*, 2010.

[100] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, Philip S. Yu, and Jian Pei. Can the utility of anonymized data be used for privacy breaches? *TKDD*, 5(3), 2011.

[101] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *SIGMOD*, 2007.

[102] Xiaokui Xiao, Yufei Tao, and Nick Koudas. Transparent anonymization: Thwarting adversaries who know the algorithm. *ACM Trans. Database Syst.*, 35(2), 2010.

[103] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2010.

[104] Toby Xu and Ying Cai. Location anonymity in continuous location-based services. In *GIS*, 2007.

[105] Yabo Xu, Ke Wang, Benyu Zhang, and Zheng Chen. Privacy-enhancing personalized web search. In *WWW*, 2007.

[106] Mingqiang Xue, Panos Kalnis, and Hung Keng Pung. Location diversity: Enhanced privacy protection in location based services. In *LoCA*, 2009.

[107] Jun Yan, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen. How much can behavioral targeting help online advertising? In *WWW*, 2009.

[108] Man Lung Yiu, Christian S. Jensen, Xuegang Huang, and Hua Lu. Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *ICDE*, 2008.

[109] Chengyang Zhang and Yan Huang. Cloaking locations for anonymous location based services: a hybrid approach. *Geoinformatica*, 13, June 2009.