UNIVERSITY OF CALIFORNIA,
IRVINE

# Differential Privacy for Non-standard Settings

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Joann Qiongna Chen

Dissertation Committee:
Professor Zhou Li, Chair
Professor Gene Tsudik
Professor Yanning Shen

2024

# DEDICATION

*To my God, whose faithfulness has been my rock.*
*To my family, whose unwavering support has fueled my academic pursuit.*
*To my friends, whose encouragement has illuminated the path of my PhD journey.*

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

To my advisor, Zhou Li, I am sincerely grateful for your unwavering support from Day 1 of my PhD journey. You have provided me with the invaluable privilege to pursue my research freely and with confidence, serving as a true role model throughout. Your patience, guidance, and support have empowered me to not only navigate the challenges of academia but also to aspire for further academic pursuits.

To my committee members, Gene Tsudik and Yanning Shen, it has been a privilege to have known and learned from you both. Your steadfast support and wise counsel have been invaluable throughout this journey.

To Athina Markopoulou, you have been a pivotal mentor throughout my journey. You have generously supported me during my job search journey and your example truly embodies what it means to be a mentor and educator.

Special thanks to my mentors and collaborators: Tianhao Wang, Somesh Jha, Kamalika Chaudhuri, Yang Zhang, Zhikun Zhang, Xinlei He, and Zheng Li. I am especially grateful to Tianhao Wang for teaching me the essential aspects of differential privacy and mentoring me through our initial research projects.

I want to express my gratitude to my friends, the GEECS community, my lab mates, and my fellow EECS and ICS cohort members for their support in balancing my research commitments with social events. Special thanks go to: Helmina Bong, Makena Crowe, Aaron San Juan, Elina van Kempen, Heather Lott, Victor Ong, Beverly Quon, Robert Marosi, Hieu Le, Floranne Ellington, Nilab Ismailoglu, Jiacen Xu, Nathan Furman, Danyu Sun, and Tom Wang. Additionally, I deeply appreciate the support and guidance from the EECS department staff throughout my PhD journey, especially: Amy Pham, Julie Strope, Lioubov Konkova, Jasmine Garcia, Marisa Mendoza, Beverly Randez, and Stephany Monterroso. I genuinely could not have made it this far without their support.

Thank you to the differential privacy community, who provided valuable insights and reviews on my projects. Thank you to the Seabreeze Church community, who have constantly encouraged and prayed for me over the past five years.

This thesis was partially supported by NSF CNS-2220434 and the European Health and Digital Executive Agency (HADEA) within the project "Understanding the individual host response against Hepatitis D Virus to develop a personalized approach for the management of hepatitis D" (D-Solve) (grant agreement number 101057917).

## Reprint Notice

Portions of this dissertation are reprints of, or largely based on, the materials in [41, 49], used with permission from the IEEE and ACM, respectively.

# VITA

## Joann Qiongna Chen

| | |
|---|---|
| **2024** | **Ph.D. in Electrical and Computer Engineering**<br>University of California, Irvine |
| **2020-24** | **Graduate Research Assistant**<br>University of California, Irvine |
| **2022** | **Research Scientist Intern**<br>Meta Reality Labs |
| **2022** | **Teaching Assistant**<br>University of California, Irvine |
| **2021** | **Research Intern**<br>CISPA Helmholtz Center for Information Security |
| **2019** | **M.S. in Electrical Engineering**<br>University of Southern California |
| **2017** | **B.Eng. in Telecommunication Engineering**<br>Xidian University |

### PUBLICATIONS


[1] **Joann Qiongna Chen**, Xinlei He, Zheng Li, Yang Zhang, Zhou Li."A Comprehensive Study of Privacy Risks in Curriculum Learning," arXiv:2310.10124.

[2] **Joann Qiongna Chen**, Tianhao Wang, Zhikun Zhang, Yang Zhang, Somesh Jha, Zhou Li. "Differentially Private Resource Allocation," *accepted by the 39th Annual Computer Security Applications Conference (ACSAC),* December, 2023.

[3] Deliang Chang*, **Joann Qiongna Chen***, Zhou Li, Xing Li. "Hide and Seek: Revisiting DNS-based User Tracking," *Proceedings of the 7th IEEE European Symposium on Security and Privacy (EuroS&P)*, June, 2022.

[4] Tianhao Wang, **Joann Qiongna Chen**, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. "Continuous Release of Data Streams under both Centralized and Local Differential Privacy," *Proceedings of the 28th ACM Conference on Computer and Communications Security (CCS)*, Virtual, Nov., 2021.

[5] Baojun Liu, Chaoyi Lu, Yiming Zhang, Zhou Li, Fenglu Zhang, Haixin Duan, Ying Liu, **Joann Qiongna Chen**, Jinjin Liang, Zaifeng Zhang, Shuang Hao, Min Yang. "From WHOIS to WHOWAS: A Large-Scale Measurement Study of Domain Registration Privacy under the GDPR," *In Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS)*, Virtual, Feb., 2021.


* First two authors contributed equally to this work

# ABSTRACT OF THE DISSERTATION

Differential Privacy for Non-standard Settings

by

Joann Qiongna Chen

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Irvine, 2024

Professor Zhou Li, Chair

In the increasingly digitized world, the intersection of data utilization and privacy protection presents significant challenges and opportunities. This dissertation explores the concept of Differential Privacy (DP), a framework that promises robust privacy protections while allowing the utility of data in diverse applications. Our research addresses the translation of DP from a theoretical construct into practical tools that can be integrated into real-world systems, focusing on DNS resolution and resource allocation.

One of the core advancements presented in this work is the development of a differentially private DNS resolution method that significantly reduces tracking accuracy rates with a provable guarantee. This is complemented by a prototype that the public can easily install on their local machines.

In the domain of resource allocation, we introduce novel differentially private mechanisms designed for environments such as cloud computing, virtual machine allocation, and network bandwidth management. These mechanisms not only ensure the confidentiality of sensitive metadata but also maintain system performance by integrating noise distribution techniques that optimize the trade-off between privacy protection and resource utility. This part of the study provides a comprehensive analysis of how differential privacy can be pragmatically applied to manage resources efficiently while adhering to stringent privacy standards,

showcasing empirical results that support the feasibility of these approaches.

Additionally, the research broadens the scope of privacy-enhancing technologies beyond DP, exploring their application in machine learning.

Through rigorous empirical studies and innovative system design, this dissertation not only contributes to the academic field but also aims to influence real-world practices by enhancing the privacy and utility of systems in which large volumes of personal data are processed. The implications of this dissertation offer directions for future work in securing digital interactions and promoting a safer, more transparent digital environment. We anticipate the widespread adoption of privacy-preserving technologies across multiple sectors, promoting a balanced approach to data privacy that is adaptable to the changing digital landscape.

# Chapter 1

# Introduction

## 1.1 Motivation

Personal data stands at the crossroads of progress and peril as technologies grow so rapidly. Its immense potential, if harnessed appropriately, can revolutionize numerous areas, from healthcare to finance. However, misusing this same data can lead to serious privacy violations. For example, the Netflix Prize competition was intended to improve its collaborative filtering algorithm, but the sequel was canceled following a privacy lawsuit [210] . While domain registration services provide a valuable directory, they can sometimes clash with the directives of the General Data Protection Regulation (GDPR) [167] . We are also seeing a rise in global surveillance with increasing instances of cross-border monitoring, yet our legal frameworks are ill-equipped to address this. Similarly, the U.S. lacks a unified federal law for consumer data tracking. Another pressing issue is the growing use of Artificial Intelligence (AI) and the privacy concerns around it. Recognizing its importance, the White House released an Executive Order on October 30, 2023, to ensure the safe and responsible development of AI [105]. These evolving trends pose significant challenges to personal privacy and digital rights. This dissertation seeks to navigate this privacy quandary, aiming to

harness data's potential while safeguarding individual privacy.

DP has emerged as a beacon of hope in this uncertain digital age [62] . While highlighting the importance of safely using AI, the recent presidential order also emphasized the power of DP [105]. Essentially, DP promises a strong provable privacy guarantee, ensuring that an individual's data footprint has minimal influence on the resulting output. This unique feature not only thwarts adversaries armed with extensive data knowledge but also offers opportunities for numerous real-world applications.

The concept of DP, initially hailed as a theoretical marvel within academic circles, has witnessed a tremendous surge in research, marked by notable theoretical advances and empirical enhancements. While my primary research focus is translating this academic theory into tangible, real-world tools, it is evident that the broader community has yet to embrace its practical implementation fully. This discrepancy becomes particularly clear when considering the relatively modest number of real-world deployments, as illustrated in Figure 1.1. For example, differentially private stochastic gradient descent (DP-SGD) has demonstrated significant promise in machine learning [13]. However, the accuracy of models trained with DP-SGD often leaves room for improvement. While recent research shows that it's feasible to train differentially private machine learning models that maintain high accuracy, this usually incurs substantial computational overhead. This overhead,

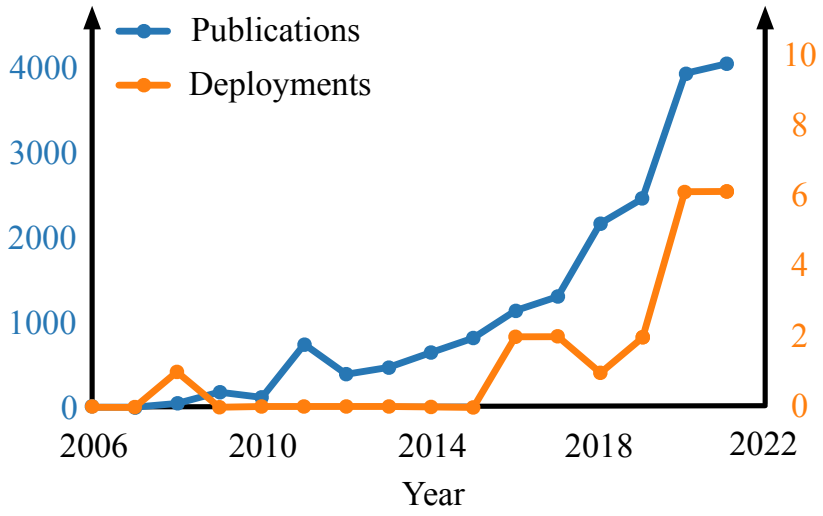



Figure 1.1: Comparison of DP Publications to Known Deployments (2006 – 2021) [57].

especially prohibitive for users with limited resources, hinders the broader deployment of DP-SGD.

## 1.2 Contributions

The outline and contributions of this thesis are as follows:

### 1.2.1 LDPResolve: Local Differentially Private DNS Resolution

In Chapter 3 and [41], we explore adapting local differential privacy (LDP) into Domain Name System (DNS) resolution to bound privacy leakage. We first study DNS-based user tracking in both open-world and closed-world settings to understand the root causes of tracking and how naive defenses may compromise legitimate applications that rely on user DNS records, such as malicious domain detection. To mitigate the DNS-based user tracking without damaging the legitimate DNS applications, we propose to integrate LDP into DNS. Based on a novel LDP notion, $(\mathcal{X}_{\mathcal{S}}, \mathcal{Y}_{\mathcal{P}}, \epsilon_1, \epsilon_2)$-ULDP [174], and parallel domain resolving, we design LDPResolve to solve the dilemma of DNS data utility and privacy. We then evaluate LDPResolve on a real-world DNS dataset and report our discoveries. A client-side prototype of LDPResolve is also developed. We demonstrate that the DNS-based user tracking can be effectively curbed with the deployment of LDPResolve, e.g., tracking accuracy degraded from 93% to 10.1%. This chapter presents the first attempt adapting LDP into DNS resolution. Our study suggests the threats coming from the DNS-based user tracking should be mitigated and it is feasible to protect users' privacy without damaging the utility of legitimate applications.

### 1.2.2 DPRA: Differentially Private Resource Allocation

In Chapter 4 and [49], we first study the existing resource allocators and find that they have no or very limited privacy guarantees against allocation-based side channel attacks. After investigating an existing resource allocation design that offers differential privacy [19], we found that the mechanism assumes the attacker knows the total number of requests after DP noise is added. However, we observe that the practical attacker only has a partial view of the Resource Allocator (RA). Therefore, we choose to model resource allocation privacy from the attacker's perspective. Due to the randomness introduced by the RA, we benefit from "privacy amplification" through such modeling and achieve a better privacy-utility tradeoff. In summary, we conduct a rigorous privacy analysis of differentially private resource allocation and derive tighter privacy bounds from the attacker's perspective for four noisy mechanisms: CST, UNI, GEO, and DGEO. We theoretically and empirically evaluate our proposed mechanisms. Our mechanism, GEO, achieves the best privacy-utility tradeoff and significantly outperforms the baseline.

### 1.2.3 Privacy Risks in Curriculum Learning

In Chapter 5 and [48], we take a quantitative approach to measure the privacy risks of curriculum learning (CL). CL, which trains a machine learning model with data following a meaningful order, i.e., from easy to hard, has been proven to be effective in accelerating the training process and achieving better model performance. We take the first attempt to investigate the privacy risk in CL despite its great success and deployment in crucial areas like image and text classification. More specifically, we select two popular CL methods, bootstrapping [89] and transfer learning [239], as the evaluation objects. Additionally, we constructed two other curricula, named baseline curriculum and anti-curriculum, to understand the impact of data ordering and repetition, respectively. We use nine real-world,

large-scale datasets (six image datasets and three tabular datasets), train target models with those CL methods and a normal method, and attack the models with representative membership inference attack (MIA) methods. Regarding MIA, our evaluation shows that the target models become slightly more vulnerable under CL. For example, the average attack accuracy (trained on ResNet-18 with transfer learning) on our selected image datasets ranges from 0.01% to 2.46%. More importantly, we find CL has a much bigger impact on the samples within the difficult group compared to the easy group, with the biggest gap of 4.23% in terms of attack accuracy for CIFAR100 (ResNet-18 is the architecture). This observation holds for both image and non-image datasets. Finally, we study the effectiveness of existing DP defense methods designed for privacy-preserving machine learning models and find that they are still effective in CL setting.

# Chapter 2

# Background

In recent years, numerous privacy incidents have highlighted the precarious balance between technological advancement and the protection of personal data. One notable example is the Netflix Prize competition, which aimed to refine recommendation algorithms but was abruptly canceled following a privacy lawsuit that exposed the risks of anonymized data [210]. Additionally, GDPR-compliant domain registration services have highlighted the complexities of data privacy regulations [167]. The rapid growth of Artificial Intelligence (AI) has also introduced new dimensions to privacy issues [105]. This dissertation explores some of these critical privacy challenges by introducing differential privacy (DP) to various applications and examining privacy issues in machine learning.

## 2.1 Differential Privacy

### 2.1.1 A Primer on Differential Privacy

In the central setting, a trusted data curator adds noise (e.g., through the Laplace mechanism or other mechanisms) to fulfill a DP notion (e.g., $(\epsilon, \delta)$-DP) given a query from a data

consumer, which bounds the information leakage provably.

**Definition 2.1** ($(\epsilon,\delta)$-Differential Privacy [61]). *An algorithm $\mathcal{M}$ satisfies $(\epsilon,\delta)$-differential privacy against an adversary, where $\epsilon,\delta \geq 0$, iff for any two neighboring datasets $D$ and $D'$, and any subset $Y$ of all possible outcomes of algorithm $\mathcal{M}$, we have*

$$\mathsf{Pr}[\mathcal{M}(D) \in Y] \leq e^{\epsilon}\mathsf{Pr}[\mathcal{M}(D') \in Y] + \delta \tag{2.1}$$

We consider two datasets $D$ and $D'$ to be neighbors, denoted as $D \simeq D'$ if and only if $D = D^{'} + u$ or $D^{'} = D + u$, where $D + u$ denotes the dataset resulted from adding one user's data $u$ to the dataset $D$. $\epsilon$ measures privacy loss at a differential change in data, which is also called privacy budget. $\delta$ models the probability when the algorithm $\mathcal{M}$ fails to be differentially private, which is also called "failure probability". The value of $\delta$ is normally very small in order to keep the algorithm satisfying DP most of the time. When $\delta = 0$, we simplify the $(\epsilon,0)$-DP to $\epsilon$-DP and call it pure DP.

**Definition 2.2** ($\epsilon$-Local Differential Privacy [235]). *An algorithm $\mathcal{M}_L$ satisfies $\epsilon$-local differential privacy ($\epsilon$-LDP), where $\epsilon > 0$, if and only if for any pair of input $x_1$ and $x_2 \in D$, we have*

$$\forall y \in \text{Range}(\mathcal{M}_L) : \mathsf{Pr}[\mathcal{M}_L(x_1) = y] \leq e^{\epsilon}\mathsf{Pr}[\mathcal{M}_L(x_2) = y] \tag{2.2}$$

*where* $\text{Range}(\mathcal{M}_L)$ *denotes the set of all possible output results of an algorithm $\mathcal{M}_L$.*

LDP has seen strong adoption from the industry. Companies like Google [68], Apple [7] and Samsung [184] have developed their own LDP implementation to compute aggregated user statistics in a privacy-preserving way. The data collection protocol under LDP consists of three steps [235]: **Encode** (users report their answers in a specific format), **Perturb** (the

answers are randomized), and **Aggregate** (the answers are merged and decoded to obtain statistics, e.g., item frequency). In our work, we focus on the LDP protocols that support frequency estimation (e.g., Google's RAPPOR [68]).

### 2.1.2 Properties

The following composition properties hold for both DP and LDP algorithms, each commonly used for building complex differentially private algorithms from simpler subroutines.

**Sequential Composition.** Combining multiple subroutines that satisfy DP for $\epsilon_1, \cdots, \epsilon_k$ results in a mechanism that satisfies $\epsilon$-DP for $\epsilon = \sum_i \epsilon_i$.

**Parallel Composition.** Given $k$ algorithms working on disjoint subsets of the dataset, each satisfying DP for $\epsilon_1, \cdots, \epsilon_k$, the result satisfies $\epsilon$-DP for $\epsilon = \max_i \epsilon_i$.

**Post-processing.** Given an $\epsilon$-DP algorithm $\mathcal{M}$, releasing $g(\mathcal{M}(V))$ for any $g$ still satisfies $\epsilon$-DP. That is, post-processing an output of a differentially private algorithm does not incur any additional loss of privacy.

# Chapter 3

# LDPResolve: Local Differentially Private DNS Resolution

## 3.1 Introducation

Domain name system (DNS) translates human-readable domain names to machine-readable IP addresses. It is an essential component of the Internet infrastructure as DNS queries underpin almost every user's Internet activity. Nowadays, *trillions* of users' requests on a single day are processed by DNS [30].

Under normal configuration, a user's DNS queries will be sent to a recursive resolver, which acts as an agent to obtain the authoritative answers from authoritative nameservers. The plaintext information of every query is visible to recursive resolvers, even when DNS encryption like DNS-over-HTTPS [110] or DNS-over-TLS [108] is used (the requests are only encrypted between users and recursive resolvers). Hence, a recursive resolver holding a large amount of DNS logs also poses privacy threats to users. One prominent threat is *user tracking* [195], through which a user's activities across different networks and devices can

be correlated, for purposes like personalized advertisement, surveillance, etc. This threat is more acute nowadays as public resolvers handle the lion's share of users' DNS requests and are well-motivated to launch user tracking [31, 109, 10, 104].

**Understanding DNS-based User Tracking.** There are in general two types of setting when considering DNS-based user tracking: *closed-world* and *open-world* settings. In the closed-world setting, all possible victim users must be known to the adversary, and it has been a well studied area [101, 100, 128, 132, 102, 129, 219]. Tracking in the *open-world* setting (i.e., the user sending DNS requests may be unknown) has been studied in [41]. More specifically, this chapter focuses on the problem of DNS-based tracking as discussed in [41]. This problem is formulated as assigning a DNS session (or a sequence of DNS queries within a short period) to a user, exploiting the similarity of DNS behaviors from the same user by utilizing the contextual correlation between domains.

**Mitigation with differentially private domain resolving.** To prevent user tracking, a straightforward solution is to add random dummy queries to the original users' queries. Spreading users' queries across resolvers is another solution, which has been exercised in K-resolver [104]. However, we argue that the threat is not adequately addressed by these solutions, as they may introduce significant overhead at the client-side to achieve a certain level of privacy. For instance, DNS queries have to be spread out to a large number of resolvers to defend against tracking, as discussed in Section 3.4.6. A greater concern is that these solutions impair the utility of DNS data irreversibly, and such DNS data is critical to legitimate applications like malicious domain detection [251]. To address the tension between utility and privacy, we make the *first* attempt to integrate differential privacy [60], a method that controls utility loss under privacy guarantee, to the process of domain resolution. We term our defense LDPResolve, which changes the behavior of a stub resolver under *Local Differential Privacy (LDP)* so recursive resolver does not need to be trusted. Yet applying LDP to our setting has to address two prominent challenges. First, take frequency estimation

(the primary usage scenario of LDP) as an example, only Direct Encoding (DE) of LDP [235] can be chosen to avoid revamping DNS protocol, but it will incur very high utility loss. Second, under the default randomized response protocol, the user has to issue false DNS queries, but doing so will give the user the wrong DNS response and break every Internet application.

To tackle these challenges, we propose a novel *($\mathcal{X}_\mathcal{S}$,$\mathcal{Y}_\mathcal{P}$,$\epsilon_1$,$\epsilon_2$)-ULDP (Utility-optimized LDP)* protocol, which is adjusted from ULDP [174], as the base of our defense. Our key insight is that though both user tracking and legitimate applications inspect domain names, different domain names have different levels of importance to them (e.g., popular domain names are important to user tracking but less so for malicious domain detection). ($\mathcal{X}_\mathcal{S}$,$\mathcal{Y}_\mathcal{P}$,$\epsilon_1$,$\epsilon_2$)-ULDP allows us to assign different privacy budgets for DNS queries. Under ($\mathcal{X}_\mathcal{S}$,$\mathcal{Y}_\mathcal{P}$,$\epsilon_1$,$\epsilon_2$)-ULDP protocol, we adapt *parallel domain resolving* to address the issue of false DNS queries. Hence, users always obtain accurate responses.

Based on the evaluation, we found LDPResolve can achieve the desired outcome: the tracking accuracy can be significantly reduced while the utility loss is controlled to a certain level. To highlight, the tracking accuracy of DSCorr [41] can be degraded from 93% to 10.1% while the utility loss measured by the standard deviation of unpopular domains is less than 10.

## 3.2 Background

### 3.2.1 DNS Communications and Dataset

DNS (Domain Name System) queries are issued before most network activities to map a user-friendly domain name (e.g., `www.google.com`) to an IP address (e.g., `216.58.193.196`). In particular, a user-end software named *stub resolver* receives the user's requests produced by

other applications and forwards them to a *recursive resolver* if the responses are not cached. The recursive resolver can be an Internet Service Provider (ISP) resolver serving users within the same network or a public resolver (e.g., Google Public DNS [2]) serving users all over the Internet. It further forwards the requests to *authoritative nameservers*, which is organized in a hierarchical structure and provides authoritative answers to the iterative queries.

Users' DNS traffic between their stub resolvers and authoritative nameservers results in a wealth of information valuable to applications like malicious domain detection [29, 153, 247, 52] and Internet traffic estimation [86]. Several organizations are gathering those telemetry data and sharing it with other parties under the concept of Passive DNS, mainly through two approaches. The first is to place a sensor array between recursive resolvers and authoritative nameservers, such that only the DNS lookups resulting in cache miss on recursive resolver are captured, and the client IP addresses are not seen. The Security Information Exchange (SIE) of FarSight [11] is operated under this model. As clients' requests are aggregated by each recursive resolver, a prior study suggests users' privacy is not violated when the sensors are configured properly [217]. The second approach is to fetch raw DNS logs *directly from recursive resolvers* and share the logs with client IP anonymized [152]. DNS Pai Project maintained by Qihoo 360 is operated under this mode [9]. While it enables more powerful applications, like finding abnormal domain associations [152], it could also raise privacy issues like user tracking. In this work, we thoroughly study such risks and propose a new approach to protect the end-users.

### 3.2.2 DNS-based User Tracking

In this dissertation, user tracking is defined as linking users' network activities across different networks without their consent, for purposes like personalizing advertisements or surveillance. When the network activities are DNS communications, user tracking can be done by linking raw DNS logs collected and shared by the recursive resolvers. Though such tracking

is trivial when a user uses a static IP address, many ISPs assign dynamic IP addresses that change periodically to their customers, so the adversary needs to re-identify the user after his/her IP address is changed. Also, the user could move between different ISP networks. Yet, a large number of users can be impacted due to the consolidation of DNS resolvers and the increasing dependency on the public resolvers [193], which can be queried wherever the users are. Here are some attack scenarios: 1) the users use a public resolver configured by their ISP (e.g., campus IT without DNS infrastructure chooses google public DNS); 2) the users' browsers set a default public resolver so all DNS traffic about the users is collected (e.g., DNS-over-HTTPS resolver in Google Chrome [3]); 3) an ISP exchanges DNS data from its resolver with another ISP; 4) a company retrieves DNS logs from multiple resolvers, e.g., DNS Pai mentioned in Section 3.2.1.

Some research has been done on DNS-based user tracking. Herrmann et al. [100] utilized classification methods such as Bayesian classifier and k-nearest neighbor to identify the re-occurrence of users based on their DNS queries. In [102, 132], the authors used a modified k-means algorithm to cluster DNS logs of the same user. DNSMiner [128, 129] re-identified the user by extracting unique and repetitive fingerprints from DNS traces. Sun et al. [219], proposed a method called constrained Dirichlet multinomial mixture for clustering DNS sessions without knowing the number of users in advance.

In this dissertation, we choose the attack methods from [100] and [41] as both cover closed and open-world settings and can easily be extended to more general scenarios, such as dynamic IP assignment or open-world setting.

### 3.2.3 DSCorr: DNS Session Correlation with Domain Embedding

We first present the threat model from [41], followed by the tracking method DSCorr, which is designed to tailor DNS session correlation based on domain embedding.

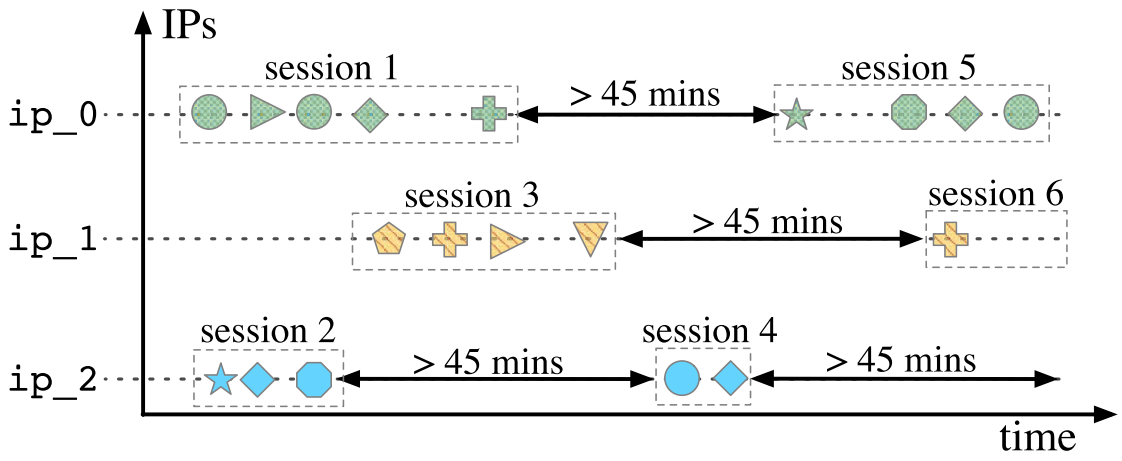



Figure 3.1: An example of sessions constructed from DNS queries. Different colors represent different source IP addresses, and different shapes represent different queried domain names.

**Threat Model.** Following the settings in existing works [41, 100, 102, 128, 129, 132, 219], we make two key assumptions: 1) Although the attacker has access to raw DNS logs, they cannot access DHCP logs, making it difficult to link a user's dynamic IP addresses to their ID. 2) Tracking users behind a NAT is beyond the scope of this dissertation. Under NAT, one source IP address (or its anonymized version) can be associated with multiple users, complicating accurate user profiling, linking, and labeling (for evaluation purposes).

**Exiting Defenses.** To defeat DNS-based tracking, a user can employ privacy-preserving DNS resolution techniques like Adaptive DNS [131] or Oblivious DNS [205]. However, these techniques are still in the early stages of adoption, and our threat model assumes they are not used by the victims. It is important to note that DNS encryption mechanisms, such as DNS-over-TLS and DNS-over-HTTPS, do not prevent adversaries in our setting (i.e., the resolver), as DNS packets are decrypted by the resolver. Ironically, these mechanisms might even enhance the adversary's capability in user tracking since most of the resolvers supporting these protocols are centralized [31, 109, 10, 104].

**DNS Session.** Following previous work [41], the general assumption is that the attacker

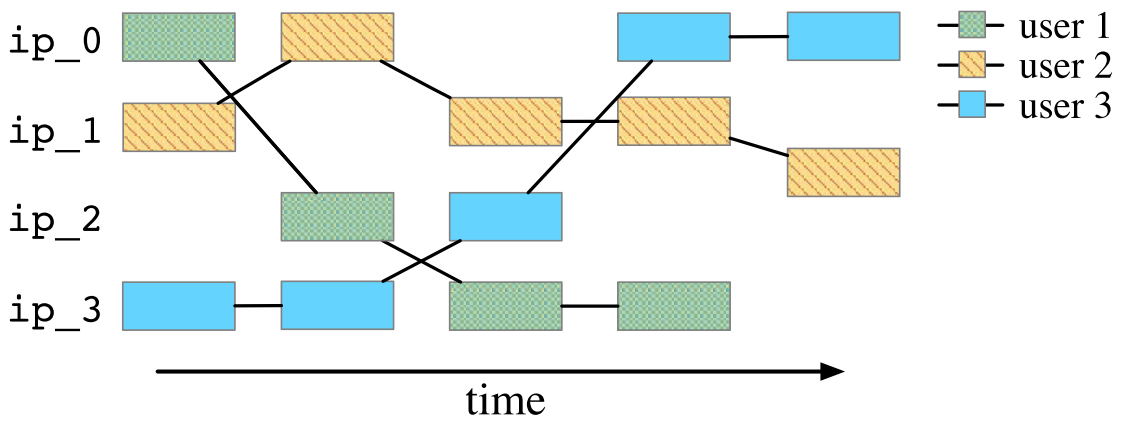



Figure 3.2: The objective of the adversary, who aims to correctly link a DNS session to its requesting user, regardless of IP churns.

constructs DNS sessions from the raw DNS logs and performs data linking at the session level. Without considering multiple users sharing the same IP address at the same time (e.g., under NAT), a DNS session is defined as a sequence of DNS queries associated with a source IP address within a period and issued by the same user [41]. A new session is created for a DNS query and its following queries if the previous DNS query was observed at least 45 minutes earlier. According to empirical analysis in [41], a device typically continues sending DNS requests as long as it stays online. A different device may use the IP address only if it has not been used for a period of time, usually longer than 45 minutes. Figure 3.1 illustrates how sessions are created from a stream of queries. Notably, a session does not have to contain all DNS queries from a source IP, and its covered time period is variable. Two consecutive sessions could belong to the same user. This setting is more flexible than previous works that set a session to be 24 hours [128, 129], and it avoids errors caused by IP churns across users. User tracking happens after session construction. Figure 3.2 illustrates the process of linking user sessions for tracking purposes.

**Open-world and Closed-world Settings.** Many previous works have studied the *closed-world setting* [100, 128, 129, 35, 102, 132, 219], while more recent research has begun exploring the *open-world setting* [234, 41]. These two settings are well-defined in the literature of

website fingerprinting, and we use the definitions proposed by Wang et al. [234]:

- In the closed-world setting, users are assumed to visit only a set of web pages known to the attacker, and the attacker must determine which of these pages were visited.
- In the open-world setting, users can visit any web pages, and the attacker must also identify pages not included in the known set.

To extend this definition to DNS-based tracking:

- In the closed-world setting, a DNS session originates from users within a set known to the attacker, and the attacker must identify which user issued the session.
- In the open-world setting, a DNS session can come from any user, and the attacker must recognize when the session is from a user not in the known set.

Generally, the open-world setting is considered more realistic yet challenging. As pointed out by Wang et al. [234], classifying pages with a *low base rate* is error-prone. Similarly, users with a low base rate (i.e., users who issue only a few sessions) also exist in DNS-based tracking.

**Ethics.** The DNS dataset used in this dissertation is provided by the IT department of an [anonymized] campus, which manages the campus DNS resolvers. The source IPs are *hashed before* the dataset is given to us. As the campus also runs a DHCP server, the user ID behind each source IP is known to the IT department. To help us build the ground truth, the IT department also hashes the user IDs and provides us the mapping between them and the source IPs. Notably, the hashed user IDs are only used to verify the effectiveness of the tracking methods employed in this dissertation; they are not used as input for any of the tracking methods. Following similar treatment of DNS datasets in other works [152, 86, 190], the raw data and all intermediate data are processed and stored on a server located on the

same campus, with strict access control. The use of the dataset in this dissertation is directly authorized by the campus administration office. Therefore, only the code of LDPResolve is made public, without the dataset.

**DSCorr Terms.** Let a DNS session be $s$ and a domain be $d$. The sequence of domains requested by $s$ can be represented as $<d_1,...,d_j,...,d_m>$, where $d_j$ is the $d$ of the jth request and $m$ is the number of requests in $s$. The user $u$ behind a session $s$ has either been labeled by the adversary before the tracking attack or is unknown. Let the sessions already labeled be **SL**, and the sessions unlabeled be **SU**. In the closed-world setting, the users behind **SL** and **SU** are identical. In the open-world setting, they might not be identical.

**Workflow of DSCorr.** The design of DSCorr is based on semi-supervised learning to handle the open-world setting. At a high level, DSCorr leverages **SL** as the training dataset and creates *session clusters* grouped by the labeled user IDs. Then, an $s$ in **SU** will be assigned to a session cluster in **SL**, or classified as "unknown". Four steps are carried out.

1. Each $d$ is converted to a numerical vector through domain embedding [169].
2. Sessions in **SL** are grouped to create labeled session clusters and build profiles for them. A cluster consists of sessions of the same user.
3. Given an $s$ in **SU**, the $k$ nearest session clusters are identified through a data-sketching process.
4. Fine-grained distances between $s$ to all neighboring session clusters are computed. $s$ will be clustered to a session cluster in **SL** with the minimum distance in the closed-world setting. For the open-world setting, $s$ might be classified as "unknown" if it is far from any session clusters.

### 3.2.4 Domain importance

For DSCorr and other methods, the similarity between sessions largely depends on the proportion of domains shared among them. Thus we are motivated to study the properties of such domains, in particular, whether they are popular among users. To this end, for each session $s_i$, we compare it to the next session $s_{i+1}$ of the same user, and categorize the domains appearing in both sessions as *shared* and the domains in $s_i$ only as *unique*. Figure 3.3 shows the distribution of shared and unique domains in terms of session-wise popularity, defined by the number of sessions in our dataset the domain appears in. It turns out shared domains are more likely to be popular: 62.7% shared domains are popular, while the number is 35.4% for unique domains. Domains ranked after 10k have a 97.6% chance of being unique.

The result indicates *popular domain names have higher importance for tracking*, which may be counter-intuitive. We speculate the main reason is that the combination of popular domains is more powerful in linking sessions of the same user. Our observation is also echoed by Kim et al. [128] that a large number of users cannot be fingerprinted by unique domains. Noticeably, DSCorr did not track users with user-specific domains (e.g., "ephemeral" or "disposable" domains) [17, 50], and the integration of this method can be an important future work.

**Other Types of Tracking.** Most of the other works in tracking looked into *web-based (or browser-based)* tracking. In its basic form, web code like JavaScript from a third-party content provider attempts to link user's visits based on tagging [133, 14, 22] or fingerprinting [135, 65, 64]. Tagging stores an identifier to a browser and attempts to make it persistent, even when the user blocks or frequently clears HTTP cookie. For example, Flash cookie and HTML5 session storage have been leveraged for this purpose. However, as surveyed by Bujlow et al., those tagging methods are ineffective under private browsing [34]. Still, recent work showed by filling DNS cache with responses customized to each user, private brows-

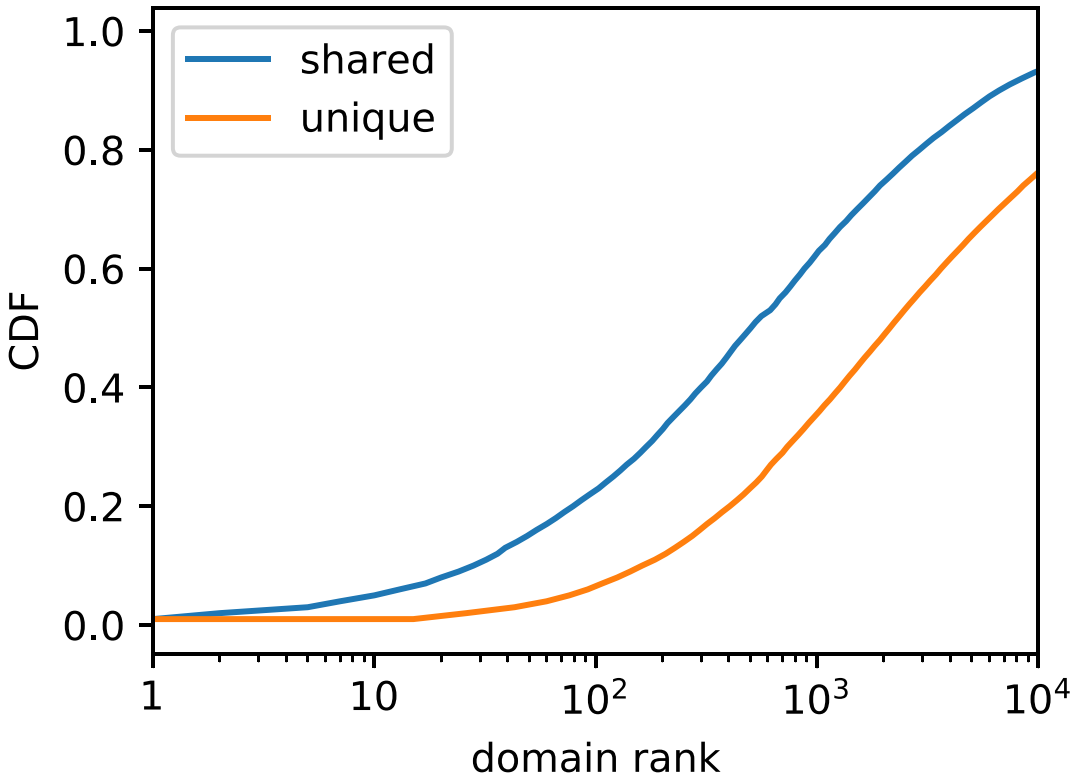



Figure 3.3: CDF of "shared" and "unique" domain for top 10k domains. Domains are ranked by session-wise popularity.

ing can be circumvented [133]. For fingerprinting, the properties unique to each browser instance (e.g., clock skew of the device [135] and fonts installed in the operating system [65]) are analyzed passively for identification. While prior research focused on tracking a user on a single device, a recent study shows cross-device tracking is possible [255].

Different from web-based tracking, *network-based tracking* looks into the characteristics of network flows and maps them to the user. For example, Kumpost et al.[141] built user profile based on HTTP/HTTPS/SSH traffic. Verde et al.[229] leveraged characteristics of Netflow to track users behind the NAT. Previous works showed that machine-learning approaches using packet sizes and intervals can correlate flows of the same user, even when the network flows are encrypted and mixed under Tor [179]. A related attack is website fingerprinting, which identifies the website visited by a targeted user from the encrypted traffic with network

features similar to flow correlation. Protocols like HTTPS [141], DNS-over-TLS [106] and DNS-over-HTTPS [209] are found vulnerable under this attack.

A few other works also use DNS to mine users' characteristics under very different settings. DNS cache-based tracking [133] uses JavaScript code to query domains and exploits the client-side DNS cache to tag a user, while we passively analyze the DNS dataset. DefecTor [85] exploits DNS packets for website fingerprinting, while DSCorr focuses on re-identifying users. DSCorr also differs from [112], which reveals sensitive queries by an institution using the logged DNS queries between recursive resolvers and authoritative name servers, while we use DNS queries between stub resolvers and recursive resolvers.

### 3.2.5 Differential Privacy

We leverage differential privacy (DP) mechanisms to protect users from DNS-based user tracking. Given a query from a data consumer [60], the original idea of DP assumes there is a trusted data curator adding noises to the result under a DP notion (also called Central DP).

Different from the central setting, local differential privacy (LDP, as defined in Definition 2.2 assumes there is no trusted data curator, and the noises are added by the data providers (e.g., Internet users) before the data are collected by the curator. The user enjoys better privacy as the data curator needs not be trusted, but the data utility is often worse than central DP under the same privacy budget.

In this dissertation, we focus on the LDP protocols that support frequency estimation (e.g., Google's RAPPOR [68]).

## 3.3 Domain Resolution under LDP

In this section, we propose a new privacy-preserving domain resolution method to protect the user against DSCorr and other tracking methods, while avoiding drastic changes to the client-side and server-side DNS infrastructure. We first overview the design of our proposed defense, LDPResolve, and its motivation. Then, we elaborate how LDP can be adapted in DNS resolution so that the privacy leakage is bounded. In the next section, we evaluate the effectiveness of LDPResolve.

### 3.3.1 Overview of LDPResolve

At a high level, similar to other works against traffic fingerprinting [21], LDPResolve adds noise to DNS queries from the users' end. Moreover, while user privacy is essential for tracking mitigation, maintaining DNS data utility is fundamental for legitimate third-party applications. Therefore, a defense that ignores data utility is unlikely to be supported by DNS service providers. This is a similar situation to crowdsourcing statistics [68], where Internet companies need to collect data from users at large while abiding by privacy laws. Legislation like the Do Not Track Act [5] is directly related to DNS-based user tracking. Thus, we envision our design also helping companies avoid legal issues while using DNS data. Our study shows that Local Differential Privacy (LDP) has the potential to maintain both data utility and user privacy simultaneously. Therefore, we develop LDPResolve, which adapts LDP for DNS resolution.

For the legitimate applications to be modeled under LDPResolve, we focus on malicious domain detection, which extensively leverages DNS data [251]. After literature survey, we found the frequency of domain visits is an important detection feature used by many research works [29, 153, 247, 52, 147, 51, 77, 194, 220]. Hence, we select the LDP protocols used for frequency estimation [235].

**Challenges.** Applying LDP to our setting involves overcoming two prominent challenges. First, though **Encode** of LDP (see Section 3.2.5) have different options (i.e., 5 options are listed in [235]), *Direct Encoding (DE)* is the only practical option for encoding DNS queries, as all other methods have to change the format of a request[1], which are unlikely to be supported by DNS stakeholders. On the other hand, DE often results in much higher utility loss than other methods when the cardinality of the answer set is large [235]. For example, the variance of the estimation under DE can be *two orders of magnitude* larger than other methods, when the privacy budget is tight and the answer set cardinality is high (see Table 2 of [235]). In our setting, users can resolve any domain among **billions** of the registered domains, resulting in an unbearable error margin potentially.

Second, randomized response [237] is the default communication protocol under DE, in which a user gives a false answer to the data curator at a certain probability. In our setting, randomized response means replacing the domain to be queried (e.g., `google.com`) with another domain drawn from a list (e.g., `yahoo.com`). Though it is expected to reduce tracking accuracy, because the user's DNS behaviors are altered, all Internet applications relying on DNS are likely to be broken.

**Solutions.** For the first challenge, it can be addressed through extending a recently proposed LDP concept, *Utility-optimized LDP (ULDP)* [174] (see Definition 3.1). The key insight behind ULDP is that not all data are equally sensitive (e.g., the answer "Yes" is more sensitive to the question "Have you ever cheated in an exam?" than the answer "No" [174]), and the non-sensitive data output can be protected in a lesser way (i.e., adding less noise). Overall, ULDP provides much better utility when non-sensitive data are dominant.

**Definition 3.1** ($\mathcal{X}_\mathcal{S}$,$\mathcal{Y}_\mathcal{P}$,$\epsilon$)-ULDP [174]**.** *Given $\mathcal{X}_\mathcal{S} \in \mathcal{X}$, $\mathcal{Y}_\mathcal{P} \in \mathcal{Y}$ , and $\epsilon \in \mathbb{R}_{\geq 0}$, an obfuscation mechanism $\mathcal{M}_o$ from $\mathcal{X}$ to $\mathcal{Y}$ provides ($\mathcal{X}_\mathcal{S}$,$\mathcal{Y}_\mathcal{P}$,$\epsilon$)-ULDP if it satisfies the following properties:*

[1]For example, assuming $v$ is the answer provided by a user, when Unary Encoding (UE) [235] is used, **Encode**$(v) = [0,...,0,1,0,...,0]$, a binary vector where only $v$-th position is 1. When DE is used, **Encode**$(v) = v$.

1. *For any* $y \in \mathcal{Y}_{\mathcal{I}}$*, there exists an* $x \in \mathcal{X}_{\mathcal{N}}$ *such that*

$$\mathcal{M}_o(y \mid x) > 0 \text{ and } \mathcal{M}_o(y \mid x') = 0 \text{ for any } x' \neq x. \tag{3.1}$$

2. *For any* $x, x' \in \mathcal{X}$ *and any* $y \in \mathcal{Y}_{\mathcal{P}}$,

$$\mathcal{M}_o(y \mid x) \leq e^{\epsilon} \mathcal{M}_o(y \mid x') \tag{3.2}$$

*where* $\mathcal{X}$ *(resp.* $\mathcal{Y}$ *) is a finite set of personal (resp. obfuscated) data.* $\mathcal{X}_{\mathcal{S}} \in \mathcal{X}$ *is a set of sensitive data common to all users, and* $\mathcal{X}_{\mathcal{N}} = \mathcal{X} \setminus \mathcal{X}_{\mathcal{S}}$ *is the remaining personal data.* $\mathcal{Y}_{\mathcal{P}} \in \mathcal{Y}$ *is a set of protected data, and* $\mathcal{Y}_{\mathcal{I}} = \mathcal{Y} \setminus \mathcal{Y}_{\mathcal{P}}$ *is a set of invertible data.*

We extend ULDP to enable two-level privacy protection. For applications like malicious domain detection, popular domains like `google.com` are usually not scrutinized because they are quite unlikely to be malicious [187, 29, 77]. On the other hand, they are "anchors" to user tracking, connecting sessions of the same user, as examined in Section 3.2.4. As such, we treat the popular domains as sensitive data, adding higher-degree noise to them under ULDP. The remaining unpopular domains are treated as non-sensitive data and be processed with lower-degree noise to keep good data utility.

The second challenge can be addressed by *parallel domain resolving*: given a domain name $d$ to query, a user can send $d$ to an alternative resolver (termed AltRR) when the risk of user tracking is high, and send a dummy query $d'$ to the primary resolver. The idea of dispersing DNS queries for better privacy was described in [38, 104]. Compared to previous works, our mechanism is fine-tuned with LDP. As such, users' privacy can be guaranteed under a privacy budget $\epsilon$. In addition, LDPResolve ensures the DNS data is usable by legitimate applications that rely on aggregated statistics.

**Workflow.** Figure 3.4 illustrates the design and workflow of LDPResolve, following the

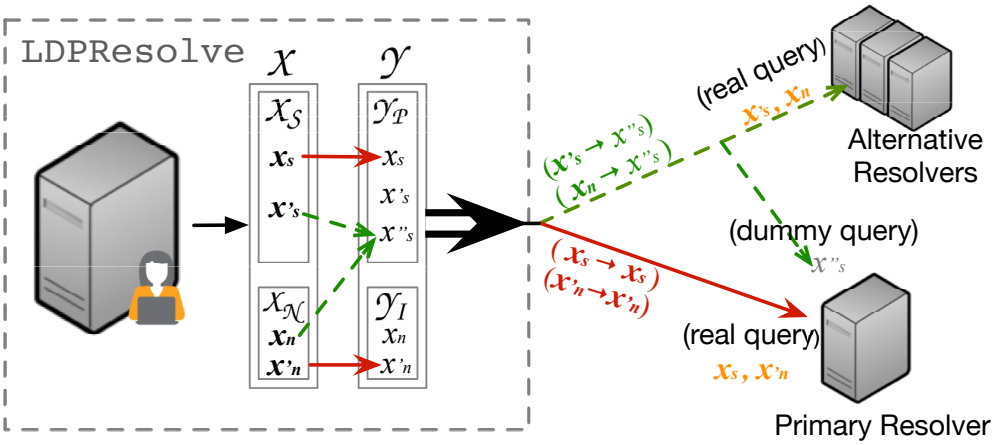



Figure 3.4: Workflow of LDPResolve. The symbols are defined in Section 3.3.2. A DNS query might be perturbed and sent to AltRR based on $(\mathcal{X}_{\mathcal{S}},\mathcal{Y}_{\mathcal{P}},\epsilon_1,\epsilon_2)$-ULDP.

concept of randomized response. When the stub resolver is about to send a DNS query for domain $d$ to a resolver configured by the user (i.e., primary resolver), she asks an oracle integrated by the stub resolver whether $d$ should be perturbed, depending on a generated probability and whether $d$ is listed in a *sensitive set* (set of popular domains). The sensitive set is generated by a third-party, e.g., Alexa [8] or authoritative nameservers, and it is periodically delivered to the stub resolver. When it is determined to be perturbed, it sends a different query with domain name $d'$ to the primary resolver and the original $d$ to an AltRR to obtain the authentic answer.

**AltRR.** AltRR can be another resolver that is not colluding with the primary resolver. It can be a local recursive resolver (RR), which directly talks to authoritative nameservers. Instead of altering the entire DNS resolution [205], we use a combination structure of AltRR and Primary RR, applying a DP-based approach to rationally distribute queries among RR. In doing so, unlike previous work that considered only privacy, we can achieve a balance between privacy, performance, and data utility. We expect the usage of AltRR will not significantly increase the latency. Firstly, Hoang et al. showed that DNS resolution is slightly longer with AltRR [104]. Secondly, only domains in the sensitive set go through AltRR and we can use the prefetch strategy since the sensitive set is known beforehand. In

Section 3.4.7, we evaluate one AltRR implementation based on a local resolver.

As described in Section 2.1.1, **Encode**, **Perturb**, and **Aggregate** are the three key steps for an LDP protocol. Since we choose DE for **Encode**, **Aggregate** becomes a trivial process as no extra decoding is needed [235]. **Perturb** needs to be designed in light of ULDP and we elaborate it next.

### 3.3.2 Perturb for LDPResolve

Let $\mathcal{X}$ be a set of DNS queries and $\mathcal{Y}$ be the perturbed queries. We use a randomized mechanism $\mathcal{A}$ to map $x \in \mathcal{X}$ to $y \in \mathcal{Y}$ with probability $\mathbf{P}(y|x)$. We divide DNS queries into sensitive queries about popular domains (termed $\mathcal{X}_\mathcal{S} \subseteq \mathcal{X}$) and non-sensitive queries about unpopular domains (termed $\mathcal{X}_\mathcal{N} \subseteq \mathcal{X}$). After perturbation, $\mathcal{Y}_\mathcal{P} \subseteq \mathcal{Y}$ and $\mathcal{Y}_\mathcal{I} \subseteq \mathcal{Y}$ are generated, which are associated with popular domains and unpopular domains respectively. We design $\mathcal{A}$ to satisfy a new DP notation $(\mathcal{X}_\mathcal{S},\mathcal{Y}_\mathcal{P},\epsilon_1,\epsilon_2)$-ULDP, as defined in Definition 3.2.

**Definition 3.2** ($(\mathcal{X}_\mathcal{S},\mathcal{Y}_\mathcal{P},\epsilon_1,\epsilon_2)$-ULDP)**.** *Given $\mathcal{X}_\mathcal{S},\mathcal{X}_\mathcal{N} \subseteq \mathcal{X}$, $\mathcal{Y}_\mathcal{P},\mathcal{Y}_\mathcal{I} \subseteq \mathcal{Y}$ and $\epsilon_1,\epsilon_2 \in \mathbb{R}_{\geq 0}$, a randomize mechanism $\mathcal{A}$, where $\mathcal{A}(X) = Y$, provides* ***$(\mathcal{X}_\mathcal{S},\mathcal{Y}_\mathcal{P},\epsilon_1,\epsilon_2)$-ULDP*** *if it satisfies the following properties:*

*1) For any $y \in \mathcal{Y}_\mathcal{I}$, $\exists\ x,x' \in \mathcal{X}_\mathcal{N}$*

$$\mathbf{P}(y|x) > 0 \text{ and } \mathbf{P}(y|x') = 0, \forall x' \neq x. \tag{3.3}$$

*2) For any $x,x' \in \mathcal{X}$ and any $y \in \mathcal{Y}_\mathcal{P}$,*

$$\mathbf{P}(y|x) \leq e^{\epsilon_1}\mathbf{P}(y|x') \tag{3.4}$$

*That is, $\epsilon_1$-differential privacy is guaranteed for $\forall x \in \mathcal{X}$.*

*3) For any $x,x' \in \mathcal{X}_\mathcal{S}$ and any $y \in \mathcal{Y}_\mathcal{P}$,*

$$\mathbf{P}(y|x) \leq e^{\epsilon_2} \mathbf{P}(y|x') \tag{3.5}$$

*Where $\epsilon_2$-differential privacy is guaranteed for $\forall x \in \mathcal{X}_\mathcal{S}$ and $\epsilon_1 \geq \epsilon_2$.*

We then provide a concrete construction $(\mathcal{X}_\mathcal{S}, \epsilon_1, \epsilon_2)$-URR (or $(\mathcal{X}_\mathcal{S}, \epsilon_1, \epsilon_2)$-Utility-optimized Randomized Response) under it (see Definition 3.3). Figure 3.5 illustrates this protocol.

**Definition 3.3** ($(\mathcal{X}_\mathcal{S}, \epsilon_1, \epsilon_2)$-URR)**.** *Given $\mathcal{X}_\mathcal{S} \subseteq \mathcal{X}$ and $\epsilon_1, \epsilon_2 \in \mathbb{R}_{\geq 0}$, let $c_1 = \frac{e^{\epsilon_2}}{e^{\epsilon_2}+|\mathcal{X}_\mathcal{S}|-1}$, $c_2 = \frac{1}{e^{\epsilon_2}+|\mathcal{X}_\mathcal{S}|-1}$, $c_3 = \frac{1}{e^{\epsilon_1}+|\mathcal{X}_\mathcal{S}|-1}$, $c_4 = \frac{e^{\epsilon_1}-1}{e^{\epsilon_1}+|\mathcal{X}_\mathcal{S}|-1}$. Then the **$(\mathcal{X}_\mathcal{S}, \epsilon_1, \epsilon_2)$-URR** can be defined as:*

$$\mathbf{P}_{uRR}(y|x) = \begin{cases} c_1 & \text{if } x_i \in \mathcal{X}_\mathcal{S}, y = x \\ c_2 & \text{if } x_i \in \mathcal{X}_\mathcal{S}, y \in \mathcal{X}_\mathcal{S} \setminus \{x\} \\ c_3 & \text{if } x_i \in \mathcal{X}_\mathcal{N}, y \in \mathcal{X}_\mathcal{S} \\ c_4 & \text{if } x_i \in \mathcal{X}_\mathcal{N}, y = x \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

*where $c_4 \geq c_1 \geq c_2 \geq c_3$, $c_1 + (|\mathcal{X}_\mathcal{S}| - 1)c_2 = 1$, $c_4 + |\mathcal{X}_\mathcal{S}| c_3 = 1$.*

With these notations, we are able to prove via two different approaches that $(\mathcal{X}_\mathcal{S}, \epsilon_1, \epsilon_2)$-URR satisfies $(\mathcal{X}_\mathcal{S}, \mathcal{Y}_\mathcal{P}, \epsilon_1, \epsilon_2)$-ULDP. The proof is shown below.

*Proof.* ***First Approach.*** We can show how the properties in Definition 3.2 hold by showing all possible scenarios of input and output combinations satisfy the definition.

1) For any $y \in \mathcal{Y}_\mathcal{I}$, there $\exists x, x' \in \mathcal{X}_\mathcal{N}$

$$\mathbf{P}_{uRR}(y|x) = c_4 > 0 \text{ and } \mathbf{P}_{uRR}(y|x') = 0, \forall x' \neq x. \tag{3.7}$$

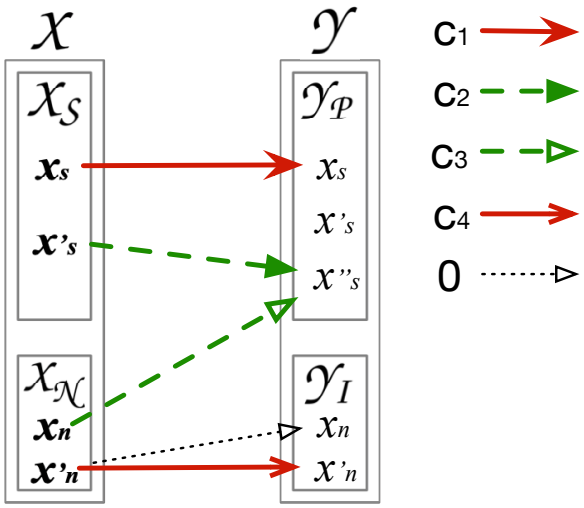



Figure 3.5: An illustration of how data are perturbed under $(\mathcal{X}_{\mathcal{S}},\epsilon_1,\epsilon_2)$-URR.

2) For any $x,x' \in \mathcal{X}_{\mathcal{N}}$ and any $y \in \mathcal{Y}_{\mathcal{P}}$,

$$\frac{\mathbf{P}_{uRR}(y|x)}{\mathbf{P}_{uRR}(y|x')} \leq \frac{c_4}{c_3} \leq e^{\epsilon_1} \tag{3.8}$$

3) For any $x,x' \in \mathcal{X}_{\mathcal{S}}$ and any $y \in \mathcal{Y}_{\mathcal{P}}$,

$$\frac{\mathbf{P}_{uRR}(y|x)}{\mathbf{P}_{uRR}(y|x')} \leq \frac{c_1}{c_2} = e^{\epsilon_2} \leq e^{\epsilon_1} \tag{3.9}$$

4) For any $x \in \mathcal{X}_{\mathcal{S}}$, $x' \in \mathcal{X}_{\mathcal{N}}$ and any $y \in \mathcal{Y}_{\mathcal{P}}$,

$$\frac{\mathbf{P}_{uRR}(y|x)}{\mathbf{P}_{uRR}(y|x')} \leq \frac{c_1}{c_3} = \frac{e^{\epsilon_2}(e^{\epsilon_1}+|\mathcal{X}_{\mathcal{S}}|-1)}{e^{\epsilon_2}+|\mathcal{X}_{\mathcal{S}}|-1} \leq e^{\epsilon_1} \tag{3.10}$$

□

*Proof.* ***Second Approach.*** We can demonstrate our protection mechanism as a two-layer model as well: all input data has $\epsilon_1$-differential privacy guaranteed while for sensitive data, another layer of $\epsilon_2$-differential privacy is provided.

– ***Layer 1.*** If we do not differentiate $\mathcal{X}_{\mathcal{S}}$ and $\mathcal{X}_{\mathcal{N}}$, then we have same conclusion as Equation 3.3 and 3.4.

1) For any $y \in \mathcal{Y}_{\mathcal{I}}$, there $\exists x,x' \in \mathcal{X}_{\mathcal{N}}$

$$\mathbf{P}_{uRR}(y|x) = c_4 \geq 0 \text{ and } \mathbf{P}_{uRR}(y|x') = 0, \forall x' \neq x. \tag{3.11}$$

2) For any $x,x' \in \mathcal{X}$ and any $y \in \mathcal{Y}_{\mathcal{P}}$,

$$\frac{\mathbf{P}_{uRR}(y|x)}{\mathbf{P}_{uRR}(y|x')} \leq \frac{c_1}{c_3} \leq e^{\epsilon_1} \tag{3.12}$$

Therefore, for all input data, $\epsilon_1$-differential privacy is guaranteed.

– ***Layer 2.*** According to our definition, $\mathcal{Y}_{\mathcal{P}} \subseteq \mathcal{X}_{\mathcal{S}}$, let $\mathcal{Z}_{\mathcal{P}}$ be the protected output set which follows same definition of $\mathcal{Y}_{\mathcal{P}}$. Thus we have:

3) For any $y,y' \in \mathcal{Y}_{\mathcal{P}} \subseteq \mathcal{X}_{\mathcal{S}}$ and any $z \in \mathcal{Z}_{\mathcal{P}}$

$$\frac{\mathbf{P}_{uRR}(z|y)}{\mathbf{P}_{uRR}(z|y')} \leq \frac{c_1}{c_2} = e^{\epsilon_2} \tag{3.13}$$

□

Our $(\mathcal{X}_{\mathcal{S}}, \mathcal{Y}_{\mathcal{P}}, \epsilon_1, \epsilon_2)$-ULDP notion is adapted from the $(\mathcal{X}_{\mathcal{S}}, \mathcal{Y}_{\mathcal{P}}, \epsilon)$-ULDP notion [174] by introducing an additional $\epsilon$ in order to provide stronger protection over the sensitive data entries (i.e., popular domains) while maintaining as much utility as possible for the non-sensitive data (e.g., unpopular domains). $(\mathcal{X}_{\mathcal{S}},\mathcal{Y}_{\mathcal{P}},\epsilon_1,\epsilon_2)$-ULDP provides $\epsilon_1$-DP and $\epsilon_2$-DP for different input, and it inherits other basic properties of ULDP [174], like *sequential composition* and *post-processing*.

**SensitiveSet.** As described in Section 3.3.1, we want to build the sensitive set and associate it with a different privacy budget. The sensitive set consists of popular domains with a high volume of visits. As implied by the results in Section 3.2.4, the repetitive queries of a user to popular domain names form his/her "identifier". Higher privacy protection for popular domain names is thus a reasonable deduction. In our experiment, the list is generated based on DNS sessions from another dataset of 9k users. When LDPResolve is installed by the users, the sensitive set can be fetched periodically from a web server, like Adblock fetching EasyList [12] The sensitive set can also be augmented with domains chosen by the user.

We assume the sensitive set is not a secret, so the adversary can obtain the set and actively prune the enlisted domains before user tracking. In Section 3.4.5, we discuss the impact of this strategy.

## 3.4 Evaluation of LDPResolve

We first describe the experiment settings for LDPResolve and three evaluation metrics. In Section 3.4.1, we investigate to what extent LDPResolve can curb DNS-based user tracking. In Section 3.4.2, the three parameters of LDPResolve are assessed. In Section 3.4.4 and Section 3.4.3, we discuss alternative settings of LDPResolve, In Section 3.4.5, we discuss the impact of the adaptive attack strategies. In Section 3.4.6, we compare against another relevant work that disperses DNS requests. In Section 3.4.7, we implement a prototype of LDPResolve and evaluate its overhead.

**Dataset.** We extract anonymized DNS query logs from a campus resolver as our evaluation dataset, which contain information such as hashed IPs, timestamps, domain names, qtypes, etc. We use 30,716 DNS sessions collected from 1,000 different users in a two-week period to evaluate both the open-world and the closed-world settings with a focus on the closed-world setting, as it favors the adversary more. The average, min, and max DNS sessions per user

are 30.7, 12, 90 respectively. For a DNS session, the longest one covers 96.3 hours, while the shortest one only issued 1 query. The median duration is 2.74 hours.

We choose 80% sessions of a user to fill **SL** and leave the remaining 20% for **SU**, which enhances the capability of the attacker.

To generate the sensitive set, we collect another DNS dataset with 272,078 sessions from 9,000 users. It has no overlap with **SL** and **SU**. We rank domains based on their frequencies in sessions and take the top $N_S$ domains as the sensitive set.

We simulate DNS queries under LDPResolve by perturbing their enclosed domains. We vary different parameters, including $\epsilon_1$, $\epsilon_2$ and $N_S$ to assess their impact. In the default setting, we only perturb sessions in **SU**, which represents the situation that the attacker has acquired a "clean" **SL** and tries to correlate it with a noisy **SU**. The campus wireless network assigns an IP for each device (though it is periodically changed based on DHCP), so NAT/VPN egress endpoint is not expected.

**Evaluation Metrics.** We consider three metrics to evaluate the data utility. The first is **tracking accuracy** (or `TrkAcc`), which is the same as the accuracy used to evaluate DSCorr. The goal of LDPResolve is to reduce it as much as possible.

The second is **standard deviation** (or `std`) of the domain frequency (used by prior works in malicious domain detection [29, 153, 247, 52]), measured by the session count. In addition to `std` across all domains, we also measure it on the ones in sensitive set and non-sensitive domains, and use `std_s` and `std_n` to represent their `std` respectively.

The third metric measures the utility at the session level. We measure the **change ratio** (or `ChgRatio`) of domain names and domain pairs after perturbation. If $O$ is original set of domain names (or domain pairs) and $P$ is perturbed set, `ChgRatio` is computed as $|O \cap P|/|O|$. For `ChgRatio` on a single domain name, sensitive domain ($s$) and non-sensitive ($n$) domain

are calculated separately. For `ChgRatio` on domain pairs, pair of two sensitive domains ($s$,$s$), one sensitive domain and one non-sensitive domain ($s$,$n$), and two non-sensitive domains ($n$,$n$) are measured. We choose `ChgRatio` because it impacts the mapping of a domain name to the source IP address or domain names to domain names, which is also utilized a lot for malicious domain detection [147, 51, 77, 194].

### 3.4.1 Impact on User Tracking

We first measure the impact of LDPResolve on all tracking methods (jac, cos, bay, ja-bi, co-bi, ba-bi, DSCorr) in the closed-world setting as described in [41]. We set $\epsilon_1 = 10$ and $\epsilon_2 = 2$ to represent high & low privacy budgets. We set $N_s$ to 2k and 10k out of more than 2 million domain names from the 9k-user set, to assess the impact of the sensitive set. When $N_s = 2000$, 72.6% domain names per session are sensitive, but 1.7% domains in **SU** are sensitive, showing a long-tail distribution. When $N_s = 10000$, the numbers are changed to 92.4% and 8.3% respectively.

Figure 3.6 shows `TrkAcc` before and after LDPResolve is applied[2]. It turns out DSCorr is influenced most: `TrkAcc` is dropped to 60.0% when $N_s$ is 2k, and **10.1%** when $N_s$ is 10k, from 93.0%. On the other hand, the impact to 1NN-Cosine (cos) is the the smallest among all tracking methods: `TrkAcc` is dropped to 62.2% and 34.1% from 86.6%, when $N_s$ is 2k and 10k respectively. The result indicates LDPResolve is effective in protecting users' privacy, and it has stronger influence on tracking methods with higher `TrkAcc`.

We also test LDPResolve in the open-world setting, by setting $N_s$ to 10k. The original `TrkAcc` for 1NN-Cosine and DSCorr are 70.9% and 82.3% respectively. Under LDPResolve, `TrkAcc` of them are dropped to 51.9% and 50.8%. In the open-world setting, one can achieve an accuracy of at least 50% by labeling all sessions as unknown. Thus, the `TrkAcc`

[2]Noticeably, bi-ba has higher `TrkAcc` comparing to DSCorr (95.7% vs 93.0%). This is because we use 80% and 20% data for training and testing here, varying the number of labeled sessions per user. DSCorr performs better with less data.

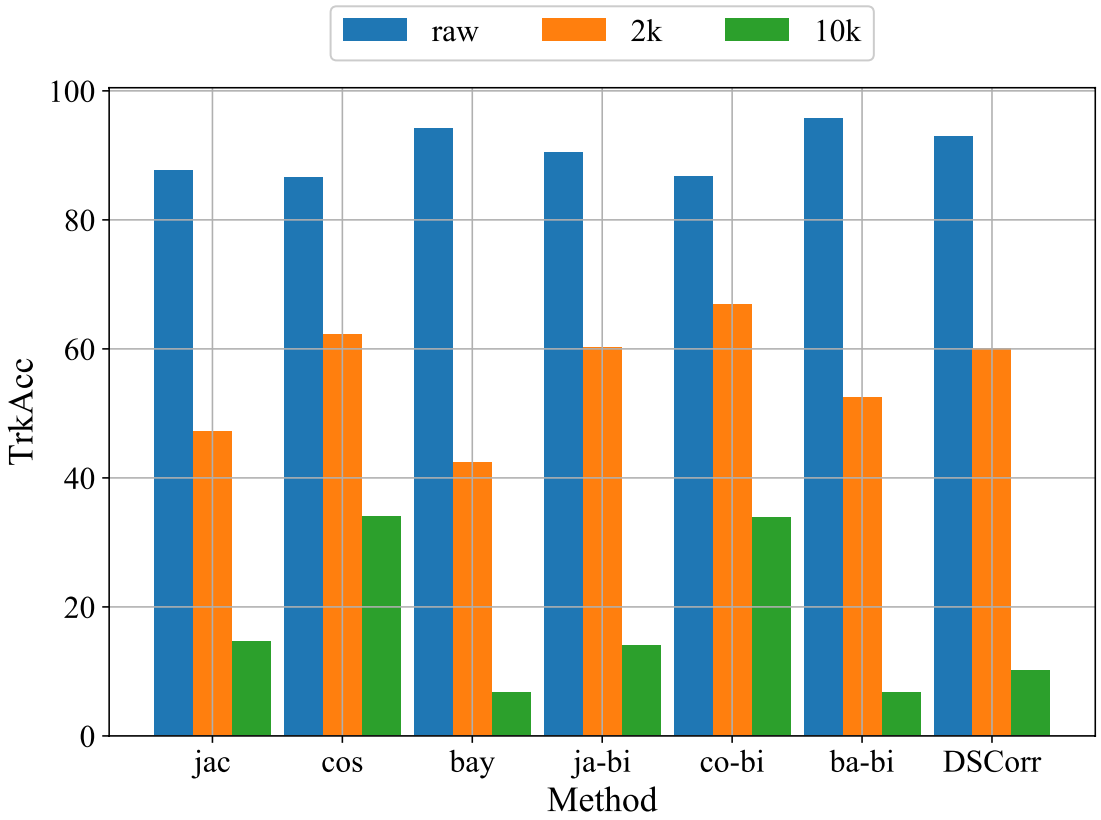



Figure 3.6: Comparison of `TrkAcc` before ("raw") and after deploying LDPResolve ("2k" and "10k"). "2k" and "10k" are values set to $N_s$. All numbers are percentage.

drop here is sufficient enough to show the effectiveness of LDPResolve.

For the follow-up experiments, we choose 1NN-Cosine (uni-gram) as the tracking method since it is more robust to noise, and focus on the closed-world setting.

### 3.4.2 Impact of Parameters

We discuss the impact of different parameters $(\epsilon_1, \epsilon_2, N_s)$ in LDPResolve on the three evaluation metrics here. We set a large $\epsilon_1$ (10) and a small $\epsilon_2$ (2) initially because we intend to preserve more privacy for sensitive domains (more noise added) while maintaining better utility for other domains (less noise added). Theoretically, $e^{\epsilon_1}$ needs to be at least the same order as the size of the sensitive set $N_s$ to avoid significant changes to non-sensitive domains.

To evaluate the impact of each parameter, we fix other parameters and vary the tested parameter and obtain privacy and utility results.

We examine $\epsilon_1$ from 2 to 15 and $\epsilon_2$ from 0.5 to 10 with different size of the sensitive set $N_s$. We evaluate the impact of these parameters based on their `TrkAcc`, `std` and `ChgRatio`. Overall, a large $\epsilon_1$ and $N_s$ with small $\epsilon_2$ is preferred. More specifically, by setting $\epsilon_1 = 10$, $\epsilon_2 = 2$ and $N_s = 20{,}000$, LDPResolve is able to decrease the tracking accuracy to 23.3% for 1NN-Cosine, which is proved to be the method most robust to noise. Regarding the utility measured by `std`, our result indicates they can be preserved (especially for non-sensitive domains). For instance, when $\epsilon_2 = 2$ and $\epsilon_1 = 10$, `std_n` is only 5.71.

**Impact of $\epsilon_1$.** $\epsilon_1$ is the privacy budget for the whole domain set and the smaller $\epsilon_1$ will introduce greater noise to all the domains. We tested 5 different $\epsilon_1$ ranging from 2 to 15 while setting $\epsilon_2$ and $N_S$ to 2 and 10000 respectively. The result of `TrkAcc` and `std` are shown in Table 3.1. We see that as $\epsilon_1$ drops, `TrkAcc` drops drastically due to the higher-level noise added. Besides, `std` of the whole domain set increases slowly with `std` of sensitive domains remaining almost the same and `std` for non-sensitive domains grows, because we use $\epsilon_2$ to control the changes on the sensitive domains.

Figure 3.7a supports this claim as well with the result on `ChgRatio`. The co-occurrence of any sets of domains involving non-sensitive domains is dropping sharply from a very high level as $\epsilon_1$ gets smaller. Meanwhile, 99.8% of non-sensitive domain names and 99.5% non-sensitive domain pairs are unchanged when $\epsilon_1$ is set to 15. Only less than 16.8% of non-sensitive domains and 1.3% non-sensitive domain pairs remain the same after the $\epsilon_1$ is decreased to 7. Since non-sensitive domains play a big role in security research, $\epsilon_1$ should be set to a relatively high value in order to guarantee reasonable utility.

**Impact of $\epsilon_2$.** $\epsilon_2$ is the privacy budget for sensitive domains only, and the smaller $\epsilon_2$ introduces more noise. Similarly to the last setting, we fix $\epsilon_1$ to 10, $N_S$ to 10000, and examine

| $\epsilon_1$ | 15 | 10 | 9 | 8 | 7 | 6 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|
| TrkAcc | 38.7 | 34.1 | 28.4 | 19.5 | 10.2 | 3.7 | 1.4 | 0.2 |
| std | 332.30 | 343.66 | 352.52 | 360.62 | 365.38 | 367.61 | 368.45 | 369.12 |
| std_s | 1279.53 | 1279.63 | 1279.94 | 1280.39 | 1279.76 | 1279.92 | 1279.01 | 1280.31 |
| std_n | 3.48 | 5.71 | 6.85 | 8.54 | 10.66 | 10.75 | 11.20 | 10.84 |

Table 3.1: Impact of $\epsilon_1$ on TrkAcc (shown in percentage) and overall std. std_s and std_n are std for sensitive and non-sensitive domains.

| $\epsilon_2$ | 10 | 8 | 7 | 6 | 5 | 2 | 0.5 |
|---|---|---|---|---|---|---|---|
| TrkAcc | 84.8 | 80.2 | 70.3 | 57.4 | 43.6 | 34.1 | 33.9 |
| std | 121.59 | 264.24 | 305.47 | 326.82 | 336.81 | 343.66 | 343.95 |
| std_s | 241.10 | 731.94 | 967.65 | 1127.27 | 1214.65 | 1279.63 | 1282.55 |
| std_n | 3.27 | 3.80 | 5.31 | 5.52 | 5.67 | 5.71 | 4.38 |

Table 3.2: Impact of $\epsilon_2$ on TrkAcc, std, std_s and std_n.

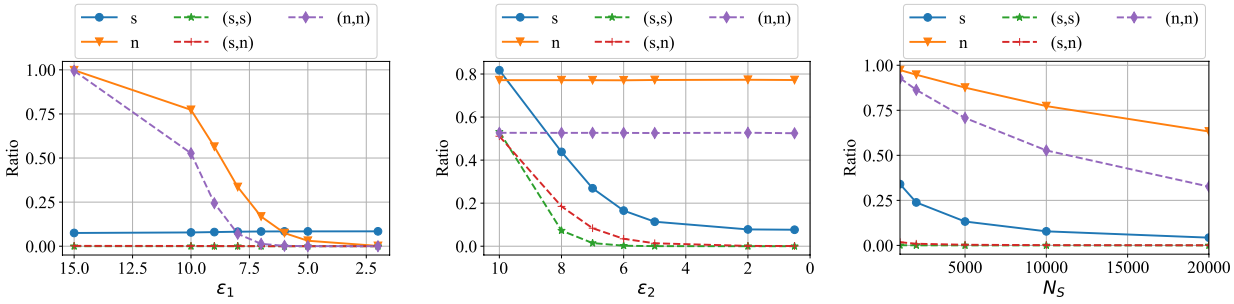
| $N_S$ | 1000 | 2000 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| TrkAcc | 68.0 | 62.2 | 48.8 | 34.1 | 23.3 |
| std | 363.13 | 388.23 | 376.73 | 343.66 | 304.17 |
| std_s | 2552.22 | 2205.18 | 1669.81 | 1279.63 | 949.84 |
| std_n | 1.72 | 2.15 | 6.54 | 5.71 | 7.13 |

Table 3.3: Impact of $N_S$ on TrkAcc, std, std_s and std_n.

the impact of different $\epsilon_2$ from 0.5 to 10. In this setting, we have the non-sensitive domains not impacted so their utility to legitimate applications is preserved. But because tracking relies on sensitive domains, it is disrupted.

As shown in Table 3.2, std of non-sensitive domains is small (all less than 6) and stable across different $\epsilon_2$. A small difference is observed because a different set of non-sensitive domains is perturbed every time we run the experiment. The same pattern can be found in Figure 3.7b where the ChgRatio of domains or domain pairs without involving sensitive domains remains almost unchanged under the fluctuation of $\epsilon_2$.

In conclusion, higher $\epsilon_1$ and lower $\epsilon_2$ are preferred for a good balance between privacy and data utility.




(a) `ChgRatio` vs. $\epsilon_1$ (b) `ChgRatio` vs. $\epsilon_2$ (c) `ChgRatio` vs. $N_S$

Figure 3.7: `ChgRatio` vs. $\epsilon_1$, $\epsilon_2$ and $N_S$. $s$, $n$, $(s,s)$, $(s,n)$ and $(n,n)$ are explained in "Evaluation metrics".

**Impact of $N_S$.** We fix $\epsilon_1$ to 10, $\epsilon_2$ to 2 and change $N_S$ from 1k to 20k. Table 3.3 shows that by increasing $N_S$, user tracking is severely interfered, with `TrkAcc` dropping from 68.0% to 23.3%. In the meantime, its impact on non-sensitive domains is controlled, with `std` of them ranging from 1.72 to 7.13. Figure 3.7c shows that with the increase of $N_S$, `ChgRatio` of sensitive domains, nonsensitive domains and pairs of nonsensitive domains are decreasing.

Here we explain this observation in depth. Followed by increase of $N_s$, 4 perturbation probabilities $c_1, c_2, c_3, c_4$ all decrease. Sensitive domains will have a greater chance to be changed as they are associated with $c_1, c_2, c_3$, while non-sensitive ones are less changed as it is only impacted by $c_4$. With $N_s$ increased, sensitive set is expanded to contain more low-frequency domains, so `std_n` also increases. For pairs associated with sensitive domains, the perturbation breaks their relations, so the `ChgRatio` of them remain low with a trend of decreasing.

From the result and explanation above, it is clear that $N_s$ is also an essential parameter for LDPResolve. Larger $N_s$ is recommended when the legitimate applications highly rely on the non-sensitive domains. Though due to the power-law distribution of domains, 1k and 20k have a big difference of influence on user sessions, only a small portion of the whole domain set is impacted, as shown in Section 3.4.1.

Our default setting assumes **SL** is clean to the adversary. We further explore the scenario

when **SL** has noises injected by LDPResolve. The parameters are the same as our default setting: $\epsilon_1 = 10, \epsilon_2 = 2$, $N_s = 10000$.

Firstly, we assume **SL** is noisy while **SU** is clean. The tracking accuracy turns out to be 23.4%, which is even worse than 34.1% when we assume a clean **SL** and a noisy **SU**. Therefore, such "data poisoning attack" is even more effective against user tracking. Secondly, we allow one clean session for each user to be included in **SL** (**SU** is still clean). Tracking accuracy will be increased to 41.9%. The result suggests even one clean session in **SL** can give adversary great lift in countering LDPResolve. Finally, if both **SL** and **SU** are noisy, the accuracy will drop back to 32.5%.

### 3.4.3 Sensitive Set with SLDs

So far we fill the the sensitive set with FQDNs (Full Qualified Domain Names). It can also be constructed by extracting the SLDs (Second-Level Domains) part from the popular FQDNs, by leveraging public suffix list [4]. Then, if the SLD of a domain name matches the sensitive set, it will be considered as in $\mathcal{X}_\mathcal{S}$. By doing so, the sub-domains under sensitive domains are also protected.

By building the new sensitive set based on only 100 most popular SLDs from the same 9k-user dataset, tracking accuracy is dropped to 14.16%. This result shows that by adding strong noises to a small set of SLDs, tracking will be significantly disturbed.

When applying this change, certain domains need to be excluded, i.e., not adding their SLDs to sensitive set. One example is domains requested under `PTR` Record. Because *all* `PTR` records are under the same SLD `in-addr.arpa` or `ip6.arpa`, if including those domains in sensitive set, security research based on `PTR` records [185] will be significantly impaired.

### 3.4.4 Noisy SL

Our default setting assumes **SL** is clean to the adversary. We further explore the scenario when **SL** has noises injected by LDPResolve. The parameters are the same as our default setting: $\epsilon_1 = 10, \epsilon_2 = 2$, $N_s = 10000$.

Firstly, we assume **SL** is noisy while **SU** is clean. The tracking accuracy turns out to be 23.4%, which is even worse than 34.1% when we assume a clean **SL** and a noisy **SU**. Therefore, such "data poisoning attack" is even more effective against user tracking. Secondly, we allow one clean session for each user to be included in **SL** (**SU** is still clean). Tracking accuracy will be increased to 41.9%. The result suggests even one clean session in **SL** can give adversary great lift in countering LDPResolve. Finally, if both **SL** and **SU** are noisy, the accuracy will drop back to 32.5%.

### 3.4.5 Adaptive Tracking against LDPResolve

To counter LDPResolve, an adaptive attacker can try to eliminate the noises introduced by different means. As ULDP ensures domains in the non-sensitive set are invertible, one feasible option is to remove the observed domains that appear in the sensitive set, therefore reduce the effect of change of domains. In this way, the queries left in the records contain only authentic domains. Another option to eliminate the effect of LDPResolve is to estimate the domain frequency by *reversing* $(\mathcal{X}_{\mathcal{S}}, \epsilon_1, \epsilon_2)$-URR.

**Removing Sensitive Domains.** By removing the sensitive domains in the DNS records, most of the noises would be removed along with them. On the other hand, the tracking effectiveness should not be restored to the level without LDPResolve, since there are less domains to be used to connect sessions of the same user.

It turns out that tracking accuracy rises from 34.06% to 53.08% after this adaptive strategy.

For the vanilla setting (attacker has access to the clean data in both **SL** and **SU** and no domains are removed), the accuracy is 86.6%. As such, we argue that even this strategy is applied, the tracking accuracy is far from optimum for the adversary.

**Reversing $(\mathcal{X}_{\mathcal{S}},\epsilon_1,\epsilon_2)$-URR.** LDPResolve uses $(\mathcal{X}_{\mathcal{S}},\epsilon_1,\epsilon_2)$-URR to perturb a request. The process depends on a few parameters ($c_1$, $c_2$, $c_3$, $c_4$). When they are known to the adversary, she might attempt to reverse the perturbation process to estimate the real distribution of a domain based on the observed distribution. We propose an implementation for this strategy and evaluate its impact on LDPResolve. Mathematical details are listed below.

Suppose $\left|\overline{X}\right|$ is the total number of observed queries, $\left|\hat{X}\right|$ the estimate and $\left|\overline{X}^N\right|$ is the number of non-sensitive queries being observed. Note that $\left|\overline{X}\right| = \left|\hat{X}\right|$ under LDPResolve. $\left|\hat{X}^N\right|$ is the estimate of the real non-sensitive queries. According to the definition of $(\mathcal{X}_{\mathcal{S}},\epsilon_1,\epsilon_2)$-URR, we have:

$$\left|\hat{X}^N\right| = \frac{1}{c_4}\left|\overline{X}^N\right| \tag{3.14}$$

Therefore, the estimate of real sensitive queries $\left|\hat{X^S}\right|$ would be:

$$\left|\hat{X}^S\right| = |X| - \left|\hat{X}^N\right| = |X| - \frac{1}{c_4}\left|\overline{X}^N\right| \tag{3.15}$$

For a specific observed sensitive query $\overline{x}_0^S$, the total number of $\overline{x}_0^S$ follows the equation below:

$$
\begin{aligned}
\left|\overline{x}_0^S\right| &= \frac{c_1}{c_1+c_2+c_3}\left|\hat{x}_0^S\right| + \frac{c_2}{c_1+c_2+c_3}\sum_{k\neq 0}\left|\hat{x}_k^S\right|\frac{1}{|\mathcal{X}_\mathcal{S}|} \\
&+ \frac{c_3}{c_1+c_2+c_3}\left|\hat{X}^N\right|\frac{1}{|\mathcal{X}_\mathcal{S}|} \\
&= \frac{c_1}{c_1+c_2+c_3}\left|\hat{x}_0^S\right| + \frac{c_2}{|\mathcal{X}_\mathcal{S}|(c_1+c_2+c_3)}(\left|\hat{X}^S\right| - \left|\hat{x}_0^S\right|) \\
&+ \frac{c_3}{c_1+c_2+c_3}\left|\hat{X}^N\right|\frac{1}{|\mathcal{X}_\mathcal{S}|} \\
&= \frac{c_1}{c_1+c_2+c_3}\left|\hat{x}_0^S\right| + \frac{c_2}{|\mathcal{X}_\mathcal{S}|(c_1+c_2+c_3)}(\left|\overline{X}\right| \\
&- \frac{1}{c_4}\left|\overline{X}^N\right| - \left|\hat{x}_0^S\right|) + \frac{c_3}{c_4(c_1+c_2+c_3)|\mathcal{X}_\mathcal{S}|}\left|\overline{X}^N\right|
\end{aligned}
\tag{3.16}
$$

Therefore, we have the estimation of any observed domains as follows:

$$
\left|\hat{x}_0^S\right| = \frac{\frac{c_2-c_3}{c_4} - c_2\left|\overline{X}\right| + \left|\overline{x}_0^S\right||\mathcal{X}_\mathcal{S}|(c_1+c_2+c_3)}{(c_1-c_2)|\mathcal{X}_\mathcal{S}|} \tag{3.17}
$$

Attacker will then use $|\hat{x}_0^S|$ for user tracking. It turns out this strategy does not work well when the sensitive set is large, which introduces large randomness to the perturbation process. When $N_s$ is 10k, `TrkAcc` is slightly increased from 34.06% to 34.54%. As an alternative solution, the adversary could choose to reverse the non-sensitive set only, which are derived by excluding sensitive set from the entire domain set. This strategy increases `TrkAcc` to 53.50%, but still far from the vanilla setting when LDPResolve is not deployed (86.6%).

### 3.4.6 Comparison with K-resolver

As described in Section 3.3.1, K-resolver [104] is expected to deter tracking by dispersing DNS queries across resolvers. We test K-resolver by splitting DNS requests into $k$ slices, and launch 1NN-Cosine tracking against it. The detailed evaluation results are shown below.

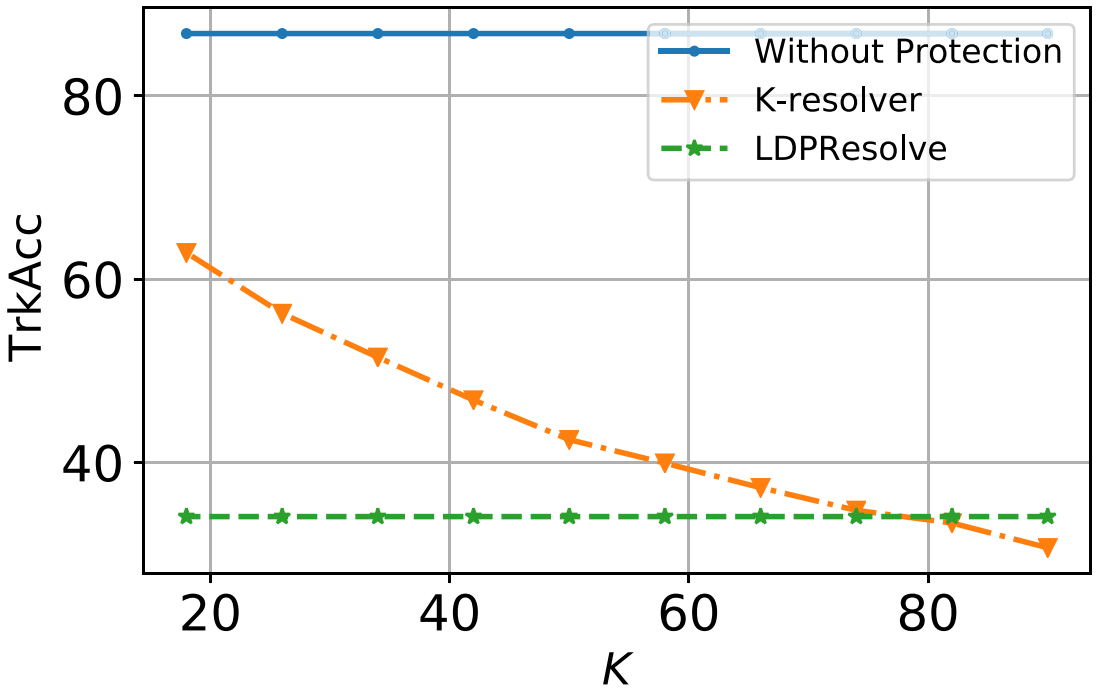



Figure 3.8: Comparison on `TrkAcc` of LDPResolve and K-resolver by $k$.

We vary $k$ from 18 to 90 (with step size 8) and `TrkAcc` is reduced from 62.89% to 30.7%, as shown in Figure 3.8. It turns out only when $k$ is very large (over 74), K-resolver can outperform LDPResolve (`TrkAcc` at 34.06%) in theory. However, finding such a big pool of resolvers is quite difficult. In fact, Hoang et al. investigated 53 DNS-over-HTTPs resolvers and found only 26 of them can provide reliable services. LDPResolve offers sufficient protection by involving much fewer resolvers (only 2). In fact, as a deterministic hash function is used by K-resolver, a group of domains will always be sent to the same resolver, regardless of how much information they leak. The randomized protocol of LDPResolve addresses this limitation.

### 3.4.7 Prototype

We implement a prototype of LDPResolve to evaluate its overhead. Our prototype is built on top of dnsdist[1], an open-source DNS load-balancer. We write Lua, Python, and Shell

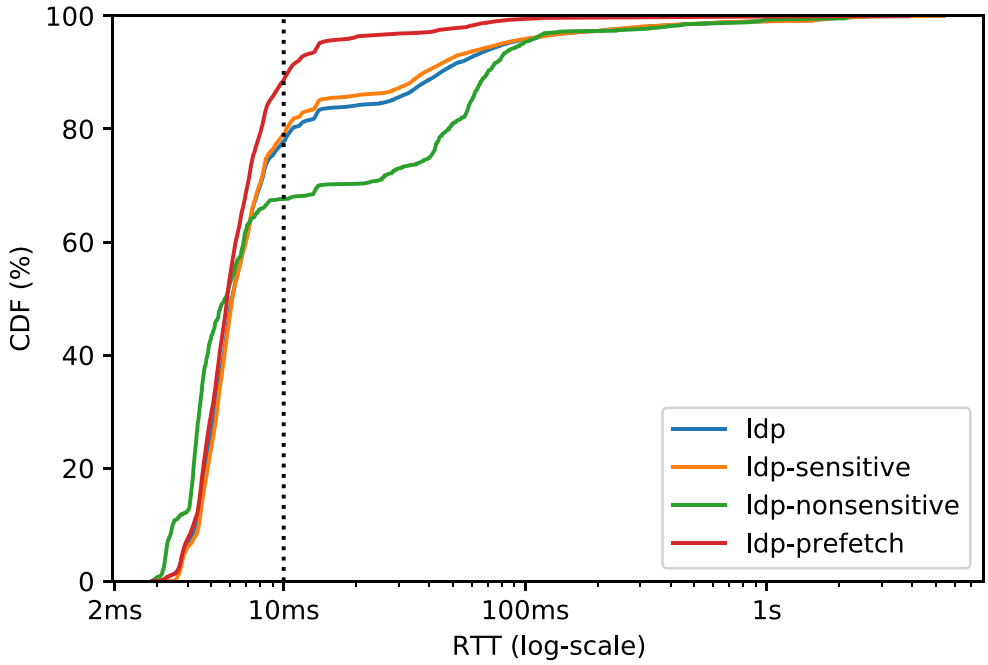



Figure 3.9: Comparison of `RTT` between different settings of LDPResolve. "ldp" are all queries. "ldp-sensitive" and "ldp-nonsensitive" are queries to domains in and not in the sensitive list. "ldp-prefetch" is `RTT` of all queries when prefetch of sensitive domains is enabled.

scripts to customize dnsdist to support LDPResolve. The code we wrote is released on GitHub.

For domains that are unperturbed, we query them through a primary resolver, which is a public resolver (`223.5.5.5`) in our experiment. For domains that are perturbed, two kinds of queries are issues: 1) queries sent to the alternative resolver (`AltRR`), which is set to be a trusted local recursive resolver running PowerDNS [6] and directly talking to authoritative nameservers, and 2) dummy queries (i.e., noise) sent to the primary resolver. The two types of queries are issued in separate processes, so the impact on the normal DNS resolution is confined. It is also worth mentioning that we create a local cache with a fixed size similar to the sensitive set.

We randomly choose a group of sessions of one user in our dataset, replay the first 10k DNS queries (`qname`/`qtype` combination) and evaluate the Round-trip Time (RTT) and traffic volume.

**RTT.** Firstly, we measure the distribution of the query RTTs under LDPResolve. Figure 3.9 shows the CDF plot of the 10K queries. Because of the local cache, RTT for most of the queries is less than 10ms (querying the primary resolver takes about 20ms). Responses to the non-sensitive domain names are slower than the sensitive ones, because they are less likely to be requested multiple times (so cached). We also examined the idea of prefetching the domain names in the sensitive set, and the RTT turns out to be even smaller generally. Overall, our result shows LDPResolve is efficient under parallel domain resolving.

**Traffic Volume.** We found the 10k queries generate about 2.76MB of DNS traffic to the public resolver. When using LDPResolve, 3.28MB DNS traffic is generated, where 2.36MB goes to the primary resolver, and 0.92MB goes to `AltRR`, which is 18.8% higher than querying DNS normally. The traffic sent to the primary resolver is actually reduced because of the local cache.

## 3.5 Discussion

**Limitations of LDPResolve.** 1) The utility could be worse for popular domains due to ULDP, which can hurt the applications like domain popularity ranking at recursive resolvers. From the level of authoritative nameservers, the impact is expected to be smaller, as queries from RRs over the world are aggregated. 2) We consider the passive attacker who only does traffic analysis and do not consider the active attacker who is able to change the state of the user's device, like changing DNS cache [133]. It would be an interesting study about whether or how an active attacker would be affected under LDPResolve. 3) According to our implemented prototype of LDPResolve, the overhead of traffic volume is moderate

and the latency can even be reduced when `AltRR` uses local resolver, local cache, or prefetch. These options might not always be available to the users, so the performance result should be perceived with a grain of salt.

**Sensitive Set.** As the sensitive set is generated based on the historical domain popularity, an attacker can manipulate this set by crafting domain visits. There exist two strategies: 1) injecting a malicious domain into the sensitive set so LDPResolve will perturb it at higher probabilities, but it also makes a user less likely to visit it; 2) injecting many irrelevant domains to "kick out" the real popular domains, but to override the query volume from users requires significant network bandwidth. As such, there is no strong motivation for an attacker to manipulate the sensitive set.

LDPResolve issues the same sensitive set to all users. An alternative approach is to let a user customize it for better performance at her end. We can allow a user to manually change or automatically generate it from her visiting history.

**Limitations of Dataset.** For evaluation, we use only one DNS dataset. We acknowledge that using more datasets can address the potential bias caused by the user population, but getting access to one, especially with session labels, is very challenging. Nevertheless, we want to emphasize that our DNS dataset contains 10,000 users launching over 300,000 DNS sessions, targeting 2 million domain names, which is sufficient for a large-scale study.

**Alternative Defense Approach.** To defend against DNS-based tracking, in addition to LDPResolve, other approaches include spreading DNS requests to multiple resolvers [104] and using an "oblivious DNS" which obfuscates the queries from clients [205]. However, none of those approaches considered the data utility for legitimate purposes like malicious domains detection. LDPResolve makes the first attempt to balance the privacy and utility of DNS data. Our adversary model assumes the recursive resolver is not trusted. An alternative setting is that recursive resolver performs privacy protection before sharing the data with

another party, e.g., Passive DNS [11, 9]. Noises can be added by the recursive resolver following a differential privacy notion and better data utility is supposed to be achieved. We leave this setting as a future work.

# Chapter 4

# DPRA: Differentially Private Resource Allocation

## 4.1 Introduction

Resource allocation (RA) is a long-standing problem relevant to various application scenarios, such as virtual machine assignment [164], storage allocation [148], network bandwidth management [162], and channel allocation [228]. Prior works mostly focus on the efficiency and cost of RA [123, 27, 90, 137, 116, 111, 175], e.g., how to improve resource utilization and guarantee the quality of service to all users [111]. However, the privacy issues of RA have been overlooked for a long time and were only studied recently. Angel et al. [20] reveal that a powerful attacker can determine the existence of other parties in the RA system. Specifically, for an allocator managing limited resources, when one party requests resources, the number of resources the other parties can obtain will be affected. Therefore, the attacker can try to send a large volume of requests and use the allocation results to infer the existence of other users. Knowing the existence of others opens the door to more serious attacks that can infer users' activities. For example, although Metadata-private messengers (MPM) are designed

to hide the calling activities between clients, such privacy guarantees can be breached with RA side-channel and traffic analysis [20].

**Existing Resource Allocators.** Most of existing allocators (e.g., the first-in-first-out allocator) do not offer any privacy guarantee [19]. Recently, Angel et al. [19] proposed an allocator AKR[1] that satisfies differential privacy (DP) [61]. Angel et al. consider the scenario where the resource allocator owns a limited number of resources and the attacker controls a large number of clients. The attacker learns of the existence of another victim when the requests to the allocator are not fulfilled. To protect privacy during RA, AKR adds dummy requests to the real ones and then assigns resources to randomly chosen requests. The number of dummy requests follows the *biased Laplace distribution*, and by a standard *post-processing argument* in DP (explained in Section 4.2.3), the existence of the victim is differentially private to the attacker. While the dummy requests puzzle the attacker, we found that the utility of AKR is not satisfactory. For instance, to achieve an acceptable level of DP (with parameters $\epsilon = 2, \delta = 10^{-6}$) *more than 40% of the resources must be wasted* in its experiment setting.

**Our Solution.** Different from AKR, which implies the attacker knows the total number of requests after noise is added, we observe that the practical attacker only has a *partial* view of RA. Therefore we choose to model the RA privacy from the attacker's view. Due to the randomness introduced by RA, we benefit from *"privacy amplification"* [23, 67] through such modeling and achieve better privacy-utility tradeoff.

Then, we implemented the DP mechanisms under four noise distributions, including constant (CST), uniform (UNI), one-sided geometric (GEO), and double geometric (DGEO), and tailored them to our new modeling. We conduct a rigorous privacy analysis and derive *much tighter privacy bounds* than AKR. We prove GEO and DGEO always satisfy $\epsilon$-DP under various parameters, while CST and UNI satisfy $\epsilon$-DP under certain conditions. Interestingly,

[1]The first letter of the authors' names.

we find that adding a constant noise (CST), which obviously violates traditional DP, can be proven to satisfy DP in the context of RA, due to the randomness of the allocation process. On the other hand, AKR only considers *non-negative* Laplace noise and relies on the post-processing argument to satisfy ($\epsilon$, $\delta$)-DP.

**Evaluation.** We evaluate the proposed mechanisms empirically by simulating the RA process of Alpenhorn [145] with *5 million to 100 million* rounds of requests, to demonstrate the privacy-utility tradeoff in real-world settings. (1) GEO outperforms other mechanisms when $\epsilon$ is smaller (i.e., $\epsilon < 2$) and has relatively stable performance; (2) DGEO performs better with a larger $\epsilon$ ($\epsilon > 2$). Compared to AKR which wastes 44% of the resources, DGEO only wastes 10% of resources with $\epsilon = 2$. Moreover, when $\epsilon = 2.25$, AKR utilizes 60% of the resources while DGEO achieves 97% utilization. (3) Parameters of the mechanisms have to be carefully tuned and negative bias should be avoided. The advantage over AKR is especially surprising as AKR is supposed to have better utility under the relaxed ($\epsilon$, $\delta$)-DP, whereas our mechanisms follow the strict $\epsilon$-DP. This justifies the effectiveness of our privacy analysis.

**Contributions.** The main contributions are summarized below:

- We conduct a rigorous privacy analysis of differentially private RA, and derive tighter privacy bounds under the attacker's view for four noisy mechanisms.
- We theoretically and empirically evaluate our proposed mechanisms. One mechanism, called GEO, leads to the best privacy-utility tradeoff and outperforms AKR by a large margin.
- We published the code in a GitHub repository [47].

## 4.2 Background

### 4.2.1 Problem Definition

Resource allocation (RA) assigns limited resources to the requesting parties, and we focus on RA within computing systems in this paper. Examples include resource management in data centers [18], assignment of virtual machines (VMs) in cloud [164], cache allocation in computers [148], and channel allocation for Metadata-private Messenger (MPM) [145]. Below we first provide an abstract view of standard RA and describe its involved parties and procedure. Then, we describe the attackers' goals and capabilities in RA. The frequently used notations are defined in Table 4.1.

| Notation | Description |
|---|---|
| $D$,$D'$ | Neighboring datasets differing in one victim |
| $k$ | Number of available resources |
| $m$ | Number of compromised clients |
| $d$ | Number of noisy requests (can be negative) |
| $y$ | Number of resources dispatched to attacker |
| $x_\ell$,$x_r$,$p$,$s$,$\mu$ | Parameters of the noisy mechanisms |

Table 4.1: Notations frequently used in this paper.

**RA Parties and Procedure.** Our abstraction of standard RA considers a scenario where an *allocator* allocates *resources* based on the *requests* submitted by a number of *clients*. The allocator can contain one server or a group of servers for fault-tolerance. In the setting of data center, the allocator can be a virtual machine manager (VMM), and the client can be a data center tenant. In the setting of MPM, where two users can set up a call in a private way, the allocator can be a callee and the client can be a caller. Regarding the RA procedure, we assume it takes rounds of interactions between the allocator and the clients. In each round, the allocator receives requests from its clients for resources (e.g., CPUs in a cloud and communication channels to be allocated to a caller in MPM) and makes the best efforts to serve the requests. Hence, for each request, the allocator either accepts it and

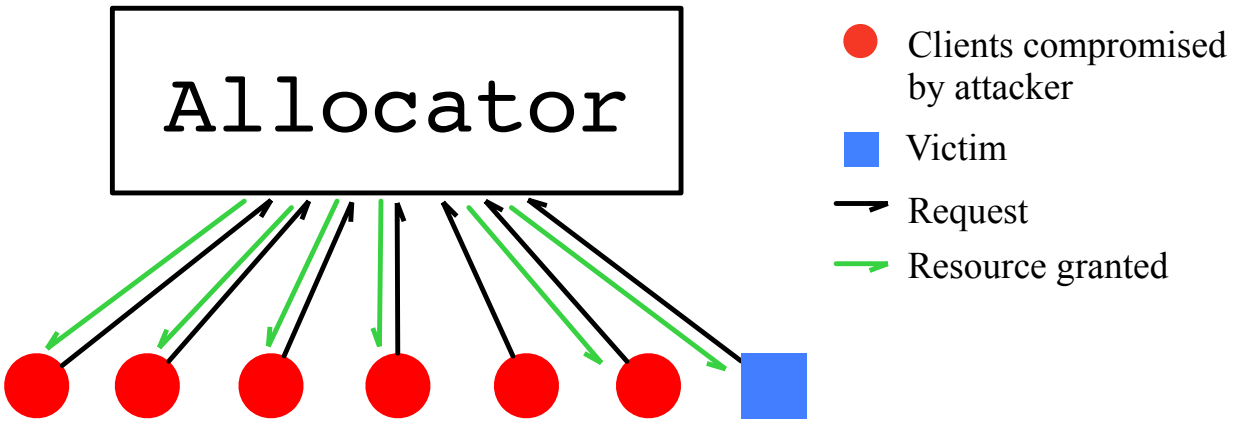



Figure 4.1: An example of RA. An *allocator* has six resources and the total number of requests sent by attacker is six. Privacy of the victim is violated when the attacker observes one of the requests is not fulfilled.

allocates the resources, or rejects it when all resources have been occupied.

Following prior work [19], we assume the quantity of the resources is a limited number $k$, and all resources are identical. Each round, some clients send requests, and each request asks for one piece of resource. Because the resources are identical, the requests are also identical (except the requesters' IDs). We note that some assumptions can be relaxed (e.g., resources are not identical and each client can request multiple resources) to match different application scenarios, and we discuss these variations in Section 4.6.3.

**Adversary Model.** Since the clients' requests might not always be fulfilled under limited resources, the allocator's response could leak information about the existence of some clients. Figure 4.1 illustrates how such inference attack can be conducted. Formally, we assume the attacker in the strongest attack scenario who can:

- *compromise all clients except one victim client*, and we denote the number of compromised clients as $m$.
- know the number of available resources $k$ before RA.
- compromise more clients than the resources, i.e., $m \geq k$, and all requests are submitted

at the same time.

The attacker can tell there is a victim requesting a resource if less than $k$ requests from the attacker are fulfilled.

We assume the adversary is malicious who can behave arbitrarily rather than being semi-honest. We only consider the privacy issues in RA and other issues like availability (e.g., the attacker blocks a victim from getting resources by overwhelming the allocator) are out of scope. We note that an adaptive attacker can exploit the correlation of results between multiple rounds, and infer more information that weakens the allocation privacy. We propose a few approaches to tackle such adversary in Section 4.6.2.

Regarding the allocator, we assume it is trustworthy, and can see all clients and requests and add noises. Hence, the allocator can analyze the historical data to estimate the parameters to be used by our mechanisms without privacy issues. We also assume the communication between the victim and the allocator is secure, so the number of the victim's requests are not leaked.

**Impact of RA Side-channel Leakage.** Even though the information about the victim during RA is seemingly insignificant, it can be leveraged as a side channel to break privacy-enhancing technologies or make the subsequent attacks more effective.

Specifically, Angel et al. described an attack based on the RA side-channel [20] against MPM. MPM like Vuvuzela [228], Alpenhorn [145], Stadium [226] and Karaoke [144] hide both the message content and its metadata (including sender, receiver, time of communication, etc.) from the network adversaries. In essence, a user within an MPM initiates a conversation with her friend on an agreed time or round and encrypts the messages with a shared key. In the conversation round, the user initiates $k$ channels to $k$ friends (including the friend to have the "real" conversation). To avoid leaking metadata, users are forced to send and

receive a message on each channel in each round [2]. Since MPM requires the clients to always be online, only the communicating parties of a client should be protected, while the client's existence is known.

It turns out the privacy guarantee of MPM can be entirely violated. As shown by Angel et al. [20], a user usually has a greater number of friends than $k$ channels. When the attacker controls $m$ ($m \geq k$) friends of the user and lets them call the user, if the user is busy (e.g., not responding) to more than $m-k$ callers controlled by the attacker, the attacker knows the user is communicating with others who are out of her control. Moreover, when the attacker compromises the friends of multiple users, she can infer which users are likely active in a given round with intersection and disclosure attacks [16, 163]. Specifically, the attacker can narrow down the possible sender-recipient pairs by ignoring all the idle users during the first round of calling. Then the attacker can build intersections of active users and keep reducing the set of possible sender-recipient pairs during additional rounds. Because the requests and resources are all identical under our assumptions, detecting such inference attack is also very challenging.

**Existing Resource Allocators.** We aim to design an RA that *hides the existence of the victim* while *maximizing request fulfillment*. One trivial solution that provides perfect privacy is to have the allocator withhold all the resources and reject every request, but obviously, this solution has zero utility. Angel et al. characterizes the existing allocators into (1) FIFO (first in, first out) allocator, (2) Uniform allocator, (3) Slot-based resource allocator (SRA) and (4) Randomized resource allocator (RRA) [19], while FIFO and uniform allocators are non-private and SRA and RRA are private. However, both SRA and RRA incur prominent utility loss.

[2] MPM is different from the normal messenger apps in that it can decline legitimate calls to provide metadata privacy. Yet, given that each conversation round has very small latency (e.g., measured in micro-seconds in the context of Alpenhorn [19]), the impact of call declining on user experience remains moderate.

### 4.2.2 Differential Privacy

Our work applies differential privacy (DP) mechanisms to RA. We briefly overview DP in this subsection and describe how AKR applies DP to RA [19] in the next subsection.

In the standard (central) setting, a trusted data curator adds noise (e.g., through the Laplace mechanism or Geometric mechanism) to fulfill a DP notion (e.g., $(\epsilon,\delta)$-DP) given a query from a data consumer, which provably bounds the information leakage. $(\epsilon,\delta)$-DP [61] is formally defined in Definition 2.1.

**Laplace Mechanism [61].** It computes a function $f$ on input dataset $D$ while satisfying $\epsilon$-DP, by adding to $f(D)$ a random noise. The magnitude of the noise depends on $\mathsf{GS}_f$, i.e., the *global* $L_1$ *sensitivity* of $f$, defined as (on any two neighboring datasets $D \simeq D'$),

$$\mathsf{GS}_f = \max_{D \simeq D'} ||f(D) - f(D')||_1 \tag{4.1}$$

When $f$ outputs a single element, $\mathcal{M}$ can be written as:

$$\mathcal{M}(D) \quad = f(D) + \mathcal{L}\left(\frac{\mathsf{GS}_f}{\epsilon}\right) \tag{4.2}$$

where $\mathcal{L}(s)$ denotes a random variable sampled from the Laplace distribution with scale parameter $s$ such that $\mathsf{Pr}[\mathcal{L}(s) = x] = \frac{1}{2s} e^{-|x|/s}$. When $f$ outputs a vector, $\mathcal{M}$ adds independent samples of $\mathcal{L}\left(\frac{\mathsf{GS}_f}{\epsilon}\right)$ to each element of the vector.

**Geometric Mechanism [142].** If the output domain is discrete, one can use this mechanism, which draws noise from the double geometric distribution: $\mathsf{Pr}[\mathcal{DG}(s) = x] = \frac{1-e^{-\frac{1}{s}}}{1+e^{-\frac{1}{s}}} e^{-\frac{1}{s}|x|/\mathsf{GS}_f}$, for $x \in \mathbb{Z}$. The Geometric mechanism satisfies $\epsilon$-DP.

**Composition.** Two properties, i.e., *composition*, and *post-processing*, of DP, are frequently used to build complicated algorithms from the basic mechanisms. See more details of these

properties in Section 2.1.2. Sequential composition states that combining multiple subroutines that satisfy DP for $(\epsilon_1,\delta_1),(\epsilon_2,\delta_2),\cdots$ results in a mechanism that satisfies $(\epsilon,\delta)$-DP for $\epsilon=\sum\epsilon_i$ and $\delta=\sum\delta_i$. On the other hand, advanced composition, e.g., Rényi DP [172], can provide smaller privacy degredation ($\epsilon$ grows sub-linearly) comparing to sequential composition.

**Definition 4.1** (Rényi Differential Privacy [172]). *A mechanism $\mathcal{M}:\mathcal{X}\to\mathcal{Y}$ is said to satisfy $(\nu,\tau)$-RDP if the following holds for any two neighboring datasets $D,D'$*

$$\frac{1}{\nu-1}\log\mathbb{E}_{o\sim\mathcal{M}(D)}\left[\left(\frac{\mathsf{Pr}[\mathcal{M}(D)=o]}{\mathsf{Pr}[\mathcal{M}(D')=o]}\right)^{\nu}\right]\leq\tau. \tag{4.3}$$

**Theorem 4.1** (RDP Sequential Composition [172]). *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are $(\nu,\tau_1)$-RDP and $(\nu,\tau_2)$-RDP respectively then the mechanism combining the two $g(\mathcal{M}_1(D),\mathcal{M}_2(D))$ is $(\nu,\tau_1+\tau_2)$-RDP.*

**Theorem 4.2** (RDP to $(\epsilon,\delta)$-DP [172]). *If a mechanism is $(\nu,\tau)$-RDP, then it also satisfies $(\tau+\frac{\log 1/\delta}{\nu-1},\delta)$-DP.*

**$k$-fold Adaptive Composition [63].** In cases where the attacker interacts with the DP algorithms over multiple rounds, the $k$-fold adaptive composition is to describe the privacy leakage in these cases over time.

**Theorem 4.3.** *For every $\epsilon>0$, $\delta$, $\delta'>0$ and $k\in\mathbb{N}$, the class of $(\epsilon,\delta)$-differentially private mechanisms is $(\epsilon',k\delta+\delta')$-differentially private under $k$-fold adaptive composition, for*

$$\epsilon'=\sqrt{2k\ln(1/\delta')}\cdot\epsilon+k\cdot\epsilon\epsilon_0, \tag{4.4}$$

*where $\epsilon_0=e^{\epsilon}-1$. For pure differential privacy, $\delta$ is set to 0.*

### 4.2.3 Differentially Private Allocation in AKR

As all the requests are identical from the allocator's point of view, the key of providing privacy is to "control" the number of resources the attacker receives. Thus, AKR asks the allocator to add dummy requests. Specifically, AKR sets the dataset $D$ to be all requests made by clients, and computes the noise $\mathcal{L}\left(\frac{\mathsf{GS}_f}{\epsilon}\right)$. To ensure the number of added requests (i.e., $\mathcal{M}(D)$ in Equation 4.2) is non-negative, a bias $\mu$ is added when sampling the Laplace noise so that the probability of the noise being negative is bounded by $\delta$, which we refer to as the *biased Laplace distribution*. The workflow of AKR is:

- **Input**: $k$, $\mu$, $\mathsf{GS}_f$, $\epsilon$, $D$
- Noise $d \leftarrow \left\lceil \max\left(0, \mu + \mathcal{L}\left(\frac{\mathsf{GS}_f}{\epsilon}\right)\right)\right\rceil$
- Set $Q \leftarrow |D| + d$ dummy requests
- $U \leftarrow$ uniformly select $\min(|Q|, k)$ items out of $Q$
- **Output**: $U$

Overall, AKR satisfies $(\epsilon,\delta)$-DP. Below is its DP proof.

**Theorem 4.4** (DP Proof for AKR [19]). *Algorithm $\mathcal{M}$ is $(\epsilon, \delta)$-differentially private for $\epsilon = 1/s$ and $\delta = \int_{-\infty}^{1} \mathcal{L}(w|\mu, 1/\epsilon)dw$. Specifically, for any subset of values $L$ in the range $[f(D), \infty)$ of $\mathcal{M}$:*

$$\mathsf{Pr}[\mathcal{M}(D) \in L] \leq e^{\epsilon}\mathsf{Pr}[\mathcal{M}(D') \in L] + \delta \tag{4.5}$$

*and*

$$\mathsf{Pr}[\mathcal{M}(D') \in L] \leq e^{\epsilon}\mathsf{Pr}[\mathcal{M}(D) \in L] \tag{4.6}$$

*where $f(S)$ computes the cardinality of set $S$.*

Note that:

$$\delta = \int_{-\infty}^{1} \mathcal{L}(w|\mu,1/\epsilon)dw = \begin{cases} \frac{1}{2}e^{\epsilon(1-\mu)} & \text{if } \mu > 1 \\ 1-\frac{1}{2}e^{\epsilon(1-\mu)} & \text{if } \mu \leq 1 \end{cases} \tag{4.7}$$

We can see $\mu$ tends to be large in order to have a small $\delta$.

Given that the noise is non-negative, what the attacker observes after allocation can be seen as a post-processing of the requests, as the victim's request is indistinguishable from the added dummy requests. Specifically, let $Y$ be a random variable denoting the number of resource attacker gets. Since the attacker only learns which requests of her were fulfilled, from her point of view dummy requests and victim are indistinguishable. Thus for each value $l \in [0,k]$, $\mathsf{Pr}[Y=l|\mathcal{M}(D)=t] = \mathsf{Pr}[Y=l|\mathcal{M}(D')=t]$, where $t$ is the number of requests with dummies. Combined with the inequalities governing the probabilities that $\mathcal{M}$ outputs each value of $t$ for $D$ and $D'$, respectively. We have that $\mathsf{Pr}[Y=l|D] \leq e^{\epsilon}\mathsf{Pr}[Y=l|D'] + \delta$, and similarly with $D$ and $D'$ exchanged. Thus the distribution of the number of attacker's requests allocated are very close for $D$ and $D'$.

## 4.3 Modeling Resource Allocation

In this section, we first demonstrate the problem of AKR's modeling of RA. Then, we present a taxonomy of different ways to "add noise" in RA and a general approach to model privacy.

### 4.3.1 Privacy Amplification from Allocation

We argue that AKR's modeling of RA leads to suboptimal utility due to the lack of consideration for the attacker's view and capabilities. Though AKR, by its definition, does not reveal

the number of total requests each round, their proof indicates a stronger statement that the DP guarantee holds even when the attacker observes the *total* number of requests after noise is added (i.e., the number of requests from both the attacker and the victim). More specifically, their proof guarantees that the noisy total number of requests is bounded by ($\epsilon$,$\delta$)-DP when honest clients are added. However, such information is not actually accessible to the attacker, thus it creates a gap between the proof and the actual definition of the RA problem. Examining the attacker's view is crucial for privacy amplification in our study. By comprehending the capabilities and limitations of the attackers, we can construct a precise analysis and avoid unnecessary noise. In real-world scenarios, the capability of an attacker can be considerably limited, as they are typically not granted access to the internal states of an allocator. In fact, if the attacker can observe the internal states of an allocator, she just needs to access the number of requests *before* adding noise, which defeats all DP-based protection.

We note that such a modeling gap is common in DP for ease of proof. For example, in DP-SGD [71], the privacy guarantee is proved on each SGD step, implying that the attacker can observe the intermediate steps, but such information should not be accessible to the attacker. A similar case also appears in the proof of privacy blanket [24, Theorem 3.1] (which assumes the attacker has unrealistic extra information for the ease of proof) for the shuffle DP model.

Hence, we propose to more precisely model the attacker's capabilities and offer a tighter bound under the notion of DP. By conducting the privacy analysis from scratch, we present a set of "*privacy amplification*" results[3]. In this dissertation, the privacy amplification stems from the fact that the attacker only has a *partial* view of the allocation result. The attacker is aware of whether the other compromised clients receive the allocated resources, except for the one uncompromised client. Compared to AKR, which has to introduce larger noise to deter the (unrealistic) attacker, we can use smaller noise to satisfy DP. In Section 4.5.2

[3]Privacy amplification refers to the effect where we can prove the privacy cost is reduced after some operations (e.g., subsampling [23] and shuffling [67]).

("Why Models Attacker's View"), we elaborate the impact of privacy amplification.

### 4.3.2 Design Space

As described in Section 4.2.1, RA takes two steps: (1) receive a request, and (2) allocate the resource if the request is accepted. Hence, for privacy protection, the allocator can add noise to either (1) the number of requests (i.e., by adding dummy requests or removing some requests), or (2) the number of available resources (i.e., by withholding some available resource). After that, the allocator can randomly select requests and assign resources to them. Therefore, the design space for the allocator is composed of:

- **DS1: Choosing Where to Add Noise.** The allocator can add noise to either the number of requests or the number of resources or both. Our analysis shows that randomizing the number of resources has the same effect as randomizing the number of requests(explained later), thus we focus on designing methods to add noise to the number of requests. In Section 4.6.4, we give a few real-world examples.
- **DS2: Choosing How Noise is Generated**. The allocator adds noise to the observed number of requests, and we have the flexibility to choose:
  - The distribution of the noise.
  - The range (support) of the distribution.

We found AKR only covered part of the design space: (1) AKR considered RA as post-processing and only adds *non-negative* noise (dummy requests) to the requests. (2) AKR did not consider distributions other than the Laplace distribution.

**Adding Noise to Resource.** Beyond adding noise to the requests, we can choose to add noise to the resources. Here we consider that the noise is always negative, or the resources are withheld from being assigned to clients. The positive noise can be seen as "creating"

resources on the fly and assigning more than what is asked by a client, which could be impractical for a real-world system. Yet, we can prove that withholding any number of resources can be equivalently modeled as assigning them to dummy requests. Specifically, the allocator could withhold $n$ resources from $k$ requests, which results in $k-n$ random requests getting resources. This is equivalent to that $n$ requests being randomly removed from the system (so that the rest $k-n$ requests are granted with resources). Thus, we only consider adding noise to requests.

### 4.3.3 Privacy Modeling

Under DS1, we model RA's privacy through the lens of DP as follows. We use $d$ to denote the random variable for the number of noisy requests. $D$ denotes the number of requests made in a round. Given two neighboring datasets $D$, $D'$, w.l.o.g., we assume $D'$ equals to $D$ *plus* the honest request from the victim client[4]. RA's privacy can be quantified as:

$$\frac{\mathsf{Pr}\big[\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(D)=y\big]}{\mathsf{Pr}\big[\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(D')=y\big]}=\frac{\sum_{i=x_\ell}^{x_r}\mathsf{Pr}[d=i]\mathsf{Pr}[y\,||D|+d]}{\sum_{i=x_\ell}^{x_r}\mathsf{Pr}[d=i]\mathsf{Pr}[y\,||D'|+d]} \tag{4.8}$$

where $\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(\cdot)$ models the allocation outcomes in the attacker's view. Note that $\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}$ differs from $\mathcal{M}$ in Equation 2.1 in that $\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}$ is a partial view of the final allocation outcome. $\mathsf{Pr}[d=i]$ denotes the probability $d=i$, where $d$ is a random variable and $i$ is within some range $[x_\ell, x_r]$, and $\mathsf{Pr}[y\,||D|+d]$ is the probability that attacker gets $y$ resources. This equation measures the difference in the attacker's observation that is impacted by the one honest request. If $d \geq 0$, the allocator adds some dummy requests; $d < 0$ models removing some requests (e.g., ignoring requests). Notice that Equation 4.8 follows $\epsilon$-DP, which is different

[4]For the other neighboring case ($D'$ equals to $D$ *minus* the honest request), the modeling and proofs are similar, so it is omitted in this version due to page limit.

from AKR that follows ($\epsilon$, $\delta$)-DP.

With $\mathsf{Pr}[y||D|+d]$, we are able to more precisely model RA privacy than AKR and captures the randomness introduce by RA, since $y$ represents only the output in the attacker's view (i.e., $y \leq |D|$). We now describe the detailed analysis of $\mathsf{Pr}[y||D|+d]$ under two cases: $d \geq 0$ and $d < 0$. We enumerate all possible situations under RA and derive the exact probability expressions for $\mathsf{Pr}[y||D|+d]$ and $\mathsf{Pr}[y||D'|+d]$(i.e., Equation 4.9 and Equation 4.10).

**Request Addition ($d \geq 0$).** For the case of $D$, assuming there are $m$ requests from $D$, given a specific number of dummy requests $d \geq 0$, we have:

$$\forall (k-d)_+ \leq y \leq \min(k,m), \quad \mathsf{Pr}[y||D|+d] = \frac{\binom{m}{y}\binom{d}{k-y}}{\binom{m+d}{k}} \tag{4.9}$$

$\mathsf{Pr}[y||D|+d] = 0$ if $y$ is outside of the above range. $y$ has to satisfy $y \leq \min(k,m)$ because what the attacker observes cannot exceed the total number of resources $k$ or the number of requests $m$. Similarly, $y \geq (k-d)_+$ (we use $x_+$ to denote $\max(0,x)$) because there are only $d$ other requests, so the attacker must get at least $(k-d)_+$ resources.

We only model the case when the number of requests $m \geq k-d$ because when $m < k-d$, all requests are fulfilled (no privacy leakage). In that case, $\mathsf{Pr}[y||D|+d] = 1$ for $y = m$ and $\mathsf{Pr}[y||D|+d] = 0$ otherwise.

The denominator of Equation 4.9 is $\binom{m+d}{k}$ because we have a total of $m+d$ requests and we allocate $k$ resources to them (equivalent to choosing $k$ from $m+d$ requests to allocate resources). Thus there are $\binom{m+d}{k}$ possible assignments. The numerator is $\binom{m}{y}\binom{d}{k-y}$ because, for the fixed set of $m$ requests controlled by the attacker, $y$ of them are fulfilled; there are $\binom{m}{y}$ possible assignments. Similarly, for the rest $d$ requests, there are $\binom{d}{k-y}$ possible assignments. So all together there are $\binom{m}{y}\binom{d}{k-y}$ possible assignments that satisfy the constraint that $y$ resources go to $m$ processes.

For the case of $D'$, which has an additional honest request, the attacker could receive one fewer resource. Thus we have:

$$\forall (k-d-1)_+ \leq y \leq \min(k,m), \quad \mathsf{Pr}[y \,||D'|+d] = \frac{\binom{m}{y}\binom{d+1}{k-y}}{\binom{m+d+1}{k}} \tag{4.10}$$

Similar to Equation 4.9, in Equation 4.10, when $m < k-d-1$, $\mathsf{Pr}[y \,||D'|+d] = 1$ for $y = m$ and $\mathsf{Pr}[y \,||D'|+d] = 0$ otherwise.

**Request Removal ($d < 0$).** For the case of $D$ (the honest request does not exist), when the number of added dummy requests is negative ($d < 0$), some requests will be removed randomly. We have:

$$\mathsf{Pr}[y \,||D|+d] = \begin{cases} 1 & \text{if } y = \min(m+d,k)_+ \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

This case is simpler than "Request Addition", and what the attacker observes is deterministic: if after adding negative noise $d$, $m+d$ is still greater than $k$, then the attacker will always receive $k$ resources; if $m+d \leq k$, then the attacker will always receive $m+d$ resources.

For the case of $D'$, there are $m+1+d$ requests, and we need to consider whether the honest request is fulfilled. Let $x = \min(m+1+d,k)_+$, which leads to two scenarios:

- Allocator assigns resources to the honest client: in this case, $y$ can only be $x-1$. The probability of the allocator assigning resources to the honest client is $\frac{x}{m+1}$, which is equivalent to the case of selecting $x = \min(m+1+d,k)_+$ items from a total of $m+1$ items without replacement and that the honest client is selected.
- Allocator does not assign resources to the honest client: $y$ must be $x$ if the honest

request is not fulfilled, which happens with probability $1-\frac{x}{m+1}$.

Thus we have:

$$\mathsf{Pr}[y\,||D'|+d] = \begin{cases} 1-\frac{x}{m+1} & \text{if } y=x \\ \frac{x}{m+1} & \text{if } y=x-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

where $x=\min(m+1+d,k)_+$.

We want to highlight that considering request removal (negative noise) is another key difference from AKR.

**Attacker's Strategy.** From the attacker's point of view, it is important to set $m$ (the number of compromised clients) to a value that can maximize privacy leakage (i.e., maximize Equation 4.8). Recall that we assume $k$ (resource capacity) is known to the attacker, and each client can submit at most one request (see Section 4.2.1). Following the previous analysis of request addition and request removal, we can derive the best attacker strategy below we follow this strategy for this rest of the paper.

**Theorem 4.5.** *The maximum privacy leakage happens when the attacker sends $m=k$ requests.*

*Proof.* We consider the cases of noise $d<0$ and $d\geq 0$, and prove $m=k$ causes maximum privacy leakage in both cases.

First, considering the case when noise is non-negative ($d\geq 0$), the attacker's goal is to choose $m$ to maximize the difference between the cases of $D$ and $D'$. Note that the difference can only be observed when $m+d\geq k$ because otherwise, all requests will be granted with resources. To ensure $m+d\geq k$ for all $d\geq 0$, we have $m\geq k$. Based on the previous analysis,

when $0 \leq d < k$, there is no privacy at $y = k - d - 1$, because

$$\mathsf{Pr}[y \,||D|+d] = 0, \mathsf{Pr}[y \,||D'|+d] = \frac{m!k!}{(k-d-1)!(m+d+1)!} \tag{4.13}$$

Thus it does not matter to the attacker what value to set to $m$ in this case. For $d \geq k$, the privacy protection is given by

$$\frac{\mathsf{Pr}[y \,||D|+d]}{\mathsf{Pr}[y \,||D'|+d]} = \frac{\binom{m}{y}\binom{d}{k-y} / \binom{d+m}{k}}{\binom{m}{y}\binom{d+1}{k-y} / \binom{d+m+1}{k}} \leq 1 + \frac{k}{m+d+1-k} \tag{4.14}$$

In order to maximize the above, we need to set $m$ to its minimum within the range of $m \geq k$, that is, $m = k$.

Now, we consider the case when negative noise ($d < 0$) is added. By observing Equation 4.11 and Equation 4.12, we know that to trigger the different outputs for case $D$ and $D'$ (i.e., $y = m + d$ for case $D$ and $y = m + d + 1$ for case $D'$), $m + d$ needs to be $< k$. The difference of $D$ and $D'$ (privacy protection) is then given by

$$\frac{\mathsf{Pr}[y \,||D|+d]}{\mathsf{Pr}[y \,||D'|+d]} = \frac{1}{1 - \frac{m+1+d}{m+1}} = \frac{m+1}{-d} \tag{4.15}$$

To have $m + d < k$ (i.e., $d < k - m$) hold for all $d < 0$, we have $m \leq k$. Now, in order to maximize Equation 4.15, $m$ is to be set to $k$. □

## 4.4 Noisy Mechanisms

In this section, we analyze different noisy mechanisms under DS2. As the RA output is discrete, we choose discrete distributions for the mechanisms. Specifically, we consider constant, uniform, one-sided geometric, and double geometric distributions, and name them CST, UNI, GEO, DGEO for short. Though these mechanisms have been studied in the standard DP [81, 79], we conducted new theoretical analysis to derive tighter privacy bounds,

| | CST | UNI | GEO | DGEO | AKR [19] |
|---|---|---|---|---|---|
| Privacy | $\epsilon$-DP | $\epsilon$-DP | $\epsilon$-DP | $\epsilon$-DP | ($\epsilon$, $\delta$)-DP |
| Noise | Constant | Discrete uniform | One-sided geometric | Double geometric | Laplace |
| Noise Sign | + | +/− | +/− | +/− | + |
| DP Condition | Noise $c \geq k$ | Right bound $x_r \geq k$ | - | - | - |
| Utility ($\epsilon$=0.65) | **0.50** | 0.46 | 0.47 | 0.44 | 0.32 |
| Utility ($\epsilon$=1.7) | 0.50 | 0.65 | **0.82** | 0.77 | 0.53 |
| Utility ($\epsilon$=2.3) | 0.50 | 0.70 | 0.90 | **0.98** | 0.59 |

Table 4.2: A summary of different mechanisms and their utility under some representative $\epsilon$ values. Note that $k=10$ and $\delta=10^{-6}$.

which require extensive proof work. In Table 4.2, a summary of different mechanisms is given. In particular, 1) we prove the DP bounds for all mechanisms, though CST and UNI only satisfy DP when certain conditions are met (i.e., noise sample space should be at least $k$); 2) our mechanisms outperform AKR in utility by a large margin[5].

### 4.4.1 Constant Noise (CST)

In this case, we consider request addition only, and the noise $d$ always equals a constant number $c$. Observing Equations (4.9) and (4.10), the valid $y$ support sets differ in one case where $y=k-d-1$. But as long as $d \geq k$, both $\mathsf{Pr}[y||D|+d]$ and $\mathsf{Pr}[y||D'|+d]$ have the same valid set of $y \in \{0,1,\cdots,\min(m,k)\}$, and the privacy can be quantified as:

$$\frac{\mathsf{Pr}[y||D|+d]}{\mathsf{Pr}[y||D'|+d]} = \frac{\frac{\binom{m}{y}\binom{d}{k-y}}{\binom{d+m}{k}}}{\frac{\binom{m}{y}\binom{d+1}{k-y}}{\binom{d+m+1}{k}}} = \frac{(m+d+1)(d+y+1-k)}{(m+d+1-k)(d+1)} \tag{4.16}$$

As a result, we have the following theorem:

**Theorem 4.6.** *Assuming an allocator has $k$ resources, constant noise has to be at least $k$ to satisfy DP.*

*Proof.* Suppose the resource allocated to the attacker is $y$, and the attacker always sends

[5]The numbers come from simulation of Section 4.5.2. The theoretical analysis of utility has also been done, but omitted due to page limit.

out $m=k$ requests. Then we have

$$\begin{cases} y\in\{(k-d)_+,(k-d)_++1,\cdots,k\} & \text{when } D \\ y\in\{(k-d-1)_+,(k-d-1)_++1,\cdots,k\} & \text{when } D' \end{cases} \tag{4.17}$$

Note that $y=(k-d)_+$ happens in $D$ when all dummy requests get resources and the remaining resources go to the attacker. Similarly, $y=(k-d-1)_+$ happens in $D'$ when the victim and all dummy requests get resources and the remaining resources go to the attacker. When $d\geq k$, $y\in\{0,1,\cdots,k\}$ for both $D$ and $D'$. Thus, given $m=k$ and Equation 4.16, we can give an upper-bound of its privacy leakage as:

$$e^{\epsilon}=\frac{\mathsf{Pr}[y\,||D|+d]}{\mathsf{Pr}[y\,||D'|+d]}=\frac{(m+d+1)(d+y+1-k)}{(m+d+1-k)(d+1)}\leq\frac{k+d+1}{d+1} \tag{4.18}$$

where $e^{\epsilon}=\frac{k+d+1}{d+1}$ is reached at $y=k$. □

This is a surprising result, as adding a fixed noise should not satisfy DP. In our case, adding a fixed noise still provides privacy because of the randomness of the allocation process. Still, we argue that it does not offer good utility. Due to the constraint $c\geq k$, the utility is never more than 0.5.

### 4.4.2 Uniform Mechanism (UNI)

In this case, the discrete noise (it can be negative or non-negative) is drawn uniformly from $[x_\ell,x_r]$:

$$\mathsf{Pr}[d=i]=\frac{1}{x_r-x_\ell+1},i=x_\ell,x_\ell+1,...,x_r \tag{4.19}$$

$x_\ell$ and $x_r$ define the shape of distribution used in UNI, with $x_\ell$ defining the starting point. Below, we prove that the attacker's view satisfies DP when $x_r\geq k$ or $[-k-1,0]\in[x_\ell,x_r]$.

**Theorem 4.7.** *Assume the server has k resources. Adding a random noise drawn uniformly from* $\{x_\ell, x_\ell+1, \ldots, x_r\}$ *(both* $x_\ell$ *and* $x_r \geq k$ *are integers) to the number of requests satisfies* $\epsilon$*-DP, where*

$$e^\epsilon \leq \max_{y \in Y} \left( \frac{f(y) + \sum_{i=\max(x_\ell, k-y)}^{x_r} g(i)}{f_1(y) + \sum_{i=\max(x_\ell, (k-y-1)_+)}^{x_r} g(i+1)} \right) \tag{4.20}$$

*and* $Y = \{1,2,\ldots,k\}$, $g(i) = \frac{(k!)^2 (i!)^2}{y!((k-y)!)^2 (i-k+y)!(k+i)!}$, $f(y) = \mathbb{1}_{x_\ell+k \leq y < k} + (-k-x_\ell)\mathbb{1}_{y=0, x_\ell < -k}$,
$f_1(y) = \frac{k-y+1}{k+1}\mathbb{1}_{x_\ell+k+1 \leq y \leq k} + \frac{y+1}{k+1}\mathbb{1}_{x\ell+k \leq y < k} + (-k-1-x_\ell)\mathbb{1}_{y=0, x_\ell < -k-1}$.

*Proof.* Assume an allocator has $k$ resources. W.l.o.g., $D$ contains $m$ requests and $D'$ contains $m+1$ requests. Before going further into examination of the privacy, we first consider the value of $m$. For the view of an attacker, it is crucial to set $m$ to an optimal value that causes maximum leakage during allocation. This optimal value is $k$ is shown in previous analysis in Section 4.3.2.

We examine the probability the attacker gets assigned $y$ resources after allocation. In the case of $D$,

$$\begin{aligned}
&\mathsf{Pr}\left[\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(D) = y\right] = \sum_{i=-\infty}^{+\infty} \mathsf{Pr}[d=i]\mathsf{Pr}[y \,||D|+d] \\
&= \sum_{i=\max(x_\ell, k-y)}^{x_r} \mathsf{Pr}[d=i]\mathsf{Pr}[y \,||D|+d] + \mathsf{Pr}[d=y-k]\cdot 1 + \mathsf{Pr}[d<-k]\mathbb{1}_{y=0} \\
&= \frac{1}{x_r - x_\ell + 1} \left( \sum_{i=\max(x_\ell, k-y)}^{x_r} \frac{\binom{m}{y}\binom{i}{k-y}}{\binom{m+i}{k}} + \mathbb{1}_{x_\ell+k \leq y < k} + (-k-x_\ell)\mathbb{1}_{y=0, x_\ell < -k} \right) \\
&= \frac{1}{x_r - x_\ell + 1} \left( \mathbb{1}_{x_\ell+k \leq y < k} + (-k-x_\ell)\mathbb{1}_{y=0, x_\ell < -k} + \sum_{i=\max(x_\ell, k-y)}^{x_r} \frac{k!k!i!i!}{y!((k-y)!)^2 (i-k+y)!(k+i)!} \right)
\end{aligned}$$

Similarly, for the case of $D'$,

$$\mathsf{Pr}\big[\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(D')=y\big]=\sum_{i=-\infty}^{+\infty}\mathsf{Pr}[d=i]\mathsf{Pr}[y\,||D'|+d]$$

$$=\sum_{i=\max(x_\ell,k-y-1,0)}^{x_r}\mathsf{Pr}[d=i]\mathsf{Pr}[y\,||D'|+d]+\mathsf{Pr}[d<-k-1]\mathbb{1}_{y=0}+\frac{k-y+1}{k+1}\mathsf{Pr}[d=y-k-1]$$

$$+\frac{y+1}{k+1}\mathsf{Pr}[d=y-k]$$

$$=\frac{1}{x_r-x_\ell+1}\left(\sum_{i=\max(x_\ell,k-y-1,0)}^{x_r}\frac{\binom{m}{y}\binom{i+1}{k-y}}{\binom{m+i+1}{k}}+\left(\frac{y+1}{k+1}\right)\mathbb{1}_{x_\ell+k\le y<k}\right.$$

$$\left.+\frac{k-y+1}{k+1}\mathbb{1}_{x_\ell+k+1\le y\le k}+(-k-1-x_\ell)\mathbb{1}_{y=0,x_\ell<-k-1}\right)$$

$$=\frac{1}{x_r-x_\ell+1}\left(\frac{k-y+1}{k+1}\mathbb{1}_{x_\ell+k+1\le y\le k}+\frac{y+1}{k+1}\mathbb{1}_{x\ell+k\le y<k}+(-k-1-x_\ell)\mathbb{1}_{y=0,x_\ell<-k-1}\right.$$

$$\left.+\sum_{i=\max(x_\ell,(k-y-1)_+)}^{x_r}\frac{(k!)^2((i+1)!)^2}{y!(k-y)!^2(i+1-k+y)!(k+i+1)!}\right)$$

Therefore, privacy protection here satisfies

$$e^\epsilon\le\max_{y\in Y}\left(\frac{f(y)+\sum_{i=\max(x_\ell,k-y)}^{x_r}g(i)}{f_1(y)+\sum_{i=\max(x_\ell,(k-y-1)_+)}^{x_r}g(i+1)}\right)\tag{4.21}$$

where $Y=\{1,2,...,k\}$, $g(i)=\frac{(k!)^2(i!)^2}{y!((k-y)!)^2(i-k+y)!(k+i)!}$, $f(y)=\mathbb{1}_{x_\ell+k\le y<k}+(-k-x_\ell)\mathbb{1}_{y=0,x_\ell<-k}$, $f_1(y)=\frac{k-y+1}{k+1}\mathbb{1}_{x_\ell+k+1\le y\le k}+\frac{y+1}{k+1}\mathbb{1}_{x\ell+k\le y<k}+(-k-1-x_\ell)\mathbb{1}_{y=0,x_\ell<-k-1}$.

□

Yet, our analysis shows UNI is also not recommended when the utility requirement is more critical. This is because utility degrades linearly to negative noise when number of requests equals to the number of resources. In a nutshell, suppose total number requests is $n$ and $n=k$, removing requests causes less resource to be allocated with certainty, while adding requests results in the same with a probability. With this, our goal is to have $x_\ell\ge 0$ and

$x_r \geq k$ in order to achieve best privacy and utility tradeoff, and Section 4.5.3 studies how these parameters should be determined.

### 4.4.3 One-sided Geometric Mechanism (GEO)

Intuitively, reducing the probability density of large noise can reduce the amount of dummy requests added, and thus improve utility. To this end, we adopt the geometric distribution within the range $[x_\ell,\infty)$ with the noise distribution:

$$\mathsf{Pr}[d=i]=p(1-p)^{i-x_\ell}, i=x_\ell,x_\ell+1,x_\ell+2,... \tag{4.22}$$

Like UNI, $x_\ell$ also models the starting point of the new distribution. For $p$, a larger value makes the noise decay faster and has negligible probability for large value $i$, thus improving utility. In terms of privacy, we can also prove attacker's view satisfies DP (See Theorem 4.8 below).

**Theorem 4.8.** *Assume the server has $k$ resources. Adding a random noise drawn from the geometric distribution (with parameter $p$ and starting from integer $x_\ell$) to the number of requests satisfies $\epsilon$-DP, where*

$$e^\epsilon \leq \max_{y\in Y}\left(\frac{f(y)+\frac{p((k)!)^2}{y!((k-y)!)^2}\sum_{i=\max(k-y,x_\ell)}^{\infty} g(i)}{f_1(y)+\frac{p(1-p)^{-1}((k)!)^2}{y!((k-y)!)^2}\sum_{i=x_0}^{\infty} g(i+1)}\right)$$

*and* $Y=\{0,1,...,k\}$, $g(i)=\frac{(1-p)^{i-x_\ell}(i!)^2}{(i-k+y)!(k+i)!}$,
$f(y) = (1-p)^{y-k-x_\ell}\mathbb{1}_{k+x_\ell\leq y<k} + \left(1-(1-p)^{-k-x_\ell}\right)\mathbb{1}_{y=0,x_\ell<-k}$, $f_1(y) = \frac{p}{k+1}(-y+k+1)(1-p)^{y-k-1-x_\ell}\mathbb{1}_{k+1+x_\ell\leq y\leq k} + \frac{1+y}{k+1}p(1-p)^{y-k-x_\ell}\mathbb{1}_{k+x_\ell\leq y<k} + \left(1-(1-p)^{-k-x_\ell-1}\right)\mathbb{1}_{y=0,x_\ell<-k-1}$, $x_0 = \max((k-y-1)_+,x_\ell)$.

*Proof.* Given an allocator with $k$ resources and an attacker sending $m=k$ requests, we assess

the probability of the attacker being allocated $y$ resources.

$$\begin{aligned}
&\mathsf{Pr}\big[\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(D)=y\big]\\
=&\sum_{i=\max(k-y,x_\ell)}^{\infty} p(1-p)^{i-x_\ell}\frac{\binom{m}{y}\binom{i}{k-y}}{\binom{m+i}{k}}+f(y)\\
=&\frac{p(k!)^2}{y!((k-y)!)^2}\sum_{i=\max(k-y,x_\ell)}^{\infty}\frac{(1-p)^{i-x_\ell}(i!)^2}{(i-k+y)!(k+i)!}+f(y)
\end{aligned}$$

where

$f(y)=p(1-p)^{y-k-x_\ell}\mathbb{1}_{k+x_\ell\le y<k}+\big(1-(1-p)^{-k-x_\ell}\big)\mathbb{1}_{y=0,x_\ell<-k}$. Similarly, for the other case,

$$\begin{aligned}
&\mathsf{Pr}\big[\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(D')=y\big]\\
=&\sum_{i=x_0}^{\infty}p(1-p)^{i-x_\ell}\frac{\binom{m}{y}\binom{i+1}{k-y}}{\binom{m+i+1}{k}}+f'(y)=\sum_{i=x_0}^{\infty}\frac{p(1-p)^{i-x_\ell}(k!)^2((i+1)!)^2}{y!((k-y)!)^2(i+1-k+y)!(k+i+1)!}+f_1(y)
\end{aligned}$$

where $x_0=\max((k-y-1)_+,x_\ell)$ and

$$\begin{aligned}
f_1(y)=&\frac{(-y+k+1)p(1-p)^{y-k-1-x_\ell}}{k+1}\mathbb{1}_{k+1+x_\ell\le y\le k}\\
&+\frac{(1+y)p(1-p)^{y-k-x_\ell}}{k+1}\mathbb{1}_{k+x_\ell\le y<k}\\
&+\big(1-(1-p)^{-k-x_\ell-1}\big)\mathbb{1}_{y=0,x_\ell<-k-1}
\end{aligned}$$

Given above numerator and denominator, we have the privacy protection satisfies

$$e^{\epsilon}\le\max_{y\in Y}\left(\frac{f(y)+\frac{p((k)!)^2}{y!((k-y)!)^2}\sum_{i=\max(k-y,x_\ell)}^{\infty}g(i)}{f_1(y)+\frac{p(1-p)^{-1}((k)!)^2}{y!((k-y)!)^2}\sum_{i=x_0}^{\infty}g(i+1)}\right)$$

where $Y=\{0,1,...,k\}$, $g(i)=\frac{(1-p)^{i-x_\ell}(i!)^2}{(i-k+y)!(k+i)!}$,
$f(y)=(1-p)^{y-k-x_\ell}\mathbb{1}_{k+x_\ell\le y<k}+\big(1-(1-p)^{-k-x_\ell}\big)\mathbb{1}_{y=0,x_\ell<-k}$, $f_1(y)=\frac{p}{k+1}(-y+k+1)(1-p)^{y-k-1-x_\ell}\mathbb{1}_{k+1+x_\ell\le y\le k}+\frac{1+y}{k+1}p(1-p)^{y-k-x_\ell}\mathbb{1}_{k+x_\ell\le y<k}+\big(1-(1-p)^{-k-x_\ell-1}\big)\mathbb{1}_{y=0,x_\ell<-k-1}$ and $x_0=\max((k-y-1)_+,x_\ell)$. □

For the same reason in Section 4.4.2, negative noise has negative influence on utility in a deterministic way. Therefore, though GEO tolerates negative noise (i.e., $x_\ell$ can be negative), we do not recommend setting $x_\ell < 0$.

The two parameters $x_\ell$ and $p$ both influence $\epsilon$ and utility: For $x_\ell > 0$, increasing $x_\ell$ reduces both $\epsilon$ and utility, and increasing $p$ raises $\epsilon$ and utility. For $x_\ell < 0$, utility and privacy varies in different cases. Section 4.5.3 studies the parameter settings.

### 4.4.4 Double Geometric Mechanism (DGEO)

AKR adds a biased Laplace noise to the number of requests (explained in Section 4.2.3). Likely, we propose to draw the noise from a biased double geometric distribution:

$$\Pr[d=i] = \frac{1-e^{-\epsilon}}{1+e^{-\epsilon}} e^{-\epsilon|i-\mu|}, \forall i \in \mathbb{Z} \tag{4.23}$$

We call $s = 1/\epsilon$ the scale of the noise and $\mu$ the bias of the noise. Adding double-geometric noise with a scale $1/\epsilon$ to the number of requests satisfies $\epsilon$-DP [61, 142], and we prove it below.

**Theorem 4.9.** *Assume the server has $k$ resources. Adding a random noise drawn from the double geometric distribution (with bias $\mu$ and scale $s$) to the number of requests satisfies $\epsilon$-DP, where*

$$e^\epsilon \leq \max_{y \in Y} \left( \frac{f(y) + \sum_{i=(k-y)_+}^{+\infty} e^{-\frac{1}{s}|i-\mu|} g(i)}{f_1(y) + \sum_{i=(k-y-1)_+}^{+\infty} e^{-\frac{1}{s}|i-\mu|} g(i+1)} \right)$$

*and $Y = \{1,2,...,k\}$, $g(i) = \frac{(k!)^2 (i!)^2}{u!((k-y)!)^2 (i-k+y)!(k+i)!}$, $f(y) = e^{-\frac{1}{s}|y-k-\mu|} \mathbb{1}_{y \neq k} + \sum_{i=-\infty}^{-k-1} e^{-\frac{1}{s}|i-\mu|} \mathbb{1}_{y=0}$, $f_1(y) = \frac{e^{-\frac{1}{s}|y-k-1-\mu|}(k-y+1)}{k+1} + \frac{e^{\frac{-1}{s}|y-k-\mu|}(y+1)}{k+1} \mathbb{1}_{y \neq k} + \sum_{i=-\infty}^{-k-2} e^{-\frac{1}{s}|i-\mu|} \mathbb{1}_{y=0}$, and $s$ is the scale parameter in double geometric distribution.*

Here $f(y)$ and $f_1(y)$ in Theorem 4.9 are from negative noise and the summations are from positive noise. When positive noise is being added, the probability of attacker getting $y$ allocation can be straightforwardly calculated by substituting $\mathsf{Pr}[d=i]$ in Equation 4.8 with biased double geometric distribution.

For negative noise, attacker can only get $y<k$ with a probability of $e^{-\epsilon|y-k-\mu|}$ for the case of $D$. Whereas for the case of $D'$ the attacker can still get $k$ resources if noise equals to $-1$, and the victim is removed. Or else, the attacker will get $y<k$ resources in all other negative noise cases. This whole process is given by $f_1(y)$. Finally, privacy bound in Theorem 4.9 is derived from the worst case $y$.

*Proof.* Given an allocator with $k$ resources and an attacker sending $m=k$ requests, we assess the probability the attacker is allocated $y$ resources.

$$\mathsf{Pr}\big[\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(D)=y\big] = \sum_{i=-\infty}^{+\infty} \mathsf{Pr}[d=i]\mathsf{Pr}[y\,||D|+d]$$

$$=\frac{1-e^{-\frac{1}{s}}}{1+e^{-\frac{1}{s}}}\left(e^{-\frac{1}{s}|y-k-\mu|}\mathbb{1}_{y\neq k}+\sum_{i=-\infty}^{-k-1}e^{-\frac{1}{s}|i-\mu|}\mathbb{1}_{y=0}+\sum_{i=(k-y)_+}^{+\infty}e^{-\frac{1}{s}|i-\mu|}\frac{\binom{m}{y}\binom{i}{k-y}}{\binom{m+i}{k}}\right)$$

$$=\frac{1-e^{-\frac{1}{s}}}{1+e^{-\frac{1}{s}}}\left(\sum_{i=(k-y)_+}^{+\infty}\frac{k!k!i!i!e^{-\frac{1}{s}|i-\mu|}}{y!((k-y)!)^2(i-k+y)!(k+i)!}+e^{-\frac{1}{s}|y-k-\mu|}\mathbb{1}_{y\neq k}+\sum_{i=-\infty}^{-k-1}e^{-\frac{1}{s}|i-\mu|}\mathbb{1}_{y=0}\right)$$

Similarly,

$$\mathsf{Pr}\big[\mathsf{View}^{\mathcal{A}}_{\mathcal{M}}(D')=y\big] = \sum_{i=-\infty}^{+\infty} \mathsf{Pr}[d=i]\mathsf{Pr}[y\,||D'|+d]$$

$$=\frac{1-e^{-\frac{1}{s}}}{1+e^{-\frac{1}{s}}}\left(\frac{e^{-\frac{1}{s}|y-k-1-\mu|}(k-y+1)}{k+1}+\sum_{i=-\infty}^{-k-2}e^{-\frac{1}{s}|i-\mu|}\mathbb{1}_{y=0}+\frac{e^{-\frac{1}{s}|y-k-\mu|}(y+1)}{k+1}\mathbb{1}_{y\neq k}\right.$$

$$\left.+\sum_{i=(k-y-1)_+}^{+\infty}\frac{e^{-\frac{1}{s}|i-\mu|}(k!)^2((i+1)!)^2}{y!((k-y)!)^2(i+1-k+y)!(k+i+1)!}\right)$$

Given above numerator and denominator, we have privacy protection as follows

$$e^{\epsilon} \leq \max_{y \in Y} \left( \frac{f(y) + \sum_{i=(k-y)_+}^{+\infty} e^{-\frac{1}{s}|i-\mu|} g(i)}{f_1(y) + \sum_{i=(k-y-1)_+}^{+\infty} e^{-\frac{1}{s}|i-\mu|} g(i+1)} \right)$$

where $Y = \{1,2,...,k\}$, $g(i) = \frac{(k!)^2 (i!)^2}{u!((k-y)!)^2 (i-k+y)!(k+i)!}$, $f(y) = e^{-\frac{1}{s}|y-k-\mu|} \mathbb{1}_{y \neq k} + \sum_{i=-\infty}^{-k-1} e^{-\frac{1}{s}|i-\mu|} \mathbb{1}_{y=0}$, $f_1(y) = \frac{e^{-\frac{1}{s}|y-k-1-\mu|}(k-y+1)}{k+1} + \frac{e^{-\frac{1}{s}|y-k-\mu|}(y+1)}{k+1} \mathbb{1}_{y \neq k} + \sum_{i=-\infty}^{-k-2} e^{-\frac{1}{s}|i-\mu|} \mathbb{1}_{y=0}$, and $s$ is the scale parameter. □

AKR chooses Laplace noise, which is similar to DGEO but in the continuous domain. AKR sets a positive bias $\mu$ so that the probability that the noise is negative is bounded by $\delta$, and the authors prove AKR follows $(\epsilon, \delta)$-DP. In order to have a small $\delta$ (i.e., the probability of failing DP to be small), $\mu$ must be fairly large which leads to unsatisfactory utility. For example, when $\delta$ equals a common value of $10^{-6}$, $\mu$ has to be *at least* 15 (it is even larger than the number of real requests and resources) to achieve $\epsilon = 1$ for $k = 10$.

Hence, accommodating negative noise without using a large bias is essential to high utility and we show *it is possible*. In a nutshell, negative noise may relax the pre-allocation $\epsilon$, but not necessarily introduces $\delta$. Though negative noise causes a discrepancy in the possible outcomes of $D$ and $D'$ from the attacker's view and in the range of $y$ (resources dispatched to attacker), which violates DP, together with non-negative noise they will not violate DP, as proved in Theorem 4.9 (i.e., the attacker's view satisifies DP). In Section 4.5.2, we provide empirical analysis to show the impact of RA on utility and privacy from the attacker's view.

## 4.5 Evaluation

In this section, we evaluate the privacy and utility of different mechanisms. Here we summarize the key results. 1) Our mechanisms outperform AKR by 11% to 65% in terms of utility (e.g., DGEO outperforms AKR by 53% given $\epsilon = 2$). GEO has a clear advantage for

smaller $\epsilon$ while DGEO is able to achieve better utility with larger $\epsilon$. 2) Different parameters can achieve similar privacy protection but lead to very different levels of utility.

### 4.5.1 Evaluation Setup

**Settings.** To compare different mechanisms in privacy-utility tradeoff, we choose to simulate RA using a real-world system setting. Similar to AKR, we take Alpenhorn [145], an MPM, as one of our target systems. In essence, a user in Alpenhorn starts a conversation with his/her friend on an agreed time or round. In a conversation round, the user initiates $k$ channels to $k$ friends, then sends and receives messages on each channel to hide the real communication pattern. Section 4.2.1 describes how its privacy guarantee can be violated. The evaluation by AKR models how Alpenhorn allocates channels for requests to defend against allocation-based side-channel attacks. Similar to AKR, we set the resource capacity $k=10$ for most of the experiments, meaning that a user has maximum of 10 channels that can be established with other clients. We also experiment with larger $k$ (15,20) to test the scalability of the proposed mechanisms and AKR. AKR sets an upper bound to the number of requests in each round and considers at most 10% of them to be honest requests. We remove the upper bound and set the number of victim requests to at most 1, to simulate the worst case for the victim, as explained in Section 4.2.1. Note that AKR uses a Poisson distribution to simulate the request number from all users while our total requests in case $D$ and $D'$ are fixed to $m$ and $m+1$, respectively. Given that we assume at most one victim exists during allocation, we did not apply the Poisson distribution. Unless otherwise stated, we simulate *10 million* independent rounds of allocation with requests of attacker $m=k$ (the optimal attacker strategy, as proved in Theorem 4.5), and measure privacy and utility.

In Section 4.6.5, we try to justify the the choice of simulation setup and discuss the limitation of simulation.

**Metrics.** We evaluate the performance of different mechanisms under three metrics: privacy, utility, and waiting overhead. Regarding privacy, we compute the empirical $\epsilon$ by Equation 4.8 with the simulation results and the larger value indicates more privacy leakage. Theoretical $\epsilon$ can be derived from Theorem 4.6 to Theorem 4.9, but their values are not always computable. For the study of parameters (Section 4.5.3), we compute some theoretical $\epsilon$ for the comparison.

As for the utility, we mainly measure the empirical *resource utilization* $U$, or how many (in ratio) resources are put into real use after allocation, from the simulation results. This differs from the classic DP that considers the *accuracy* of the analysis results as utility, or how close the noisy output is to the ground truth. The same utility measure is chosen by AKR as well. $U$ is given by:

$$U = \sum_{j=0}^{k} \mathsf{Pr}[r = j] \frac{j}{k} \tag{4.24}$$

where $r$ is the number of fulfilled requests, $k$ is the number of resources, and $\mathsf{Pr}[r = j]$ is the probability of $j$ requests being fulfilled.

While resource utilization is relevant to the overhead on the allocator, the overhead on the client can be measured by their waiting time (or waiting overhead). We use the probability of the victim getting the resource in any round, as the higher probability should lead to a shorter waiting time for the resource. For example, in Alpenhorn (original version that is not protected by DP) with $k$ resources and $m$ attacker requests, the probability that the victim gets resource $\mathsf{Pr}[V_a]$ is given by:

$$\mathsf{Pr}[V_a] = \frac{\binom{k-1}{m}\binom{1}{1}}{\binom{k}{m+1}} \tag{4.25}$$

Denoting the $\mathsf{Pr}[V_b]$ as the probability that the victim gets resources after DP, the ratio between $\mathsf{Pr}[V_a]$ and $\mathsf{Pr}[V_b]$ represents the amount of waiting overhead caused by a DP mech-

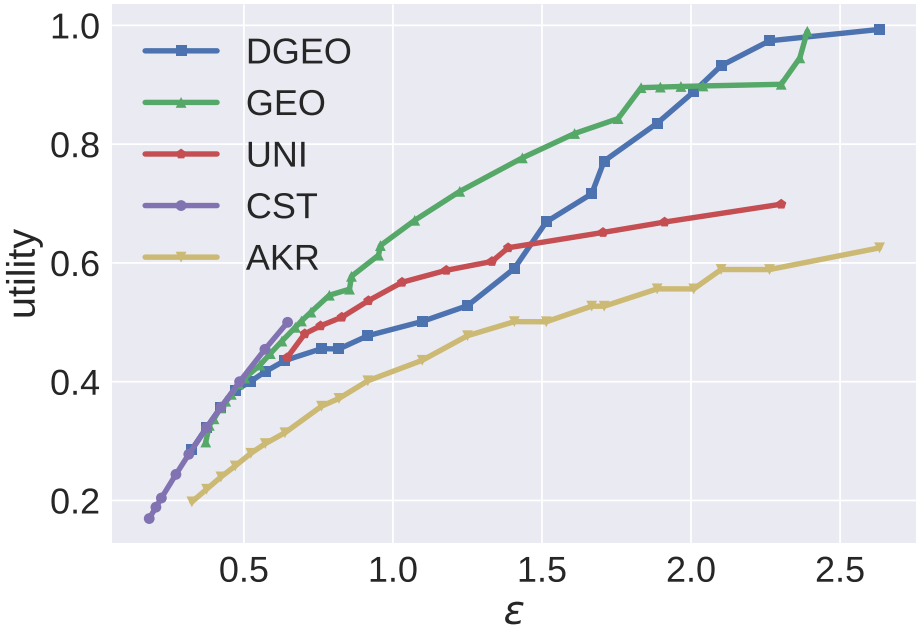



Figure 4.2: Comparison of different mechanisms. The ranges of $\epsilon$ for CST and UNI are limited. CST's utility never exceeds 0.5 because at least $k$ dummy requests are required to make it differentially private. The utility of GEO does not increase when $\epsilon$ is between 1.8 to 2.3, and we speculate this is because the parameters leading to the optimal utility have not been discovered through simulation.

anisms.

**Implementation.** We implement our code in Python 3.7.10 with NumPy 1.19.5 libraries. The implementations are open-sourced[6].

### 4.5.2 Evaluation Results

We compare the performance of different mechanisms, i.e., CST, UNI, GEO, DGEO, and AKR with simulation.

First, we enumerate different $\epsilon$ values for each mechanism and compute the *best* utility value, which is derived by searching in the space of possible mechanism parameters. Figure 4.2 illustrates the quantitative results of the tradeoff between privacy and utility. Note that, for AKR, since it is $(\epsilon,\delta)$-DP, we set $\delta = 10^{-6}$, which is commonly chosen by other DP works

[6]https://github.com/dpra-dp/dpra

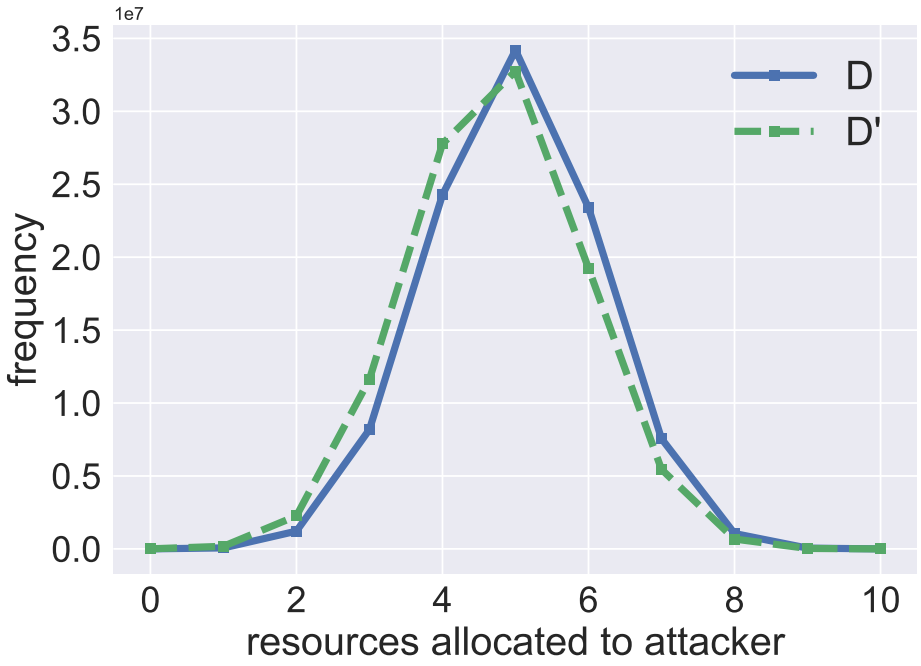



Figure 4.3: Allocation results by GEO with $p = 0.90$, which sets the bias to 10. The $x$-axis represents the number of fulfilled requests of the attacker, and the $y$-axis represents the frequency of each output out of 100 million rounds. We increase the simulation rounds from 10 million to 100 million in order to yield precise results.

(Angel et al. even chooses a larger value, $\delta = 10^{-4}$ [19]).

In general, we found that all of our proposed mechanisms have better utility than AKR for *every* $\epsilon$ when the parameters are fine-tuned. Specifically, GEO has better utility given lower $\epsilon$ (i.e., under 2) while DGEO yields better utility given more relaxed $\epsilon$ (i.e., over 2). AKR reaches the utility of 0.58 with $(2, 10^{-6})$-DP, while GEO and DGEO are able to achieve the utility of 0.89 with 2-DP, increasing the utility by 0.31 (53%). Overall, the margin of DGEO over AKR ranges from 0.05 to 0.39, GEO is able to outperform AKR over a range of 0.08 to 0.36, and UNI is able to outperform AKR by at most 0.15. This result is surprising as $(\epsilon, \delta)$-DP usually yields better utility than $\epsilon$-DP. We believe this is due to the fact that our mechanisms have the ability to accommodate negative noise, while AKR has to use a large bias to satisfy DP.

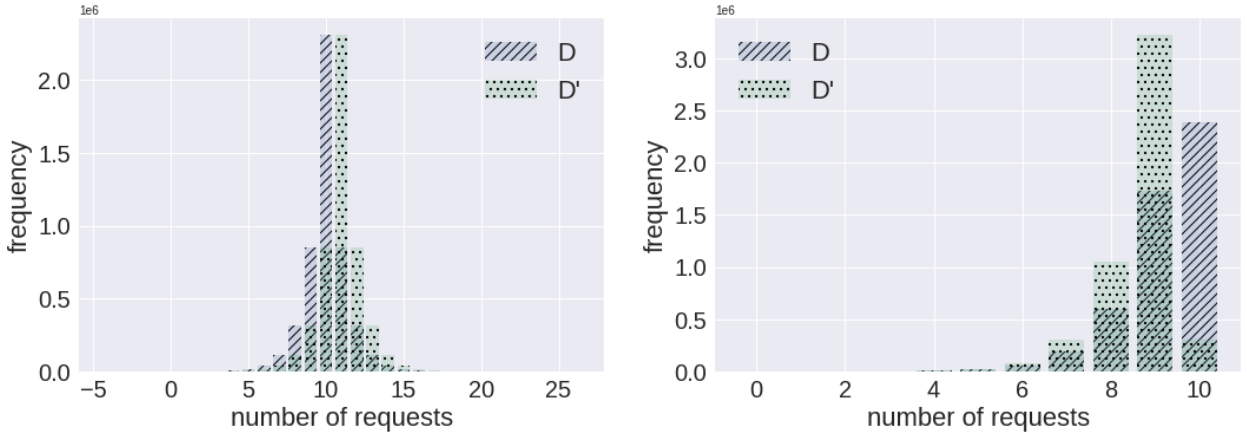
Since CST cannot achieve a utility value of more than 0.5, in the following experiments, we

focus on the other mechanisms. In the previous experiment, we change the mechanism parameters to fit $\epsilon$, but in the real-world deployment, the parameters are determined ahead. Here we evaluate the impact of the parameters related to bias. For DGEO and AKR, they are represented by $\mu$. For UNI and GEO, the starting point ($x_\ell$ for UNI and GEO) model the bias. We configure bias to a small value of 10. With the utility targeting 0.5, GEO and DGEO are able to bound privacy with $\epsilon$ of 0.80 and 0.77. UNI and AKR result in much higher $\epsilon$ at 1.28 and 1.5. Hence, with a small bias, our mechanisms can protect allocation with better privacy while achieving the same utility as AKR.

So far, the prior experiments quantitatively measure how the mechanisms perform. Like Angel et al. [19], we visualize privacy protection under a fixed set of parameters. Specifically, we measure the difference in allocation results ($D$ and $D'$) based on the number of resources allocated to the attacker. Figure 4.3 shows the visualization of GEO, when bias is configured to 10. The lines of $D$ and $D'$ stay close, suggesting the privacy leakage of GEO is small.

Regarding the waiting overhead $\mathsf{Pr}[V_a]$, we found UNI, GEO, DGEO and AKR reach 1.45, 1.92, 1.91 and 1.88 when configuring bias to 10, suggesting our mechanisms either have similar or lower waiting overhead than AKR. Still, we acknowledge that such overhead is significant and we discuss this issue in Section 4.6.5.

**Why Models Attacker's View.** In Section 4.4.4, we argue that modeling the attacker view is better than modeling the whole view that is adopted by AKR. Here we justify this claim under the same simulation. Figure 4.4 shows an example with the zero-mean ($\mu=0$) double-geometric distribution under simulation. Given two different cases $D$ and $D'$, Figure 4.4a depicts the difference of output before allocation and Figure 4.4b shows the output after allocation from the attacker's view, assuming DGEO with scale 1 is applied to allocate $k=10$ resources, and $D$ contains $m=10$ requests. Our study shows that the existence of the victim can drastically affect the portion of resources an attacker can get after allocation.




(a) Before RA. (b) After RA.

Figure 4.4: Distribution of output over 5 million runs. Before RA, we draw noise from a double geometric distribution with $\epsilon = 1$ and $k = 10$. After RA, the distribution changes, and the privacy leakage increases (the empirical $\epsilon$ rises to 2.07).

| Original $\epsilon$ used | 4 | 2 | 1 | 0.5 | 0.2 |
|---|---|---|---|---|---|
| Empirical $\epsilon$ before RA | 4.00 | 2.00 | 1.03 | 0.54 | 0.27 |
| Empirical $\epsilon$ after RA | 3.28 | 2.26 | 2.01 | 1.90 | 1.79 |
| Theoretical bound of $\epsilon$ after RA | 3.29 | 2.26 | 2.07 | 1.91 | 1.79 |

Table 4.3: Comparison of different settings of DGEO with $k = 10$. We use 5 different $\epsilon$ values (first row). Row 2 shows the empirical $\epsilon$ is close to the original $\epsilon$, which indicates our simulation has only small errors. Row 3 is the empirical $\epsilon$ after RA, which deviates from the original $\epsilon$. The last row shows our theoretical bound of $\epsilon$ given in Theorem 4.9 is close to the empirical value.

Table 4.3 uses four different zero-mean double geometric distributions to further explain why RA itself should be part of the privacy modeling. First, when the original $\epsilon$ decreases, more noise is expected, which leads to an increase in privacy protection for both before and after RA. However, given a relatively high scale (i.e., small $\epsilon$), the privacy protection after RA can be 6 times worse than that before RA. Such extra information leakage is an indicator that the privacy budget is affected by RA.

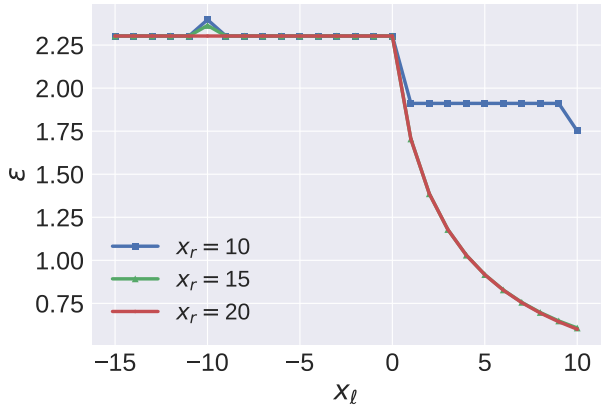
We also take a step forward to measure the privacy amplification caused by modeling the attacker's view. We adjust AKR by replacing its Laplace noise with double geometric noise, which we denote as AKR-DGEO, and compare it with DGEO. As their noise mechanisms

become the same, we can exemplifies the privacy-utility tradeoff without and with privacy amplification. Our empirical analyses indicate that, for a utility measure of approximately 0.42, privacy amplification results in a decrease of the privacy parameter $\epsilon$ from 1.00 to 0.59. Likewise, when the utility measure is near 0.60, $\epsilon$ diminishes from 2.00 to 1.43 after amplification.

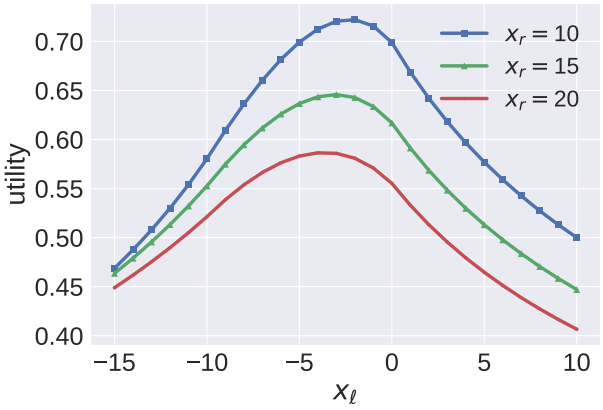
### 4.5.3 Impact of Parameters

**Impact of Parameters.** To assess the impact of mechanism parameters, we compute the privacy and utility values theoretically, as explained in Section 4.5.1. Here we first discuss the guidelines for setting parameters before diving into the details. For UNI, one should avoid large $x_r$ as the privacy benefit diminishes and utility drops noticeably. Regarding $x_\ell$, we found negative values do not offer good privacy and small $x_\ell$ is necessary to maintain good privacy. For GEO, a negative starting point $x_\ell$ should be avoided as it does no good to utility or privacy. We suggest that a small positive starting point $x_\ell$ with a moderately high $p$ value would be optimal for GEO. For example, an $x_\ell$ of 3 with $p = 0.7$ can achieve reasonable privacy $\epsilon = 1.24$ and a good utility of 0.75. Our evaluation in Section 4.5.3 also indicates that a small positive bias $\mu$ with a scale $s$ around 1 would be optimal for DGEO. For the larger resource capacity $k$, GEO and DGEO still perform well.

**Starting Point $x_\ell$ and End Point $x_r$ of UNI.** In Figure 4.5, we display privacy and utility across various $x_r$ ($x_r = 10,15,20$) and $x_\ell$ values (along the $x$-axis). Notably, $x_r = 15$ largely mirrors $x_r = 20$ in terms of $\epsilon$, even though $x_r = 20$ is expected to offer superior privacy. Regarding utility, $x_r = 10$ consistently ranks highest for different $x_\ell$, followed by $x_r = 15$ and $x_r = 20$. Regarding $x_\ell$, increasing its value enhances privacy (resulting in a lower $\epsilon$), with utility peaking when $x_\ell$ ranges between $[-5,0]$. However, we observe two outliers related to $x_\ell$ in Figure 4.5a. First, a peak is observed when $x_\ell = -10$, because all requests in $D$ are removed deterministically but the probability of the same situation for $D'$ is $\frac{1}{k+1}$, where

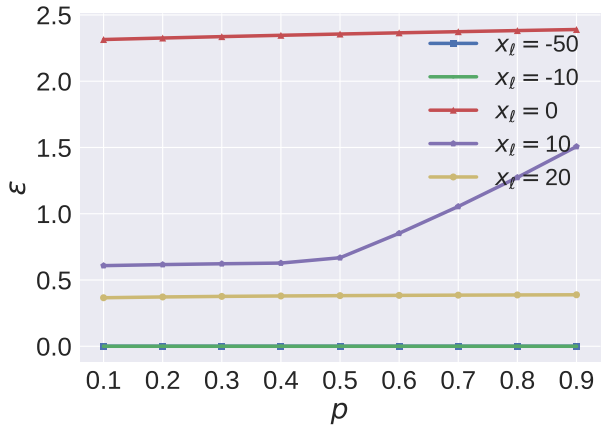



(a) $\epsilon$ of UNI given different $x_\ell$ and $x_r$.

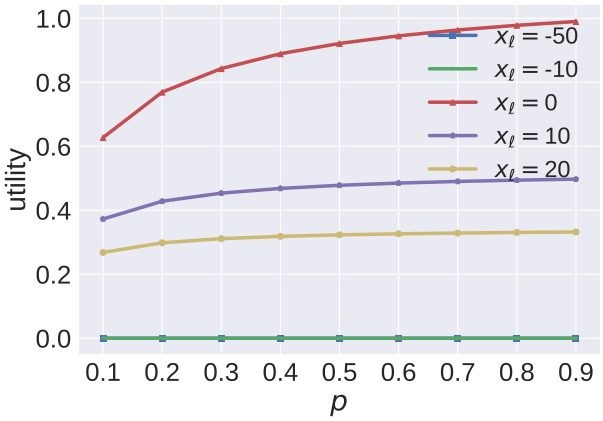



(b) Utility of UNI given different $x_\ell$ and $x_r$.

Figure 4.5: Impact of $x_\ell$ and $x_r$ on UNI.




(a) $\epsilon$ of GEO given different $p$ and $x_\ell$.

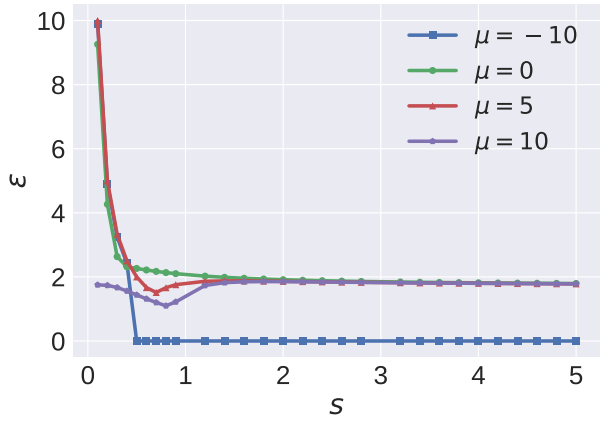



(b) Utility of GEO given different $p$ and $x_\ell$.

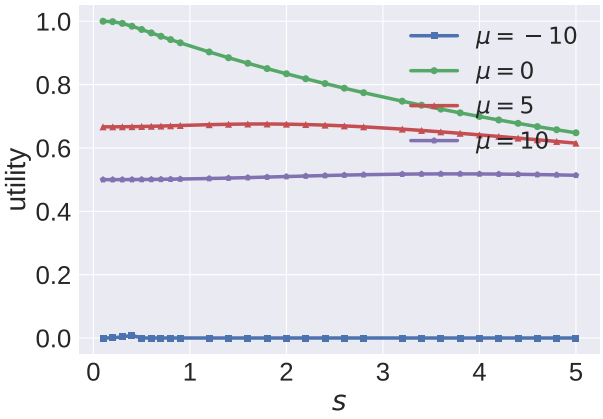
Figure 4.6: Impact of $p$ and $x_\ell$ on GEO.

victim exists. Second, when $x_\ell = x_r = 10$, $\epsilon$ drops to 1.75 because this special case implies that attacker gets no resource in victim's absence.

**Geometric Parameter $p$ and Starting Point $x_\ell$ of GEO.** Figure 4.6 depicts how $p$ and $x_\ell$ affect GEO. For $x_\ell = -50$ and $x_\ell = -10$, both $\epsilon$ and utility approach 0 due to the high likelihood of request removal. At $x_\ell = 0$, utility is high but $\epsilon$ consistently exceeds 2. For $x_\ell = 10, 20$, $\epsilon$ is below 1.5, with utility rising as $x_\ell$ increases. For $p$, its influence on $\epsilon$ is minimal, except at $x_\ell = 10$ where $\epsilon$ increases sharply after $p = 0.5$. Utility consistently grows with $p$ across all settings.

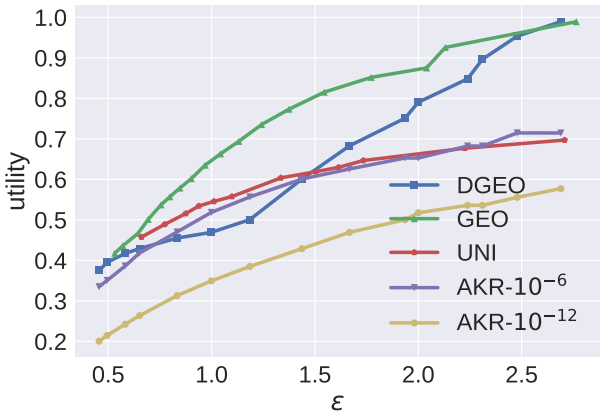



(a) $\epsilon$ of DGEO given different scale $s$ and bias $\mu$.

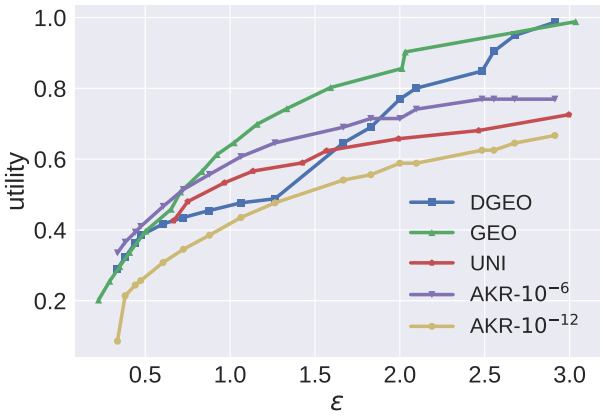



(b) Utility of DGEO given different scale $s$ and bias $\mu$.

Figure 4.7: Impact of $s$ and $\mu$ on DGEO.




(a) Comparison of mechanisms under $k$=15.




(b) Comparison of mechanisms under $k$=20.

Figure 4.8: Privacy protection and utility under $k = 15,20$. The ranges for the x-axis differ for $k$ because not all utility values can be derived under every $\epsilon$.

**Geometric Scale $s$ and Bias $\mu$ of DGEO.** In DGEO, the scale parameter $s$ determines the noise's decay rate. A smaller $s$ results in noise more closely concentrated around the bias $\mu$. $\mu$ introduces more noise to the allocation, impacting post-allocation privacy. We evaluate the influence of these parameters on privacy and utility, presenting the findings in Figure 4.7. Introducing bias $\mu$ improves privacy, especially when $s < 1$. For larger $s$, the distribution resembles a discrete uniform, keeping $\epsilon$ stable (around 2 for $\mu \geq 0$). $s$ has limited utility impact unless $\mu = 0$.

**Resource Capacity $k$.** We set $k$ to 10 for the prior experiments like Angel et al. [19]. Here we test our mechanisms and AKR on $k=15,20$. Figure 4.8 shows the privacy-utility tradeoff. For AKR, besides the default $\delta=10^{-6}$, we also evaluate $\delta=10^{-12}$, bringing its privacy closer to $\epsilon$-DP. Figure 4.8 illustrates that $\delta$ significantly impacts AKR's utility, with average gaps of 0.2 for $k=15$ and 0.1 for $k=20$. GEO and DGEO still perform well for these new $k$ values and better than AKR.

## 4.6 Discussion

### 4.6.1 Related Work

**Joint DP.** We focus on the partial view of the attacker. The Joint DP definition proposed by Kearns et al. [127] formalizes this intuition, primarily to compute equilibrium in games with incomplete information [127, 196, 197]. Note that Joint DP is just a definition, and classic DP primitives like Laplace mechanism are still used. We are the first to formally investigate the design space and adapt various DP mechanisms to RA.

**Private Matching and Allocation.** Our problem can be seen as a variation of the private allocation/matching problem, through which users have (non-binary) valuations for products (potentially in multiple rounds), and the goal is to maximize welfare while protecting users' private value for each good. Existing works [32, 124, 176, 125, 55, 107] have applied DP algorithms (e.g., Laplace mechanism) that are asymptotically interesting. Our modeling of RA is different and we explored different noisy mechanisms.

**Biased Noise.** AKR employs biased noise to satisfy DP, while DGEO uses it to improve the privacy-utility tradeoff. Biased noise has been examined before. Mazloom and Gordon [166] introduced a modified 2-sided geometric distribution to generate noise that enables differentially private access patterns with high efficiency. DJoin [177] cuts Laplace noise at zero to

provide distributed queries with DP. Shrinkwrap [25] offers a truncated Laplace mechanism for differentially private data federation, where dummies are introduced to pad intermediate results. He et al. [97] proposes a model for private record linkage, allowing the disclosure of the true matching records while keeping the protocol executions indistinguishable when non-matching records are replaced.

**DP Against Side-channel Leakage.** The leakage from RA can be considered as allocation-based side channels [19]. A more common type of side channels is consumption-based, which happens when the system resources (e.g., network bandwidth and cache) are consumed. A number of works have applied DP to protect the system against the latter type of leakage. The protected resources/services include `procfs` of system statistics [242], streaming traffic [248], Trusted Execution Environment (TEE) [243], health data (e.g., ECG data) [201], task schedules [43], and packet scheduler [26].

Another related line of work is differentially oblivious [40], which was proposed to address the fundamental limitation of ORAM (Oblivious RAM). Though ORAM can protect the program's secret by hiding its memory access pattern, it incurs very high performance overhead. By converting full obliviousness to differential obliviousness, one can obtain meaningful privacy with little overhead [40, 231, 136]. While this paper also hides a victim's secret (i.e., its existence at certain time), it considers an orthogonal adversary model where the attacker observes part of the true results without any mechanism to hide the victim-related information.

### 4.6.2 Privacy Consumption over Multiple Rounds

Like Angel et al. [19], our analysis focuses on a single round. Privacy normally degrades over multiple rounds rapidly. For instance, naively applying the sequential composition property of DP over multiple rounds deteriorates the privacy guarantee (i.e., $\epsilon$) linearly. Inspired by

previous work, we identify three ways to curb the privacy consumption: (1) using advanced composition [172] to reduce the total $\epsilon$, (2) reusing noise for repeated requests [223, 69], and (3) bounding the number of requests. Though relaxations could happen for attacker's background knowledge [58], our approach does not limit the attacker's background knowledge but rather their view, and therefore we believe composition works in our case. Next, we discuss how the three methods can be applied in more detail.

**Using Advanced Composition.** Traditional composition theorem in DP may result in a union bound over noise, which is sub-optimal. Avoiding union bound for multiple queries has been an important open problem in differential privacy [218]. The well-known advanced composition theorem [63] adjusts pure DP to approximate $(\epsilon,\delta)$-DP with $\delta>0$ to yield better composition results. In cases where the attacker interacts with the allocator over multiple rounds, we argue that the leakage can be modeled by the $k$-fold adaptive composition [63].

Mironov [172] proposed new bounding techniques for advanced composition under Rényi DP (RDP) to this end. In our case, we can transform $\epsilon$-DP to $(\alpha,\epsilon)$-RDP for any $\alpha>0$ [172], compose RDP with Theorem 4.1, and transform back to $(\epsilon,\delta)$-DP with Theorem 4.2. Popular DP libraries like Opacus have supported RDP advanced composition [191]. Alternatively, we can utilize Equations 4 and 5 in [182] to derive the $(\epsilon, \delta)$-DP bound directly and employ numerical methods [82] to obtain more accurate results.

Yet, it is an open question to directly prove the RDP guarantee for our mechanisms (to avoid conversions mentioned above and compose better). One possible route is to follow the proof of the discrete Gaussian mechanism [36] and we leave it as a future work.

**ReusingNoise.** When new incoming requests are from the same set of clients of the previous round, the server can avoid consuming extra privacy budget by reusing the noise generated for the previous round [223, 69]. In this way, the attacker gains no more information than the previous round while the server consumes no extra budget. Specifically, the output of the

algorithm remains the same if we fix the randomness that happens in a certain round. Thus, the server can utilize a persistent secret key for a pseudo-random function (PRF) over the same set of clients, where in each round the server is able to simulate the same randomness for the same set of clients.

**Bounding the Number of Requests.** Drawing from [66], we can simplify the privacy analysis by eliminating the need to consider every RA round for each client by capping client requests over a period (e.g., a maximum of 2 calls daily for MPM clients).

### 4.6.3 Other Settings

Though our study primarily examines clients submitting binary requests for a single resource under worst-case privacy, it can be extended to (1) the non-binary setting that clients can submit requests for more than one resource, (2) the multi-resource setting that there are multiple kinds of resources and clients can request arbitrary resources, and (3) the average-case privacy.

**Non-binary Requests that *Can* be Fulfilled Partially.** This setting can be transformed to the binary case by casting each non-binary request as multiple binary requests. The global sensitive will be changed to the number of maximum requests per client.

**Non-binary Requests that *Cannot* be Fulfilled Partially.** The problem is transformed into an optimization problem aiming for maximum utilization of resources [127]. In general, the allocator picks the requests that maximize its target function. The allocator can add noise to the number of requests, which we expect to yield worse utility compared to our primary setting. This is because when requests for large resources are added or removed from the allocator, a great amount of resources are wasted.

**Multiple-resources Allocation.** A multiple-resource allocator deals with multiple types

of resources simultaneously. In this setting, the privacy protection of the allocator subjects to sequential composition, thus the overall privacy depends on the summation of all privacy losses. The intuition is that the privacy leakage of each allocation can be seen as auxiliary information, and be combined with leakage from allocations of other types of resources.

**Multiple Honest Requests.** Multiple honest requests in allocation happens when the attacker is not strong enough to control all other clients except the victim. Assume the requests are binary in this setting and the attacker does not know the resource distribution among the honest requests. In this case, the honest requests (other than the one from the victim) are equivalent to the dummy requests in our primary setting because the distribution among them remains unknown to the attacker. Therefore, we can add less noises in this setting in order to achieve the same privacy guarantee. We have justified the above assumption by experimenting with DGEO (the results are omitted due to page limit).

### 4.6.4 Real-world Examples and Utility Analysis

Here we first give a few examples of how the noise under $d \geq 0$ and $d < 0$ can be instantiated in real-world systems. We follow the basic setting as described in Section 4.2.1 first (i.e., all resources are identical and one request asks for one piece of resource).

- In the cloud setting, users request for VMs and whether they are served is based on the available resources like CPU and memory. When $d > 0$, the allocator creates dummy VMs that potentially occupy resources. When $d < 0$, not all the requested resources are allocated to the VM (even though there are available resources).
- Inside a computer, requests to cache resources (e.g., cache ways) are automatically generated during a memory access, which can lead to cache side-channel attacks [246]. $d > 0$ will assign cache ways to dummy programs and $d < 0$ will skip the caching of some memory content. Either option will reduce the accuracy of the attack which relies on

cache contention between attacker and victim.

- In MPM, the requests are from a user's friends who intend to start a conversation in a round. Noise $d>0$ is to add fake friends and $d<0$ means to reject some requests.

For more complex allocators, we can extend the DP mechanisms following Section 4.6.3. For example, the buddy system manages memory in power of two increments [134] and we can support it by considering the memory requests as non-binary. When concurrent requests are supported by multiple resource pools (e.g., hypervisor resource pools [230]), multiple-resources allocation can be applied.

Regarding the results about privacy-utility tradeoff (e.g., summarized in Table 4.2), we argue they are practical in the real-world setting. For example, a study of Google Cloud shows the resource utilization is 40% - 60% and the resource waste due to early task termination is 4.53 - 14.22% [78]. In this case, the utility after DGEO and GEO should be acceptable (e.g., 0.82 for GEO at $\epsilon=1.7$).

### 4.6.5 Limitations

**Empirical Study on Privacy.** The privacy analysis in our evaluation is empirical-based (i.e., $\epsilon$'s are calculated empirically based on our simulation result). We choose simulation for two main reasons. First, we aim to compare the privacy-utility tradeoff of different mechanisms at different privacy parameters (e.g., Figure 4.2) and the computational overhead will be very high if the experiments are executed on the large-scale real-world systems. Second, for the MPM system we evaluate, there is no published dataset about its communication data, so we have to simulate the allocations. In fact, Angel et al. took a similar approach to evaluate privacy empirically [19], and the scale of our simulation is comparable or larger (from 5 million rounds to 100 million rounds). Simulation has been leveraged to evaluate other privacy-preserving systems for the same reason, like differentially oblivious database [192].

We also acknowledge the limitation of our simulation, which does not fully approximate real-world, large-scale systems.

**Efficiency.** Adding dummies results in higher waiting overhead because the clients now need to go through more rounds in order to get the desired resources. However, once the resources are allocated, no additional delay should be observed.

The spatial overhead due to serving the dummy clients could be prominent, especially for systems that operate on very limited resources. The same limitation exists in AKR, and the overhead is often unavoidable for systems leveraging DP. On the other hand, our approach provides better resource utilization than AKR, e.g., 98% under DGEO and 59% under AKR when $\epsilon = 2.3$. Higher resource utilization also leads to smaller waiting overhead. For example, for an approach with 40% utilization, the chances for a user to get resource allocated within 5 dialing rounds in Alpenhorn is about 99%. Our proposed mechanisms all surpass 40% as shown in Table 4.2.

**Attacks against DPRA.** Potential side-channel attacks against DP algorithms, such as timing attacks [122], may compromise our DPRA, but require adaptation to the RA setting.

# Chapter 5

# Privacy Risks in Curriculum Learning and DP Defenses

## 5.1 Introduction

Key to the success of machine learning (ML), especially deep learning (DL), is the advancement of algorithms, software, and hardware in training models on large-scale datasets. The traditional way to train a neural network (NN) is by feeding the training pipeline with random mini-batches in a sequence sampled from the training dataset. In other words, NN is forced to "remember" samples repeatedly in random order. On the other hand, human always learns the easy concepts first and then the hard ones, as guided by curricula. Given that NN is inspired by the human brain [199], curriculum learning (CL), which simulates human learning by ordering the training data with difficulty scores and repeating the order across training epochs, has been proposed [28]. With a "teacher" network, the difficult scores can be generated ahead of the samples and guide the training process. Previous studies have shown that CL can achieve both fast learning speed and high test accuracy [215, 236], and CL has been adopted in many application domains like computer vision [28, 200, 59, 214],

natural language processing [28, 216, 254, 87, 155], and claiming prominent success [236].

Despite the huge success of ML, the privacy issues of ML are becoming more and more concerning, given that the training data could contain a large amount of sensitive information. The two most notable privacy attacks are membership inference attack (MIA) [118, 207] and attribute inference attack (AIA) [212], where MIA aims to infer whether a given data sample is used to train the target model and AIA aims to infer the sensitive attribute of a data sample. Numerous attacks have emerged and demonstrated that privacy threats are real (e.g., over 80% MIA accuracy against CIFAR100 [203]). Recent studies have also shown the data samples are not equally vulnerable under privacy attacks [244], and the attack effectiveness could differ across target classes [118], target individuals [159], and subgroups [42]. Yet, all previous works assume standard, stochastic training is employed by the target model. Hence, one interesting and important research problem is how new training techniques impact privacy for the overall population and individual samples. In this work, we specifically study the privacy risks of CL. We are particularly motivated because CL modifies the data order, which differs from many new techniques such as contrastive learning [99] and other self-supervised learning techniques [156]. In general, CL increases a model's overall performance and lets a model treat samples differently based on their difficulty levels[1] Furthermore, Shumailov et al. [208] studied the connection between data ordering and backdoor attacks, which indicates data ordering could have negative impacts. This further motivates us to investigate the privacy risks of CL.

**Our Study.** We take a quantitative approach to measure the privacy risks of CL. We selected two popular CL methods, bootstrapping [89] and transfer learning [239], as the evaluation objects, and constructed two other curriculum, named baseline curriculum and anti-curriculum, to understand the impact of data ordering and repeating, respectively. We selected 9 real-world, large-scale datasets (6 are image datasets and 3 are tabular datasets),

[1]The terms "difficulty level" and "difficulty score" are interchangeable.

trained target models with those CL methods and a normal method, and attacked the models with representative MIA and AIA methods.

Regarding MIA, our evaluation shows that the target models become slightly more vulnerable under CL, e.g., the average attack accuracy (trained on ResNet-18 with transfer leaning) on our selected image datasets ranges from 0.01% to 2.46%. More importantly, we found CL has a much bigger impact on the samples within the difficult group compared to the easy group, with the biggest gap of 4.23% in terms of attack accuracy for CIFAR100 (ResNet-18 is the architecture). This observation sustains both image and non-image datasets. We found the reason is that the data order reinforces the learning process hence making the model memorize difficult samples better, which is supported by measuring the memorization scores. Regarding AIA, we found CL does not increase the attack accuracy, which can be explained by the fact that the sensitive attribute to be inferred is not influenced by data ordering and repeating.

In addition to understanding the attacks, we also study existing defenses under the CL settings, including DP-SGD [13], MemGuard [118], MixupMMD [149] and AdvReg [180]. The result shows that none of them can mitigate the threats from MIA without dampening the model accuracy. In particular, DP-SGD is effective in curbing MIA, and the drop of attack accuracy for the CL setting is even more than the normal setting. However, the privacy provided by DP-SGD is at the cost of dropping the classification accuracy of the target model.

Inspired by CL and a recent MIA that calibrates membership scores to achieve better attack accuracy [238], we consider the difficulty score as input for calibration and proposed a new MIA method, named Diff-Cali (difficulty calibrated MIA) . Our attack cannot only bring the difficult samples to a more vulnerable stage but also achieves a higher true-positive rate at low false-positive rate regions.

**Contributions.** The contributions of this work are summarized below.

- We take the first step to understanding the privacy risks introduced by CL.
- We conduct a comprehensive analysis to quantify the privacy risks and our results show CL introduces disparate impacts to samples separated by difficulty levels.
- We propose a new MIA that exploits the difficulty scores for better attack performance.

## 5.2 Preliminary

### 5.2.1 Curriculum Learning

Curriculum learning (CL) [28] is designed to emulate the concept of the human learning process. The general idea is to have a *curriculum* that imposes a structure on the training data so the "student" ML models can learn from the easier samples before the harder ones. As a result, training ML models under CL observes a shorter duration of convergence and higher testing accuracy[28, 239, 84, 89]. For example, Weinshall et al. proposed to use transfer learning to build the curriculum and achieved 0.5% to 3.5% higher accuracy than a model trained in the normal setting [239]. CL has gained significant interest from the ML community, powering real-world applications in many domains. Section 5.7 provides a more detailed survey.

Below, we formalize CL following the definition of Hacohen et al. [89]. Let $\mathcal{X} = \{X_i\}_{i=1}^{N} = \{(x_i,y_i)\}_{i=1}^{N}$ be the training dataset, where $N$ is the number of samples, $x_i$ is a data point, and $y_i$ is the label of $x_i$. $T$ is the ML model to be trained. Assuming Stochastic Gradient Descent (SGD) is used for optimization, and each training iteration takes a mini-batch of $\mathcal{X}$, and a sequence of $M$ mini-batches $\mathcal{B}_1,...,\mathcal{B}_M$ will be used for each epoch. The standard training procedure will sample $\mathcal{X}$ uniformly to generate the mini-batches. Instead, CL uses

a *difficulty measurer* $f(\mathcal{X},C)$ to generate difficult scores for $\mathcal{X}$, and a *training scheduler* sorts $\mathcal{X}$ by the difficult scores in an ascending order ahead of training. $C$ is the curriculum, and we will elaborate on its common options in Section 5.4.1. A sequence of subsets $\mathcal{X}'_1,...,\mathcal{X}'_M \subseteq \mathcal{X}$ are extracted from $\mathcal{X}$ after sorting, and the size of $\mathcal{X}'_i$ is determined by a *pacing function* $g(i)$. A mini-batch $\mathcal{B}_i$ is sampled uniformly from $\mathcal{X}'_i$. Algorithm 1 summarizes the process. Noticeably, slight changes can be applied (e.g., skip the step of mini-batch sampling), but they should not affect the conclusions drawn from this study.

**Algorithm 1:** Curriculum learning framework.

**Input:** Training dataset $\mathcal{X}=\{\mathbf{X_i}\}_{\mathbf{i=1}}^{\mathbf{N}}$, difficulty measurer $\mathbf{f}(\mathcal{X},\mathbf{C})$, pacing function $\mathbf{g(i)}$, number of iterations $\mathbf{M}$, number of epochs $\mathbf{E}$, target model $\mathbf{T}$

$\mathcal{X} \leftarrow f(\mathcal{X},C)$;
**for** $e \in 1,...,E$ **do**
  **for** $i \in 1,...,M$ **do**
    $\mathcal{X}'_i \leftarrow \mathcal{X}[1,...,g(i)]$;
    $\mathcal{B}_i \leftarrow sample(\mathcal{X}'_i)$;
    $T \leftarrow train(T,\mathcal{B}_i)$

### 5.2.2 Privacy Risks in Machine Learning

Prior works have shown that the ML models could memorize sensitive information from the training data, which can be inferred by an adversary who keeps querying the model. Two major types of attacks are MIA [207, 180, 181, 203] and AIA [168, 212], which have been extensively studied. The detailed literature survey of privacy attacks and other attacks is left to Section 5.7. **Membership Inference Attack (MIA).** Given a target model $T$ and any adversary's external knowledge $K$, the goal of MIA is to determine whether a data sample $x$ was used to train the model. Formally, we have:

$$\mathcal{A}_{MI}: x,T,K \mapsto 1 \text{ or } 0 \tag{5.1}$$

where $T$ is the target model and $K$ is the adversary's external knowledge, e.g., the distribution of the training data for $T$. 1 (0) denotes the sample is a member (non-member).

MIA can lead to serious privacy threats. For example, given a model trained on clinical records of cancer patients to determine the medicine dosage [118], the attacker can learn whether a person has cancer by applying MIA to the model. We follow previous work [207, 203, 213, 151, 53] and assume that the adversary only has black-box access to $T$, which means that the adversary can only query $T$ with the data sample and obtain its corresponding output. Then, $\mathcal{A}_{MI}$ predicts membership with the output of $T$. Section 5.4.2 elaborates the details.

**Attribute Inference Attack (AIA).** Different from MIA, the goal of AIA is to infer attributes of a data sample that are not related to the target model's original classification task. A specific attack scenario is when AIA is used to infer some hidden sensitive attributes. For instance, a target model is trained to conduct gender classification, while AIA aims to infer the political view of a data sample. Such attribute is often hidden when training the target model. However, due to the intrinsic *overlearning* property of ML [212], a target model may try to capture attributes not directly relevant to its task. Note that AIA is different from property inference attack (PIA) [76] which infers a property about the entire dataset rather than a sample: e.g., PIA can tell whether a training dataset is gender-balanced.

Instead of having direct access to the sample, we follow previous work [168, 212] and consider the adversary only has its *representation* (e.g., embedding) generated by a target model $T$. Formally, AIA can be defined as:

$$\mathcal{A}_{AI} : h \mapsto s \tag{5.2}$$

where $h$ is a sample's representation provided by $T$ and $s$ is the sample's sensitive attribute predicted by $\mathcal{A}_{AI}$. Section 5.4.4 elaborates the details.

## 5.3 Datasets and Target Models

In this work, we aim to quantify the privacy risks introduced by CL through the lens of MIA and AIA. To this end, we select popular datasets and models that are used for ML classification tasks. In our study, a total of 9 unique datasets are used, with 8 datasets used for MIAs and 3 datasets used for AIA. Among these datasets, 6 of them are image datasets, while the remaining 3 datasets consisted of non-image data.

**MIA Datasets.** We use the following 8 datasets for MIA evaluation, which are also adopted by previous work [99, 154, 171, 207] to study MIA. They are CIFAR100 [139], Tiny ImageNet [146], Place100, Place 60[253], SVHN [183], Purchase[207], Texas hospital stays[207] and Locations [245]. We focus on image datasets mainly (the first 5 datasets), but tabular datasets are also evaluated. Below are the detailed descriptions for the datasets.

- **CIFAR100 [139].** This dataset consists of 60,000 colored images in 100 classes, with 600 images per class. The size of each image is $32 \times 32$.
- **Tiny ImageNet [146].** This is a subset of the ImageNet dataset[56]. It contains 100,000 colored images of 200 classes (500 for each class). The size of each image is $64 \times 64$.
- **Place100.** This dataset is a subset of Places365[253] dataset, which is composed of more than 1.8 million images with 365 scene categories. Place100 is generated by randomly selecting 100 scene categories with 600 random images per category.
- **Place60.** This dataset is similar to Place100, except that it has 60 classes containing 1,000 images each.
- **SVHN [183].** The Street View House Numbers (SVHN) dataset is a real-world image dataset containing over 600,000 digit images. This dataset includes images of house

numbers taken from Google Street View images. The size of each image is $32 \times 32$.

- **Purchase.** This is a tabular dataset about purchase styles. Following Shokri et al. [207], we leverage the Purchase-100 dataset (abbreviated as Purchase) and use $10,000$ records for training. The dataset itself contains $197,324$ records from 100 classes, where each record has 600 binary features.

- **Texas hospital stays.** This dataset contains information about inpatient stays in several health facilities. Following Shokri et al. [207], our task is to predict a patient's main procedure. After pre-processing, the resulting dataset has $67,330$ records and 6,170 binary features.

- **Locations [245] .** The original dataset was released by Foursquare about its mobile users' location "check-ins", which has 11,592 users and 1,136,481 check-in records. Our task is to predict the user's geo-social type (128 in total). Here we use the version pre-processed by Shokri et al. [207], which has 446 binary features.

**AIA Datasets.** Datasets with multiple attributes are required for AIA. To this end, we adapt Place100 and Place60 used as MIA datasets to AIA setting as they both contain multiple attribute labels. More specifically, the model for Place100 outputs whether a sample is an indoor scene, while the sensitive attribute is the category of the scene, which contains 100 labels. Place60 has the total number of categories as 60. In addition to Place100 and Place60, we introduce UTKFace [250] specifically for AIA study.

- **UTKFace [250].** This is a large-scale facial dataset, which consists of over $20,000$ face images with annotations of age, gender, and ethnicity. In our evaluation, we set gender classification as the the task for target model, and the sensitive attribute to be inferred is ethnicity, which includes 5 classes.

**Target Models.** We adopt three popular neural network architectures as the target model's

architecture for the image datasets. They are ResNet-18 [95], ResNet-34 [95] and MobileNet [204]. We choose these models because variants of ResNet are still achieving SOTA (State of The Art) or near SOTA performance in image classification, and MobileNet is widely used on mobile devices. We adopt cross entropy as the loss function and SGD as the optimizer. We train all models for 200 epochs with a batch size of 128. The learning rate is set to 0.1[2]. For the non-image dataset Purchase and Location, we choose a 3-layer MLP with the same number of epochs and batch size. The number of neurons in the hidden layer is 256. For the Texas dataset, we use a 5-layer MLP with 512 neurons in the hidden layer because this dataset contains more features. To avoid fortuitous outcomes, all experiments are repeated 5 times with the standard deviation presented.

## 5.4 Methodology

In this section, we describe the curriculum designs experimented with by our study, the implementation of the basic MIA and AIA, our proposed MIA, and the defense techniques to be tested.

### 5.4.1 Curriculum Designs

We choose two popular curriculum learning (CL) methods, which also have open-sourced implementations [88, 221], to train the target model. We expect our major observations (described in Section 5.5) are also applicable to other CL methods, like self-paced curriculum [140, 120], and automated curriculum [84], because they share similar high-level ideas (e.g., self-paced curriculum differs from bootstrapping only in that self-paced curriculum does not let the curriculum completely guide its learning process). Below we explain the two CL methods.

[2]This learning rate is empirically chosen and has a very limited effect on attack accuracy. For example, when using a learning rate of 0.001, the MIA accuracy is affected by less than 0.2% when attacking a ResNet-18 model trained on CIFAR100.

- **Bootstrapping [89].** The target model $T$ is first trained without CL, then it serves as a difficulty measurer ($f$ in Algorithm 1) to order the training samples by their loss.
- **Transfer learning [239].** Different from bootstrapping, a pre-trained model is used for the difficulty measurer. We use inception-v3 [222][3] as the pre-trained model to evaluate the image datasets. The evaluation on tabular datasets with transfer learning is skipped, as we did not find a widely used pre-trained model in such a setting.

To better assess the improvement brought by the above two CL methods and their vulnerabilities under attacks, we establish two other methods for comparison.

- **Baseline curriculum.** It uses a random curriculum that is irrelevant to the data samples' difficulty. This curriculum is then used across all training epochs. The normal training process is different in that a random order is drawn for every training epoch.
- **Anti-curriculum.** It shares the same difficulty measurer with bootstrapping but arranges the samples from difficult to easy, reversing the outcome of bootstrapping.

For the pacing function $g$, we choose varied exponential pacing [89], exponentially increasing the fraction of data by steps (a step denotes the iterations with the same output of $g$). According to [89], different pacing functions perform similarly.

In summary, the four CL methods differ in the difficulty measurer and each CL method feeds training data using the same curriculum (or ordering) across all epochs. The baseline and anti-curriculum methods help us understand the contribution of data ranking and order fixing separately (e.g., anti-curriculum can be considered as using a wrong curriculum but still repeating the order across epochs as advised by CL).

[3]It is a widely-used image recognition model that achieves over 78.1% accuracy on the ImageNet dataset [56].

As described in Section 5.2.1, CL can accelerate the training process to reach higher accuracy. We first validate this claim by evaluating the training performance and the testing accuracy and comparing them to the normal training method, which does not use any curriculum as guidance.

| Dataset \ Method | Normal | Bootstrapping | Anti-curriculum | Baseline | Transfer Learning |
|---|---|---|---|---|---|
| Tiny ImageNet | 0.3842 | **0.4002** | 0.3776 | 0.3798 | 0.3803 |
| CIFAR100 | 0.6081 | **0.6232** | 0.5991 | 0.6099 | 0.6127 |
| Place100 | 0.2992 | **0.3159** | 0.2967 | 0.3088 | 0.3007 |
| Place60 | 0.4756 | **0.4903** | 0.4815 | 0.4847 | 0.4707 |
| SVHN | 0.9592 | 0.9598 | 0.9566 | 0.9593 | **0.9599** |
| Purchase | 0.4931 | **0.5324** | 0.4760 | 0.5289 | - |
| Texas | 0.4809 | **0.4975** | 0.4606 | 0.4877 | - |
| Location | 0.5861 | **0.5914** | 0.5563 | 0.5838 | - |

Table 5.1: Target model's average test accuracy on different datasets. ResNet-18 is used for all image datasets, and MLP for non-image datasets Purchase, Texas, and Location. Transfer learning CL does not apply to non-image datasets. The target model accuracy is relatively low except for SVHN because we use a subset of the original training data.

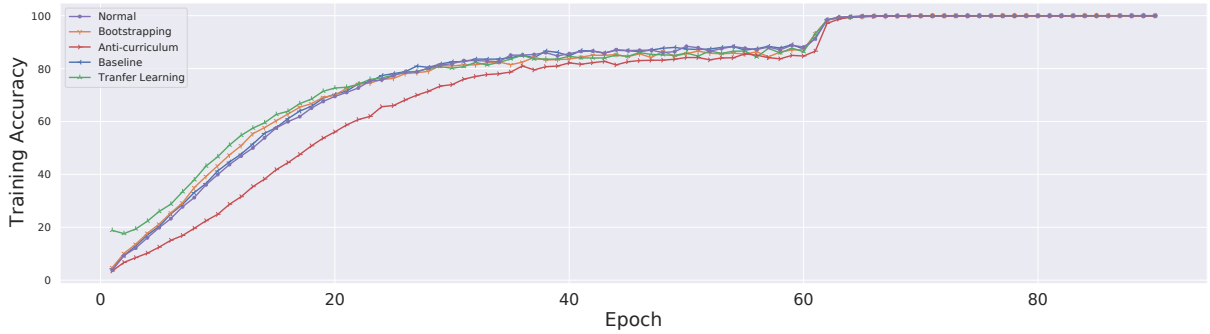


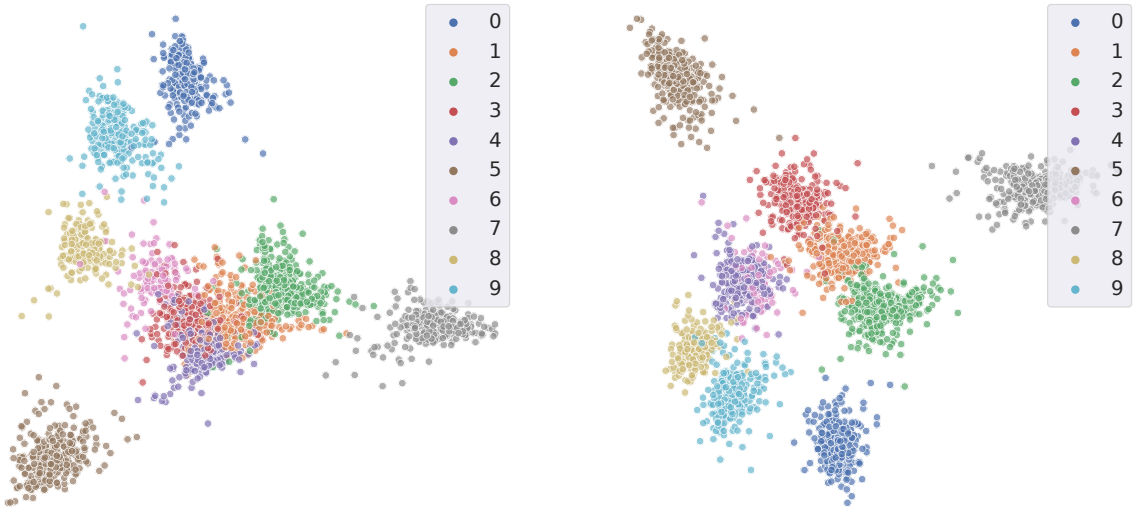

Figure 5.1: The training accuracy of different training methods with ResNet-18 on CIFAR100 along the increase of epochs (total of 90 epochs). Bootstrapping, transfer learning, and baseline reach higher accuracy faster and converge to a better result.

Table 5.1 validates the effectiveness of CL. At least one of the four CL methods can outperform the normal training by 0.06% to 4.42%, and the corresponding average training accuracy is given in Table 5.2. The maximum standard deviation in Table 5.1 is 0.0221 while 32 out of 37 results have a standard deviation less than 0.01. This indicates the difference among various CL methods is statistically significant.

| Dataset \ Method | Normal | Bootstrapping | Transfer | Baseline | Anti-curriculum |
|---|---|---|---|---|---|
| Tiny ImageNet | 100.0 | 100.0 | 100.0 | 100.0 | 99.963 |
| CIFAR100 | 100.0 | 100.0 | 99.997 | 99.993 | 100.0 |
| Place100 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| Place60 | 100.0 | 99.996 | 99.972 | 100.0 | 99.918 |
| SVHN | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| Purchase | 100.0 | 100.0 | / | 99.990 | 100.0 |
| Texas | 96.770 | 94.030 | / | 95.600 | 97.410 |
| Location | 100.0 | 100.0 | / | 100.0 | 100.0 |

Table 5.2: The average training accuracy of datasets in Table 5.1. Image datasets are trained on ResNet-18 while non-image datasets are trained on MLP. Numbers are all in percentage. We observe that all training accuracies are nearly 100%. Note that for non-image datasets, we skip the transfer method as there is no commonly used pre-trained model for the tabular dataset.

It is worth noticing that bootstrapping and transfer learning always outperform normal training, and anti-curriculum performs the worst consistently. Interestingly, we observe that the baseline performs as well as the transfer learning curriculum for Place100 and Place60, which means the transfer learning curriculum does not suit these two datasets well. Figure 5.1 validates the major motivation of adopting CL, i.e., reaching higher accuracy while converging faster. Throughout most of the training, bootstrapping and transfer learning reach higher accuracy faster than all the other methods. At the same time, it takes the longest for the anti-curriculum to reach the same training accuracy compared to all other methods. This indicates that repeating a meaningful data order improves training. This observation aligns with the discovery from previous work [241, 89]. Finally, CL is expected to have a disparate impact on classification accuracy across samples. Besides the analysis in Section 5.5, we also use t-distributed stochastic neighbor embedding (t-SNE) to visualize the classification tasks carried out by bootstrapping and normal ML on the most difficult batch of data of SVHN. Figure 5.2 shows all samples within the difficult batch, and it turns out bootstrapping can separate samples from group "1", "2" and "3" better than normal training.




(a) Normal training (b) Bootstrapping

Figure 5.2: t-SNE of the classification results on the difficult batch of SVHN.

### 5.4.2 Basic MIA

After providing a high-level overview of MIA in Section 5.2.2, we now delve into the details, focusing on the three well-known attacks: NN-based (Neural Network-based) [207, 202], metric-based [213], and label-only attacks [151, 53].

NN-based attack assumes a vector of *prediction posteriors* (e.g., confidence scores or loss) of all class labels can be returned by the target model $T$ when querying $T$ with a data sample $x$. It is also assumed that the adversary has a *shadow dataset* ($\mathcal{D}$) that has the same distribution and format as $T$'s private training dataset. $\mathcal{D}$ is used to train a set of *shadow models* $\mathcal{S}$ that behave similarly as $T$ (e.g., having the same architecture as $T$ like previous work [207, 203, 213]). The attacker trains an *attack model* $\mathcal{A}_{MI}$ using $\mathcal{S}$. In particular, the attacker queries every shadow model $\mathcal{S}$ with the samples from its own training dataset and a disjoint testing dataset. The prediction posteriors of all samples and whether they are in training (denoted member) or testing (denoted non-member) are used as input to train $\mathcal{A}_{MI}$. Finally, the attacker queries $T$ with a sample of interest $x$ and uses the prediction posteriors

as the input to $\mathcal{A}_{MI}$ to predict the membership status.

Compared to the NN-based attack, the model $\mathcal{A}_{MI}$ of metric-based attacks does not need to be trained. Instead, $\mathcal{A}_{MI}$ generates a privacy risk score from the output of $T$ and compares it to class-specific thresholds.

For the label-only attack, it assumes only the prediction label instead of the prediction posteriors are returned from $T$. Still, the adversary can continuously add adversarial perturbations to the input sample $x$ until its prediction label has been changed. The key insight is that the magnitude of the adversarial perturbation is larger for the member sample as $T$ gives a more confident prediction. $\mathcal{D}$ and $\mathcal{S}$ can be used to select a threshold to separate the perturbation magnitudes of members and non-members.

**MIA Models.** Following the original setting of the NN-based attacks [207], we adopt a 3-layer MLP with 64 and 32 hidden neurons, and 2 output neurons, as our attack model $\mathcal{A}_{MI}$. We use cross-entropy as the loss function and Adam as the optimizer with a learning rate of 0.01. $\mathcal{A}_{MI}$ is trained for 100 epochs. For metric-based attacks, we follow the implementation of Song et al. [213] and consider 4 metrics, including correctness, confidence, entropy, and modified entropy. The associated attack methods are named metric-corr, metric-conf, metric-ent, and metric-ment. For label-only attacks, we leverage the implementation from ART [225].

Related research has shown that NN-based attacks often, though not universally, achieve better attack accuracy compared to metric-based and label-only attacks [207, 203, 99]. Thus we use NN-based attack (specifically black-box-top3) for most of our evaluation in Section 5.5.

### 5.4.3 Our Proposed MIA

Given that CL orders training samples by difficulty, impacting the model, we investigate the potential enhancement of MIA when the target model is trained under CL. For this purpose, we propose a novel MIA method called *Diff-Cali* specifically tailored for CL. We first introduce calibrated MIA, which serves as inspiration for designing Diff-Cali, followed by the details of Diff-Cali.

**Calibrated MIA.** Recently, Watson et al. [238] proposed to use a calibrated membership score instead of the standard membership score (e.g., loss) to determine whether a sample is a member. Assume $s(T,x)$ is the original membership score, where $T$ is the target model, and $x$ is a sample. The calibrated membership score $s_{cal}(T,x)$ is defined as follows:

$$s_{cal}(T,x) = s(T,x) - \mathbb{E}_{\mathcal{S}\leftarrow\mathcal{A}(\mathcal{D})}[s(\mathcal{S},x)] \tag{5.3}$$

where $\mathcal{S}$ are shadow models[4] that behave similarly as $T$, $\mathcal{D}$ is the shadow dataset, function $s(T,x)$ and $s(\mathcal{S},x)$ output the membership scores from target and shadow models, $\mathcal{A}$ randomly samples subsets of $\mathcal{D}$ to train $\mathcal{S}$, and $\mathbb{E}$ computes the expectation of $s(\mathcal{S},x)$. Finally, $s_{cal}(T,x)$ is compared to a fixed threshold $\theta$, and a sample is considered a member if $s_{cal}(T,x) \geq \theta$.

Previous MIA methods could have a high false positive rate (FPR) on non-members, often over-represented in the samples to be tested by the attacker. Equation 5.3 addresses this issue by using the *difference* between the target model and shadow models to derive the membership signal: if $x$ is non-member to $\mathcal{S}$, it is also more likely non-member to $T$, therefore $s_{cal}(T,x)$ should be small. The evaluation results in [238] show the area under ROC curve (AUC) can be improved "by up to 0.10" (e.g., after calibrating the loss-based membership score with Equation 5.3).

[4] $\mathcal{S}$ are named as reference models in [238], which resemble shadow models [207] as they are also trained on the same data distribution of $T$.

**Difficulty Calibrated MIA (Diff-Cali).** Calibrated MIA compares $s_{cal}(T,x)$ of all samples to a fixed threshold $\theta$, and we argue that $\theta$ can be *calibrated as well*. We observe that a CL curriculum re-orders the samples by their difficulty before the target model is trained, and such strategy changes how a sample is memorized and vulnerable under MIA (see Section 5.5.1 and Section 5.5.2). More specifically, we observe that CL makes the target model more vulnerable to MIA, especially for difficult samples (Finding 1 in Section 5.5.1). Therefore, we can update $\theta$ according to the curriculum and make the attack model more accurate. We assume the attacker can generate a curriculum similar as the one used by the target model. For example, the attacker can use the publicly released pre-trained model to generate the curriculum. Alternatively, the attacker can train shadow models similar to the target model and then build a curriculum according to loss from them.

We implement this idea for NN-based MIA. When the attack model $\mathcal{A}_{MI}$ outputs the prediction posteriors for an input $x$, the posterior of the label "member" is compared against $\theta$, and $x$ is predicted as a member when the posterior is larger. When training $\mathcal{A}_{MI}$, we adjust $\theta$ based on samples' difficulty level to improve the training accuracy, and the pseudo-code is shown in Algorithm 2. Specifically, in each epoch, the calibrated membership scores $s_{cal}(T,\mathcal{D})$ are generated for $\forall x \in \mathcal{D}$, and we use the loss to compute $s$ (Line 2). Next, we try to find the threshold $\theta_0$ (ranging from 0 to 0.1 based on our empirical study) that achieves the best accuracy in separating members and non-members from $\mathcal{D}$ (Line 3). After that, $\mathcal{A}_{MI}$ is updated by minimizing the training loss on $\mathcal{D}$ (Line 4) through adjusting the threshold with the following function:

$$g(x,C,\theta_0) = \frac{(|\mathcal{D}| - C(x))(\theta_0 - 0.0001)}{|\mathcal{D}| - 1} + 0.0001 \tag{5.4}$$

where $C(x)$ indicates the rank of sample $x$ given by curriculum $C$. The rank for the easiest sample is 1, while the most difficult is $|\mathcal{D}|$. $g(x, C, \theta_0)$ is to assign a threshold $\theta$ from $[0.0001,\theta_0]$ (0.0001 is the initial threshold suggested by [238]) to each $x$ based on its difficulty

level (determined by a curriculum $C$), that is, calibrating threshold of each $x$ based their difficulty level. The most difficult sample compares to 0.0001, the easiest one compares to $\theta_0$, and others compare to $\theta$ that is ranged in $[0.0001, \theta_0]$. The more difficult $x$ has a smaller threshold, meaning that we are lowering the bar for them to be predicted as members compared to the easy samples. During the testing phase, the threshold for a sample $x$ is also adjusted with $g(x,C,\theta_0)$.

**Algorithm 2:** Training the attack model and adjusting threshold under Diff-Cali. "pred" is "prediction".

**Input:** Target model $\boldsymbol{T}$ , reference model $\boldsymbol{\mathcal{S}}$, shadow dataset $\boldsymbol{\mathcal{D}}$, labels of shadow dataset $\boldsymbol{L}$, attack model $\boldsymbol{\mathcal{A}_{MI}}$, curriculum $\boldsymbol{C}$, number of epochs $\boldsymbol{E}$

**for** $e \in 1,...,E$ **do**
- $s_{cal}(T,\mathcal{D}) = s(T,\mathcal{D}) - s(S,\mathcal{D})$;
- $\theta_0 = \underset{\theta}{\text{argmax}}\ \text{pred}(\mathcal{A}_{MI}, L, s_{cal}(T,\mathcal{D}))$;
- $\mathcal{A}_{MI} \leftarrow train(\mathcal{A}_{MI}, s_{cal}(T,\mathcal{D}), g(x,C,\theta_0))$;

Diff-Cali follows the direction of addressing the issue caused by over-represented non-members [238, 37]. On top of those works, Diff-Cali is customized under CL to amplify the effects of MIA. To demonstrate the benefit of Diff-Cali, we compare it with the score-based membership attack after difficulty calibration with default threshold in [238] (Cal).

Overall, Diff-Cali outperforms Cal by 4.0% to 9.9% of attack accuracy while maintaining the same AUC. Besides, Diff-Cali improves MIA's TPR at extremely low FPR, making the difficult sample more vulnerable. This focus (on the low FPR regime) is the setting with the most practical consequences, i.e., de-identifying even a few users contained in a sensitive dataset is far more significant than making an average-case statement like 'most people are not in the sensitive dataset' [37]. Moreover, we conclude that the knowledge of the actual curriculum being used is not required for the proformance boost from introducing Diff-Cali (See Figure 5.5). The detailed evaluation of Diff-Cali across all metrics such as attack accuracy, confidence score, and TPR at low FPR are presented in Section 5.5.3.

Some recent works suggest to use class-specific thresholds [213]. We did not adjust the

threshold by classes because our threshold has been fine-tuned with difficulty levels.

### 5.4.4 Basic AIA

Song et al. proposed an inference-time attack and model-repurposing attack [212] for AIA, and here we focus on the first attack and follow the same setting as this work. We consider the model evaluation to be partitioned [212] or the model is trained under federated learning [168]. The target model $T$ is split into two parts, i.e., an encoder and a classifier, and the adversary has black-box access to the encoder $E$. The attacker has an auxiliary dataset $D$ containing pairs of $(x,s)$ where $s$ is the sensitive attribute. The embeddings $h$ can be generated by querying $E$, i.e., $h = E(x), \forall x \in D$. All pairs of $(h,s)$ will be used to train the attack model $\mathcal{A}_{AI}$ and later used to predict the values of $s$ in the target model $T$.

**AIA Model.** Our $\mathcal{A}_{AI}$ is a 3-layer MLP with 128 hidden neurons in each hidden layer. We use cross-entropy as the loss function and SGD as the optimizer with a learning rate of 0.01. The attack model is trained for 100 epochs. The dimension of each sample's embedding (i.e., second to the last layer's output) is 512 for ResNet-18, 512 for ResNet-34, and 1024 for MobileNet. To train the target model $T$, we use the label for the original classification (e.g., gender). To train $\mathcal{A}_{AI}$, we use the label from another field (e.g., race).

### 5.4.5 Defense Methods

Some defense methods have been proposed to reduce the success rate of privacy attacks, in particular, MIA. We are interested in how they perform under curriculum learning and our proposed attack. To this end, we select DP-SGD [13], MemGuard [118], MixupMMD [149] and AdvReg [180]. DP-SGD and MemGuard represent two directions in privacy protection, while MixupMMD and AdvReg are two more recent defense methods. Below, we explain the four defense methods.

**DP-SGD.** Differentially-Private Stochastic Gradient Descent (DP-SGD) modifies the stochastic gradient descent (SGD) algorithm and integrates $(\epsilon, \delta)$-DP [61] (see Definition 2.1) to provide provable privacy guarantee.

After a per-sample gradient is computed, DP-SGD clips it to a fixed maximum norm $C$ and Gaussian noise is added to the aggregated parameter gradient with standard deviation $\delta C$. The output of the trained model will satisfy $(\epsilon,\delta)$-DP.

**MemGuard.** Different from DP-SGD, MemGuard does not change the training process. At a high level, it obfuscates the predictions of the target model by adding noises to its output. It is designed to defend against MIA in particular, while DP-SGD deals with all sorts of privacy risks. Assuming an attack model $\mathcal{A}_{MI}$ has been trained with shadow training [207], and $\mathcal{A}_{MI}(T(x),y)$ outputs a confidence score ranging in [0,1], where $T(x)$ is the prediction of the target model and $y$ is the label for $x$. A sample is considered a member if the score is larger than 0.5 and a non-member if smaller than 0.5. MemGuard has two phases. In Phase 1, it crafts adversarial noise and adds it to $T(x)$ to force $\mathcal{A}_{MI}(T(x),y)$ to be 0.5 to confuse the attacker, while the distance between the original prediction and the noisy prediction is minimized. In phase II, the adversary adds the noise to the original prediction with a certain probability of trade-off the utility and privacy.

**MixupMMD.** Li et al. [149] found a model vulnerability under MIA relates to the difference between the training and testing accuracy, and they proposed MixupMMD to intentionally reduce the training accuracy to validation accuracy. A new penalty, Maximum Mean Discrepancy (MMD), is used by the regularizer.

**AdvReg.** Nasr et al. [180] proposed to mitigate MIA by formulating the defense as a min-max optimization problem. Given a validation set that serves as "non-members", AdvReg introduces an adversarial classifier to infer the membership status using the posteriors generated from the target model. The optimization goal is to minimize the original classification

| Dataset \ Method | Normal | Bootstrapping | Anti-curriculum | Baseline | Transfer Learning |
|---|---|---|---|---|---|
| Tiny ImageNet | 0.9193 | 0.9385 | 0.9116 | 0.9207 | **0.9439** |
| CIFAR100 | 0.8577 | **0.8751** | 0.8376 | 0.8582 | 0.8718 |
| Place100 | 0.9425 | 0.9549 | 0.9335 | 0.9416 | **0.9617** |
| Place60 | 0.8773 | **0.8987** | 0.8625 | 0.8827 | 0.8902 |
| SVHN | 0.5570 | **0.5605** | 0.5514 | 0.5599 | 0.5580 |
| Purchase | **0.9524** | 0.9453 | 0.9118 | 0.9458 | - |
| Texas | 0.6749 | **0.7068** | 0.5950 | 0.7039 | - |
| Location | 0.9153 | **0.9194** | 0.8980 | 0.9169 | - |

Table 5.3: Accuracy of NN-based MIA on models trained on 8 datasets. Transfer learning CL does not apply to non-image dataset Purchase, Texas and Location.

loss and maximize the loss of the adversarial classifier.

## 5.5 Evaluation Results

In this section, we present the evaluation results of MIA and AIA when CL is applied to train the target model. We also attempt to explain the observations from the angle of data memorization and show the impact of CL on the existing defenses. We highlight our insights with text boxes.

**Evaluation setup.** To evaluate MIA and AIA, we split each dataset described in Section 5.3 into three disjoint parts: one for training the target model, one for training a shadow model, and one for testing both the target and shadow model.

To evaluate the defense methods, we split each dataset into five parts as some advanced methods need reference datasets for training. More details about the defenses can be found in Section 5.5.5. All experiments were *repeated 5 times* to minimize the fortuitous outcomes, and the mean value and standard deviation were reported.

**Evaluation metrics.** First, we compute the attack accuracy, measured by the correct predictions (member/non-member) versus all predictions, to assess the effectiveness of MIA/AIA,

and the classification accuracy of the target model to assess the impact of curriculum learning and defenses. Second, to better understand the attack results, we retrieve the confidence scores of members and non-members, respectively. Note that the confidence score indicates the likelihood of a sample being classified as a member or non-member. Third, we compute the true-positive rate (TPR) at the false-positive rate (FPR) of the attacks. As noted by Carlini et al. [37], attacks should emphasize the member guesses over non-member guesses, so they should be evaluated by considering TPR at low FPR. This cannot be precisely modeled by the overall accuracy, precision, or recall.

### 5.5.1 Evaluation of Basic MIA

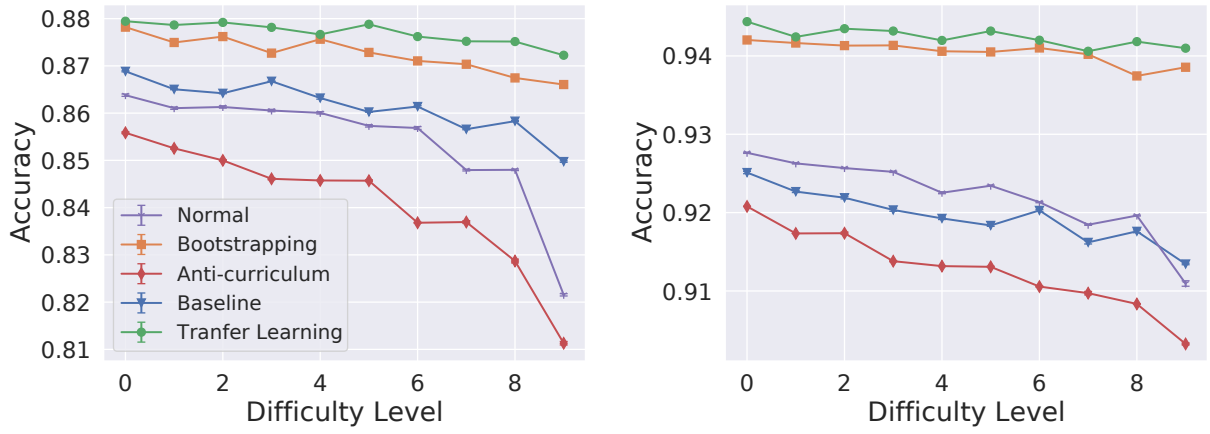
We start with the experiments on the 5 image datasets (CIFAR100, Tiny ImageNet, Place100, Place60, and SVHN), using ResNet-18 as the target model architecture and later ResNet-34 and MobileNet for comparison. The evaluation of the tabular datasets (Purchase, Texas hospital stays, and Locations) is presented at the end. The attack models are described in Section 5.4.2.

**MIA Accuracy.** We found that models trained using meaningful CL methods (i.e., bootstrapping and transfer learning) are slightly **more vulnerable** to MIA. Table 5.3 shows the accuracy of NN-based black-box-top3 MIA [207] by datasets and CL methods. The biggest attack accuracy improvement observed for image datasets is 2.46% (Tiny ImageNet with transfer learning) while the biggest improvement for non-image datasets is 3.20% (Texas with bootstrapping). Among different CL methods, bootstrapping and transfer learning are the most vulnerable, with an average of 1.29% and 1.44% improvement in the attack accuracy against the normal training, respectively. For baseline CL, the attack accuracy decreases for Place100, whereas a slight increase is observed for the attack accuracy on other datasets. For anti-curriculum CL, the attack accuracy decreases for all datasets. This result indicates both the data repeating (reflected by the results of baseline) and ordering (reflected by the results

of bootstrapping and anti-curriculum) of CL (explained in Section 5.4.1) contribute to the vulnerability under MIA. The consistent performance of bootstrapping and anti-curriculum indicates that **data ordering plays a bigger role**.

Regarding the impact of datasets, we found more complex datasets (e.g., with more classes of labels) tend to have higher attack accuracy in general. For example, the average MIA accuracy is 94.39% for Tiny ImageNet (200 classes), 87.18% for CIFAR100 (100 classes), 96.17% for Place100 (100 classes), 89.02% for Place60 (60 classes), and 55.80% for SVHN (10 classes), all under transfer learning. The same effects have also been observed in other works like [207].

Regarding the metric-based and label-only attacks, the result is similar to the NN-based attack, as suggested by the evaluation on CIFAR100, shown in Table 5.4. The only exception is metric-corr, which performs worse than other attacks with bootstrapping. This result can be explained by the assumption of metric-corr that the target model is trained to predict correctly on its training data, which may not generalize well on the test data. In the rest of the evaluation, we fix the attack model to black-box-top3, and the NN-based attack in the rest of the paper primarily refers to black-box-top3, unless indicated otherwise.




(a) CIFAR100

(b) Tiny ImageNet

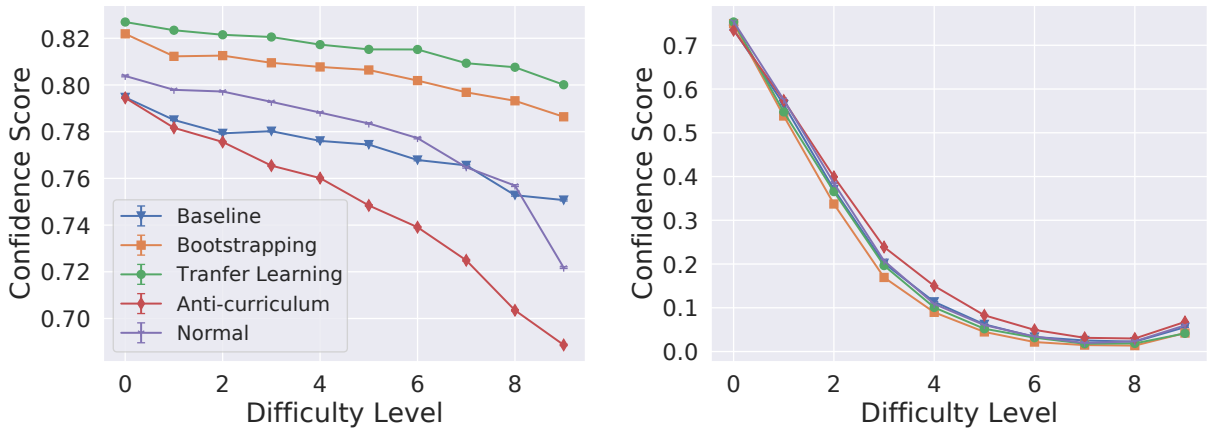
Figure 5.3: MIA accuracy on CIFAR-100, Tiny ImageNet. ResNet-18 is used for target model training.

| Attack \ Method | Normal | Bootstrapping | Anti-CL | Baseline | Transfer Learning |
|---|---|---|---|---|---|
| NN-based [207] | 0.8577 | **0.8751** | 0.8376 | 0.8582 | 0.8718 |
| Metric-corr [213] | 0.6920 | 0.6820 | 0.6905 | **0.6930** | 0.6855 |
| Metric-conf [213] | 0.8600 | **0.8810** | 0.8458 | 0.8553 | 0.8740 |
| Metric-ent [213] | 0.8490 | **0.8750** | 0.8320 | 0.8435 | 0.8685 |
| Metric-ment [213] | 0.8620 | **0.8820** | 0.8463 | 0.8568 | 0.8760 |
| Label-only [225] | 0.8200 | **0.8263** | 0.7963 | 0.8050 | 0.8088 |
| Cali [238] | 0.7889 | **0.8272** | 0.7532 | 0.7781 | 0.8148 |
| Diff-Cali | 0.8519 | **0.8670** | 0.8382 | 0.8438 | 0.8614 |

Table 5.4: Average accuracy of NN-based, metric-based, label-only and our Diff-Cali attacks on models trained on CIFAR100 with ResNet-18.

Figure 5.3 shows the attack accuracy of samples from different difficulty levels. More specifically, we construct the test dataset as half member samples and half non-member samples. Member samples are divided into different difficulty levels while non-member samples across each difficulty level are fixed. Figure 5.3 demonstrates that using a meaningful curriculum (i.e., bootstrapping and transfer learning) makes the model more vulnerable, especially for the difficult samples.

**Confidence Score.** Since the key contribution of CL is to factor in the samples' difficulty levels during the training procedure, here we evaluate how difficulty levels impact the samples' vulnerability individually. Intuitively, the difficult samples should be harder to attack. However, since CL forces the model to learn the samples in a repetitive manner, we want to find out whether samples will be remembered by the model differently. To assess and quantify the possible privacy risk discrepancy caused by CL, we first arrange samples according to their difficulty level. Then, we use the confidence score and attack accuracy to analyze individual samples. Note that we train a separate model and use the sample loss given by this model as a guide to determine how difficult a sample is. This model is used solely for getting the difficulty levels of all samples and is different from the target model in our following evaluation.




(a) Members of CIFAR100

(b) Non-members of CIFAR100

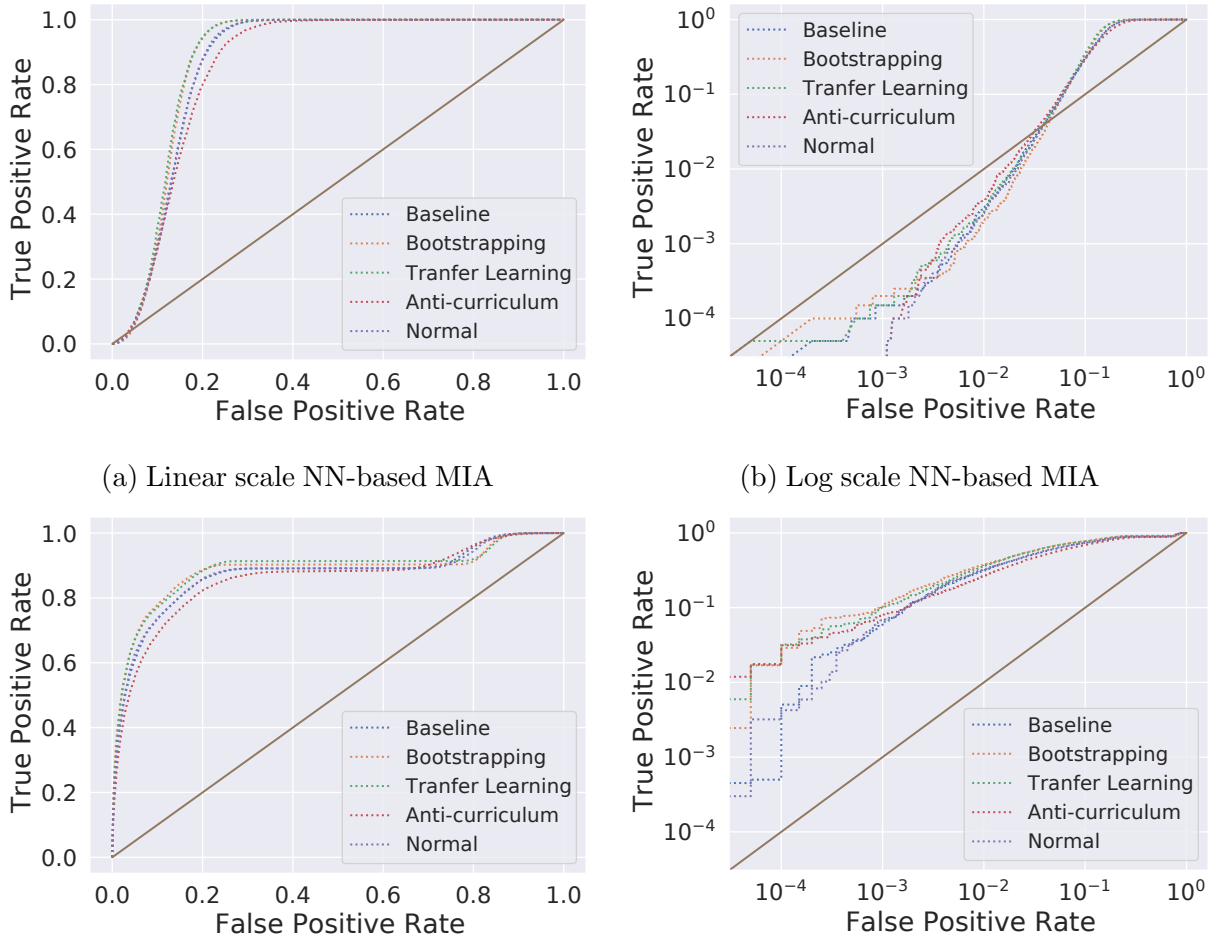
(c) Members of Tiny ImageNet

(d) Non-members of Tiny ImageNet

Figure 5.4: Attack model's confidence score for both member and non-member samples on CIFAR-100 and Tiny ImageNet. ResNet-18 is used for target model training, and data samples are arranged according to their difficulty scores from bootstrapping.

Figure 5.4 depicts the attack model's confidence score by samples' difficulty levels. Though the difficult samples are not more vulnerable than the easy samples, **the gap in confidence scores is much narrower** (especially for the confidence score of members). Take the target model in CIFAR100 as an example, our attack model can recognize the most difficult member samples (scored as difficulty level 9) from this model with over 7.83% (absolute growth from 72.19% to 80.02%) more confidence, thanks to transfer learning (Figure 5.4a). Interestingly, for the most difficult member samples, it is even possible for anti-curriculum to have a higher confidence score compared to the normal training ((Figure 5.4c) ). This observation indicates

that enforcing difficult samples to the training process first does not necessarily make the model more likely to forget them. If we perceive feeding difficult samples first to a model as negative, the repetition of a curriculum can possibly compensate for such a negative effect, i.e., making the target model memorize the difficult samples better than a normal ML where these samples are presented at random times throughout training.




(a) Linear scale NN-based MIA

(b) Log scale NN-based MIA

(c) Linear scale Diff-Cali

(d) Log scale Diff-Cali

Figure 5.5: TPR/FPR of NN-based MIA and Diff-Cali under different training method trained with ResNet-18 on CIFAR100.

**TPR at Low FPR.** In addition to the attack accuracy, we measured the relation between TPR at low FPR, as explained in "evaluation metrics" (Section 5.5). Following Carlini et al. [37], we present the ROC curve for the attacks with both linear scaling and log scaling to emphasize the low-FPR regime. Figure 5.5a and Figure 5.5b demonstrate the ROC curve

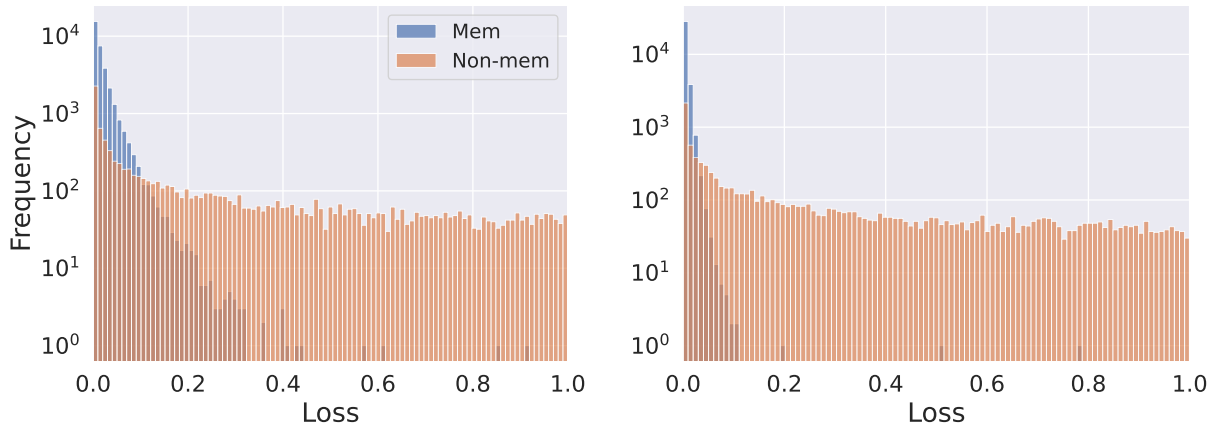
| Method \ Architecture | ResNet-18 | ResNet-34 | MobileNet |
|---|---|---|---|
| Normal | 0.8572 ± 0.0011 | 0.8564 ± 0.0001 | 0.7979 ± 0.0001 |
| Bootstrapping | **0.8751** ± 0.0001 | **0.8746** ± 0.0003 | 0.8308 ± 0.0000 |
| Anti-curriculum | 0.8376 ± 0.0002 | 0.8481 ± 0.0002 | 0.7763 ± 0.0002 |
| Baseline | 0.8582 ± 0.0001 | 0.8559 ± 0.0002 | 0.8318 ± 0.0000 |
| Transfer Learning | 0.8718 ± 0.0001 | 0.8715 ± 0.0002 | **0.8430** ± 0.0001 |

Table 5.5: The average accuracy of NN-based attacks on models trained on different network architectures with CIFAR100.

for NN-based attack. The results show that using curriculum increases ROC. The TPR of transfer learning and bootstrapping are generally higher than the others except at extremely low FPR ($< 10^{-4}$). This indicates CL introduces disparate impact to members and non-members for most samples. Moreover, the NN-based attack fails to achieve a TPR better than random chance at any FPR below 0.045, indicating potential for further improvement.

**Loss Distribution.** The previous evaluation presents a macro-level understanding of CL's impact on MIA. Here we present a micro-level analysis by examining the loss distribution between members and non-members in models trained with normal and CL methods. Due to the space limitation, here we only show the results of ResNet-18 trained on Tiny ImageNet in Figure 5.6 which shows a clearer discrepancy in terms of the loss distributions comparing to other datasets. Note that the loss scores are normalized. As one can see, there is a clear difference between their loss distributions, e.g., bootstrapping makes the overall members' loss much lower and the members' loss distribution less overlapped with non-members', especially for those members with higher difficulty levels. In Section 5.5.2, we also reason this observation from the perspective of data memorization.

**Target Model Architectures.** To study the impact of the architecture of the target model, we launched MIA against ResNet-34 and MobileNet and compare the results against ResNet-18. Table 5.5 demonstrates the average attack accuracy of MIA when target models are trained with ResNet-18, ResNet-34, and MobileNet, respectively. It shows that they



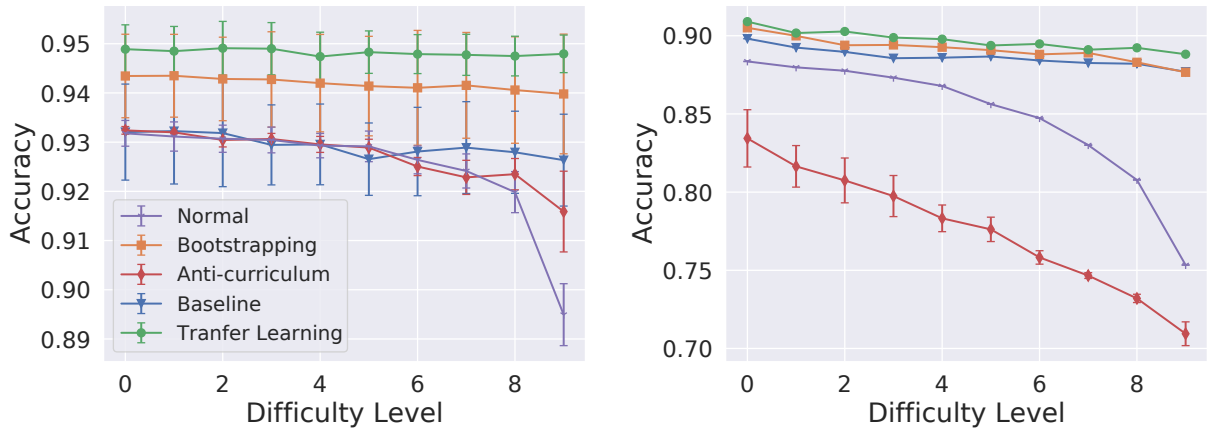

(a) Normal training

(b) Bootstrapping

Figure 5.6: Loss distribution for models trained on Tiny ImageNet with ResNet-18.

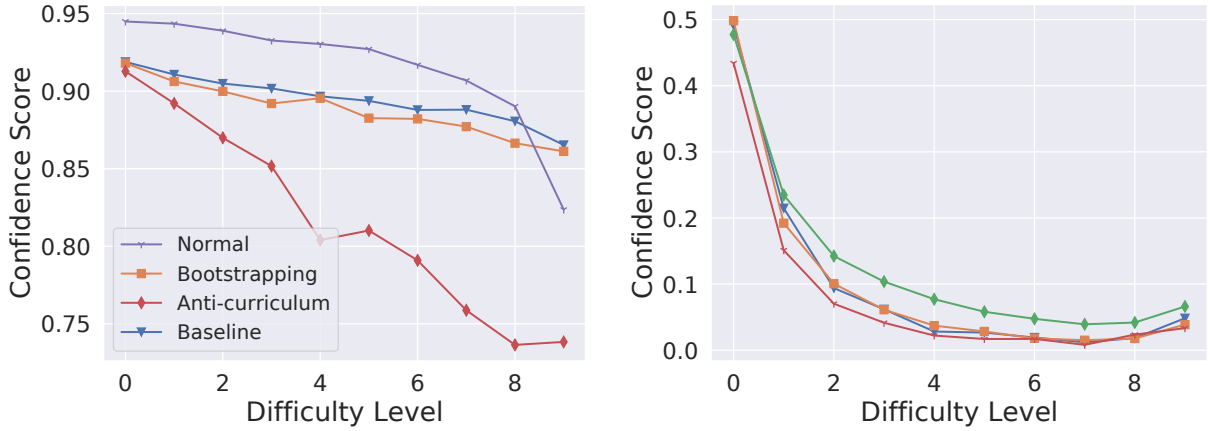
all share a similar trend of how CL affects MIA. Though MobileNet turns out to be less vulnerable (5.85% and 5.93% less attack accuracy compared to ResNet-34 and ResNet-18, respectively), bootstrapping, transfer learning, and baseline all increase the overall attack accuracy. Figure 5.7 demonstrates the results by difficulty levels on ResNet-34 and MobileNet when training with Tiny ImageNet, which can be viewed together with Figure 5.3b about ResNet-18. Though MobileNet turns out to be less vulnerable (4% less attack accuracy compared to ResNet-34 and ResNet-18), bootstrapping, transfer learning, and baseline all increase the overall attack accuracy and narrow down the gap between difficult and easy samples. As such, the privacy concerns in CL cannot be addressed by changing the target models' architectures. This observation is consistent with other works [151, 99] about MIA vs. architectures.

**Non-image Datasets.** As shown in Table 5.3, most experiments remain to have the same trend they are showing in image datasets. For Purchase, however, attack accuracy on normal training is 0.71% higher than bootstrapping for example. This shows that CL does not always empower MIA more. In Figure 5.8, we show the confidence score of members and non-members on Purchase, and the result is similar to the image datasets, where difficult samples are more vulnerable.




(a) ResNet-34 (b) MobileNet

Figure 5.7: MIA accuracy for target model trained on Tiny ImageNet with ResNet-34 and MobileNet, respectively.




(a) Purchase member (b) Purchase non-member

Figure 5.8: Attack model's confidence score for both member and non-member samples on Purchase. MLP is used for target model training, and data samples are arranged according to their difficulty scores from bootstrapping.

In the meantime, we found the changes caused by different CL methods are more drastic on the non-image datasets, compared to the image datasets. For example, Texas has a more prominent attack accuracy drop (8.0%) on anti-curriculum. The non-image datasets are relatively simple, containing only binary features after pre-processing, hence they are more likely to be impacted by CL. Table 5.1 also shows the target model accuracy varies more for the non-image datasets under CL.

Finding 1: CL makes the target model more vulnerable to MIA, especially for difficult samples .

Finding 2: Both data ordering and data repeating make a model more vulnerable under MIA, while data ordering plays a bigger role in influencing the vulnerability of a model under MIA.

### 5.5.2 Analysis with Data Memorization

The previous experiments show CL makes the difficult samples more vulnerable. Here, we attempt to explain this observation with a more principled analysis. Recent works [70, 72] suggest the effectiveness of MIA could be tied to how well the target model *memorizes* individual data sample. The notion of memorization is formally defined as [70]:

$$\mathbf{mem}(\mathcal{A},\mathcal{D},i) := \Pr_{T\sim\mathcal{A}(\mathcal{D})}[T(x_i)=y_i] - \Pr_{T\sim\mathcal{A}\left(\mathcal{D}^{\setminus i}\right)}[T(x_i)=y_i] \tag{5.5}$$

where $\mathcal{A}$ denotes the training algorithm, $\mathcal{D}$ denotes the training dataset, $T$ is the trained model, $(x_i,y_i)$ denotes one sample with its ground-truth label, and $\mathcal{D}^{\setminus i}$ denotes $\mathcal{D}$ with $i$-th sample removed. The model is likely to memorize the data sample if adding $(x_i,y_i)$ to training significantly changes the model's prediction on $y_i$. Though Equation 5.5 models the memorization of a single data sample, we can easily extend it to quantify the memorization of multiple samples at once.

Specifically, we evaluate ResNet-18 trained with CIFAR100. We first leave out 800 most difficult data samples (4% of all samples) and train a model without these data via bootstrapping ("not seen"). Then, we train the model under CL according to data memorization: the curriculum makes the 800 data samples either be seen at the beginning ("first seen"), end ("last seen"), or random places ("random") of each training epoch. Figure 5.9 depicts the prediction probability of the true labels of the 4 scenarios. Data memorization under

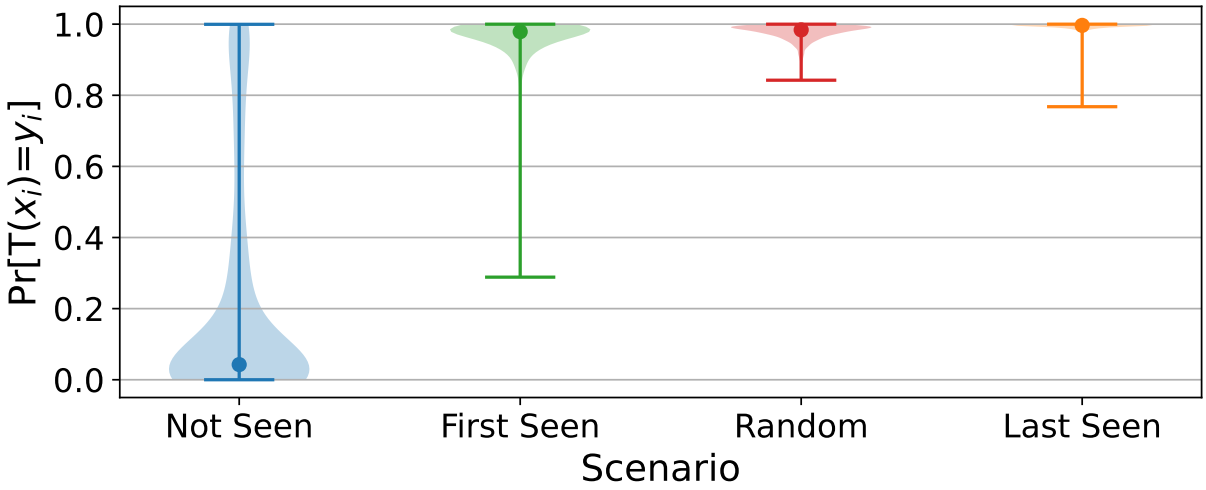



Figure 5.9: Memorization: violin plots of prediction probability of 800 most difficult samples, according to bootstrapping CL. The horizontal bars of each violin represent the minimum and maximum of the prediction probability.

CL can be assessed by comparing "first seen", "last seen", and "random" to "not seen", following the idea of Equation 5.5. We observe that other than "not seen", the other three scenarios memorize the difficult samples fairly well (higher prediction probability of the true class). It turns out that data ordering has a strong impact on data memorization, e.g., "last seen" provides the strongest memorization compared to "first seen" and "random". The difficult samples are more vulnerable under CL because they are memorized better after data ordering.

Here, we elaborate on the topic of data Shapley and study if our observation in this section can be explained from the angle of data valuation. Specifically, we choose Shapley value [80] as the metric, as it has the "strongest theoretical foundation" in data valuation research [91]. In essence, the data with high Shapley values are ones that on average contribute significantly to a model's prediction performance. We follow most of the experiment steps in this section and only change how the samples are selected for "not seen" (i.e., selected based on their Shapley values rather than difficulty levels).

**KNN-Shapley.** Calculating Shapley values is intractable for a DNN model that is trained

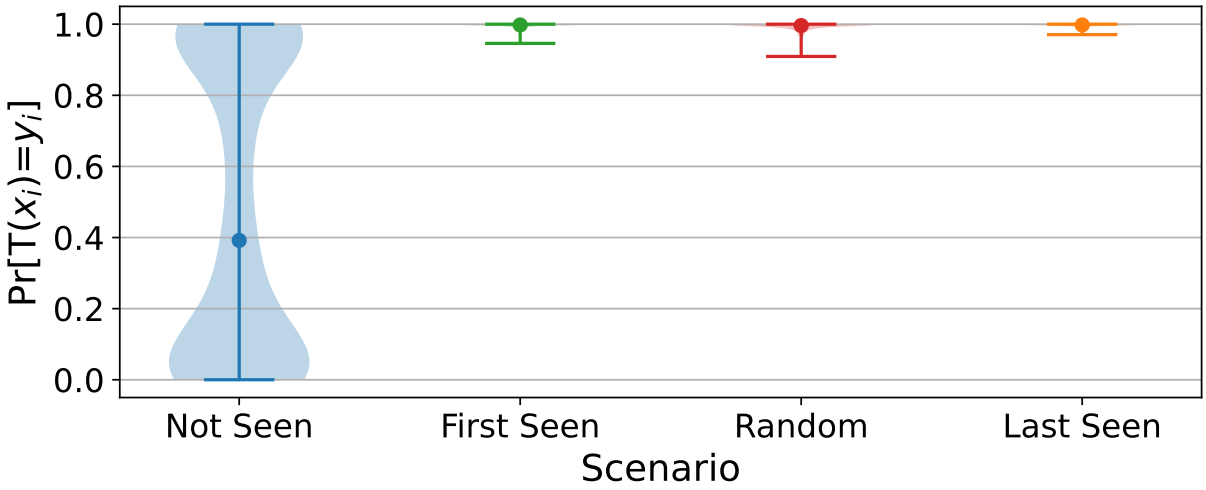



Figure 5.10: Shapley: violin plots of prediction probability of 800 most valuable samples according to KNN-Shapley.

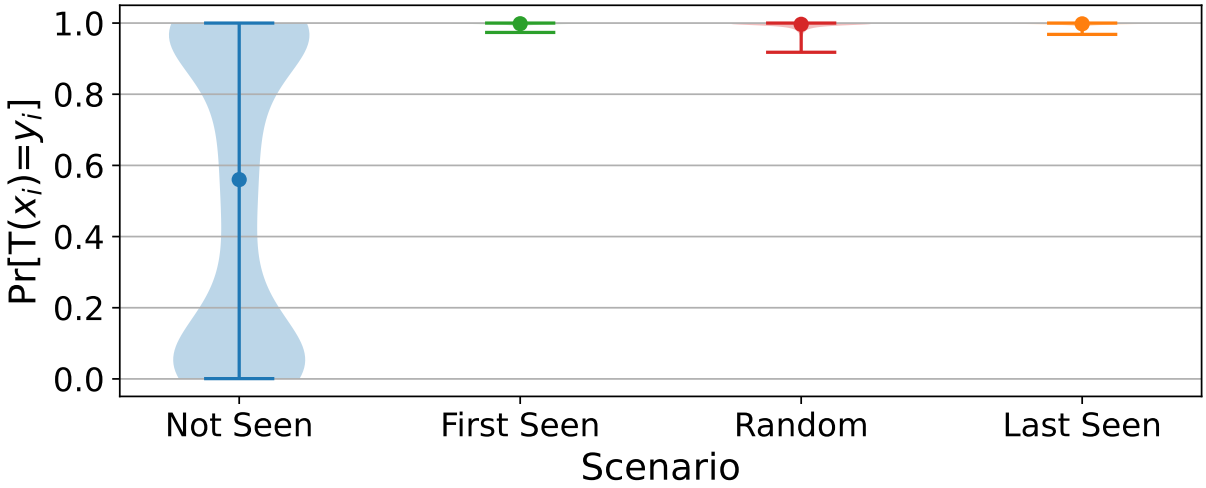



Figure 5.11: Reverse Shapley: violin plots of prediction probability of 800 least valuable samples according to KNN-Shapley.

on a large dataset, as it requires a model to be retrained for $2^n$ times, where $n$ is the number of data points, to assess the contribution of one data point versus all possible subsets of the training set [91]. To address this scalability issue, Jia et al. [119] proposed KNN-Shapley, which uses a lightweight KNN *surrogate* model to reduce the overhead of model retraining. The time complexity is reduced to $O(nlogn)$ and still, a good approximation of Shapley values can be obtained. As such, we use KNN-Shapley to calculate the Data Shapley values.

Figure 5.10 and Figure 5.11 show the prediction probability of true label with 800 most and least valuable data samples according to KNN-Shapley. From the results of "not seen", we observe that the least valuable data have higher prediction accuracy on average (51%), meaning that their absence in training has less impact compared to the more valuable data as presented in Figure 5.10. Similarly, feeding the least valuable data first or at last to the training does not affect the prediction much.

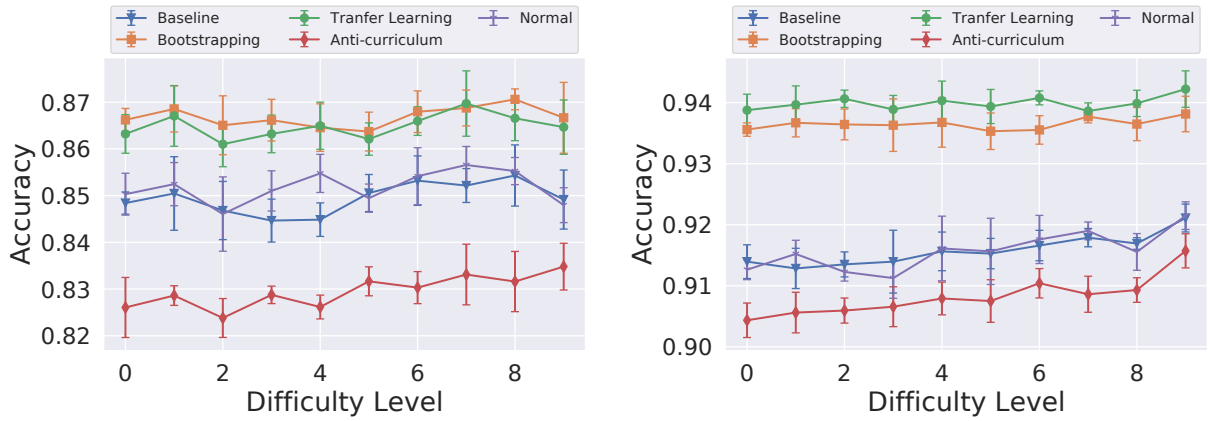
Then, we compare the impact of difficulty level and Shapley value on data memorization, from Figure 5.9 and Figure 5.10. Though both show that the absence of the most difficult or valuable data leads to poor prediction and seeing these data lastly benefits more than seeing them first during training, these changes are much more drastic for difficult samples (Figure 5.9) than the valuable samples (Figure 5.10). For example, the median prediction probability of the "not seen" difficult samples and valuable samples are 39.19% and 56.01%. As such, the data reordering of CL makes the difficult samples more vulnerable, but not so for the valuable samples.

Finding 3: CL forces the model to memorize the difficult samples harder, which makes them more vulnerable.

### 5.5.3 Evaluation of Diff-Cali

In order to fully utilize the information of difficulty levels exposed by CL, we propose Diff-Cali as described in Section 5.4.3. Overall, the NN-based attack still has a slightly better attack accuracy compared to Diff-Cali, but Diff-Cali has higher confidence scores for difficult samples and has better TPR at the low FPR regime.

**Attack Accuracy.** Table 5.4 presents the accuracy of Diff-Cali, which is about 1% lower compared to NN-based attack on all CL methods. Figure 5.12 depicts the attack accuracy on CIFAR100 and Tiny ImageNet. ThoughDiff-Cali achieves slightly lower (less than 1.44%)



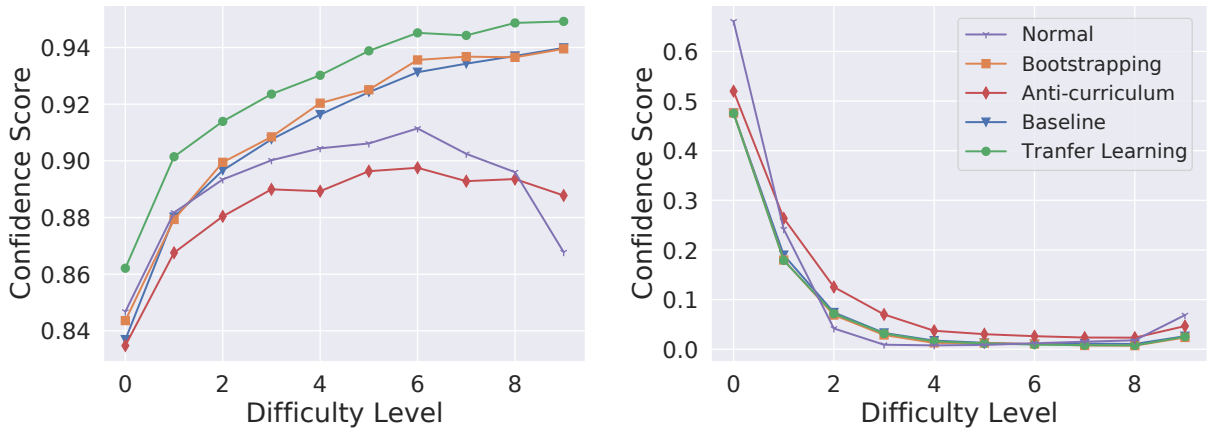

(a) CIFAR100 (b) Tiny ImageNet

Figure 5.12: Diff-Cali's accuracy for models trained on CIFAR100 and Tiny ImageNet with ResNet-18.

accuracy compared to NN-based attack, with adaptive calibration, we are able to make **the difficult samples more vulnerable**: e.g., the attack accuracy of difficulty level at 9 and 0 are 86.47% and 86.32% for transfer learning under CIFAR100. The most difficult samples now can be predicted 2.64% and 2.35% more accurately for normal and anti-curriculum ML, respectively. Overall, Diff-Cali is able to overcome the privacy risk discrepancy of different samples through calibration and results in better attack accuracy for difficult samples for normal ML and anti-curriculum ML.

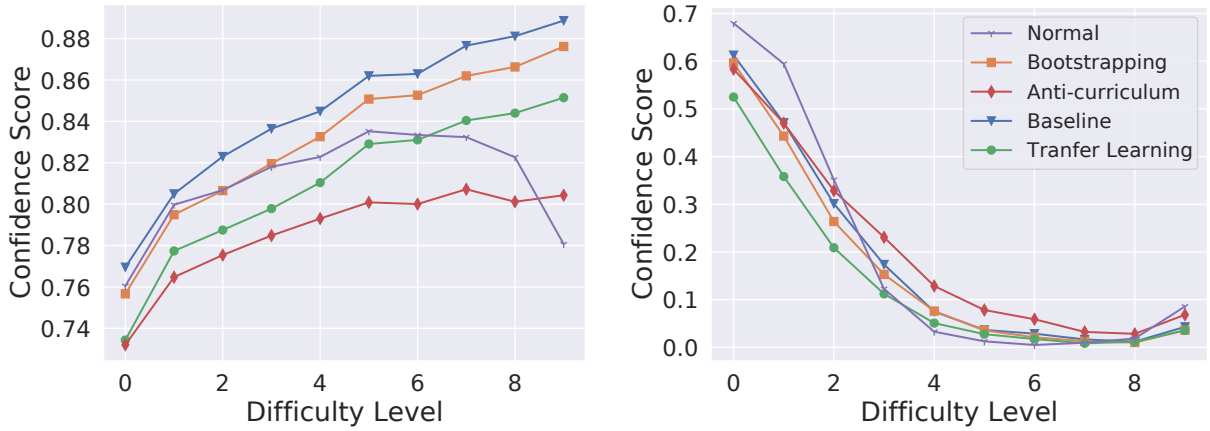
**Confidence Score.** Like the evaluation of basic MIA, we show the confidence scores of samples according to their difficulty level in Figure 5.14 and Figure 5.13.

Overall, we are able to achieve confidence scores greater than 0.7807 (normal) for CIFAR100 and 0.8678 (normal) for Tiny ImageNet for all member samples, whereas the minimum member confidence score from NN-based is 0.6889 for CIFAR100 and 0.8333 for Tiny ImageNet (Figure 5.4). In short, we are able to improve the normal training confidence score for all members by 3.29% for CIFAR100 and 3.45% for Tiny ImageNet. Similarly, we reduce the confidence score of non-members (note that a lower confidence score means less chance to be misclassified as non-members) by 0.0414 for CIFAR100 and 0.1751 for Tiny ImageNet.




(a) Member (b) Non-member

Figure 5.13: Diff-Cali's member and non-member confidence score for models trained on Tiny ImageNet with ResNet-18.




(a) Member (b) Non-member

Figure 5.14: Diff-Cali's member and non-member confidence score for models trained on CIFAR100 with ResNet-18.

Unlike previous NN-based attack, the accuracy of Diff-Cali does not share a similar trend as the confidence score because the final prediction of the membership status of Diff-Cali is not based on the confidence score solely.

**TPR at Low FPR.** In Figure 5.5, we show that Diff-Cali can achieve much higher TPR at low FPR ($<10^{-4}$). We present the ROC curve for the attacks with both linear scaling and log scaling to emphasize the low-FPR regime. Figure 5.5c and Figure 5.5d demonstrate the

ROC curve for Diff-Cali. The results show that using curriculum increases ROC (Figure 5.5a, Figure 5.5c). We observe that our proposed Diff-Cali performs better at low FPR. More specifically, Figure 5.5b shows that NN-based attack fails to achieve a TPR better than random chance at any FPR below 0.045 while Diff-Cali can be better than random guessing at all times.

Finding 4: Diff-Cali improves MIA performance in terms of TPR at low FPR, making the difficult samples more vulnerable.

### 5.5.4 Evaluation of AIA

| Method \ Dataset | Place100 | Place60 | UTKFace |
|---|---|---|---|
| Normal | **0.107**±0.003 | **0.173**±0.002 | **0.528**±0.005 |
| Bootstrapping | 0.092±0.003 | 0.168±0.004 | 0.515±0.006 |
| Transfer Learning | 0.104±0.001 | 0.150±0.005 | 0.512±0.006 |
| Baseline Curriculum | 0.079±0.004 | 0.143±0.001 | 0.506 ±0.008 |
| Anti-Curriculum | 0.033±0.001 | 0.128±0.005 | 0.517±0.007 |

Table 5.6: Average accuracy of AIA (± standard deviation (STD)) on model trained with different methods. ResNet-18 is the target model architecture.

We evaluate the 4 CL methods and normal training under the AIA setting described in Section 5.4.4 and Table 5.6 to demonstrate the overall attack accuracy. Generally, our results indicate that CL does not make the target model more vulnerable. This somehow contradicts a recent study [99] showing that a model is more vulnerable under AIA when trained under special settings, i.e., contrastive learning. Interestingly, the normal training yields the highest average attack accuracy (e.g., 0.107 for Place100), even compared to anti-curriculum. UTKFace has a much higher attack accuracy because the baseline accuracy (random guessing based on majority class labels) of UTKFace is already quite high (42.1%). Our further investigation also shows that the attack accuracy is about the same for samples in different groups of difficulty levels (see Figure 5.15). We speculate that this is because

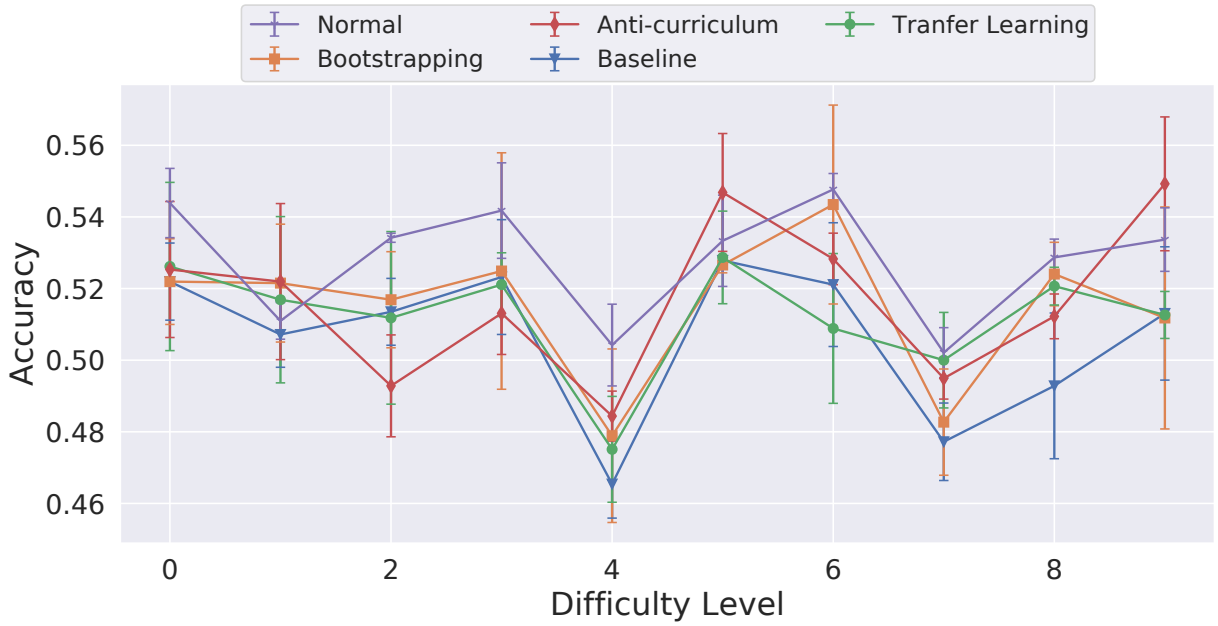



Figure 5.15: Attribute inference attack accuracy on UTKFace

the attributes of a sample themselves are already very complex and hard to learn. Besides, the difficulty score (e.g., bootstrapping) is calculated based on the original ML task, which emphasizes the specific attribute the original ML task tries to learn. That means the data ranking is effective only for the attribute chosen for the classification task but does not influence the sensitive attribute that one intends to infer.

Finding 5: The model trained under CL is less vulnerable under AIA compared to MIA.

### 5.5.5 Evaluation of Defense

We evaluate how the defenses including DP-SGD, MemGuard, MixupMMD, and AdvReg perform under the impact of CL. Table 5.7 shows the attack accuracy on ResNet-18 which is trained with CIFAR100. Because MixupMMD and AdvReg require reference datasets for defense deployment, we equally divided CIFAR100 into 5 parts for fair comparison among all the defense techniques. More specifically, all target models in Table 5.7 are trained with only 12,000 data points, which also explains why the accuracies are lower. Regarding the

setup of the defense methods, bootstrapping and anti-curriculum with DP-SGD are trained with the same curriculum as previous experiments. DP-SGD* uses a *noisy* curriculum for bootstrapping and anti-curriculum, and the difficulty measurer is trained under DP-SGD. For transfer learning, it is not impacted as we use a pre-trained model. $\epsilon$ and $\delta$ in our evaluation are $124,496$ and $1e-5$ for DP-SGD. We have a large $\epsilon$ because we have 200 epochs of training and ResNet-18 contains a large number of parameters. We did not change these settings for a fair comparison with other defense techniques. Previous studies have used large $\epsilon$ for DP-SGD in order to achieve good model accuracy [113, 143]. Based on a recent work [33], we are able to make $\epsilon$ 10 times smaller after proper parameter tuning while achieving similar target accuracy. The $\epsilon$ can be brought down even first with a large batch size. Pulling tricks of DP-SGD based on the above recent work can further boost the tradeoff, we do not discuss it here as that is a parallel line of research. Note that in this section, we still use small batch size for DP-SGD evaluation though that results in large $\epsilon$. This is because we want to keep parameters across all target models the same for a fair MIA evaluation, and we have limited computing resources for handling large batch numbers.

| | | Normal | Bootstrapping | Transfer Learning | Baseline | Anti-CL |
|---|---|---|---|---|---|---|
| None | Target | 48.0 | 51.4 | 48.9 | 50.0 | 49.3 |
| | MIA | 90.3 | 91.4 ±0.03 | 91.3 ±0.03 | 91.5 ±0.02 | 89.5 ±0.02 |
| DP-SGD* | Target | 17.4 | 18.0 | 17.2 | 18.3 | 11.2 |
| | MIA | 50.6 ±0.11 | 50.6 ±0.06 | 50.6 ±0.01 | 50.4 ±0.11 | 50.3 ±0.11 |
| DP-SGD | Target | 17.4 | 17.2 | 17.2 | 17.6 | 17.2 |
| | MIA | 50.8 ±0.07 | 50.7 ±0.01 | 50.6 ±0.01 | 50.4 ±0.11 | 50.4 ±0.10 |
| MemGuard | Target | 48.0 | 51.4 | 48.9 | 50.0 | 49.3 |
| | MIA | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 |
| | Label-only | 83.0 | 84.5 | 84.5 | 84.0 | 81.3 |
| MixupMMD | Target | 54.1 | 54.4 | 55.7 | 55.0 | 52.6 |
| | MIA | 81.6 ±0.02 | 83.1 ±0.02 | 76.1 ±0.03 | 84.4 ±0.02 | 79.1 ±0.02 |
| AdvReg | Target | 51.2 | 54.2 | 50.4 | 53.0 | 52.1 |
| | MIA | 89.2 ±0.01 | 91.6 ±0.02 | 92.8 ±0.04 | 91.6 ±0.01 | 87.3 |

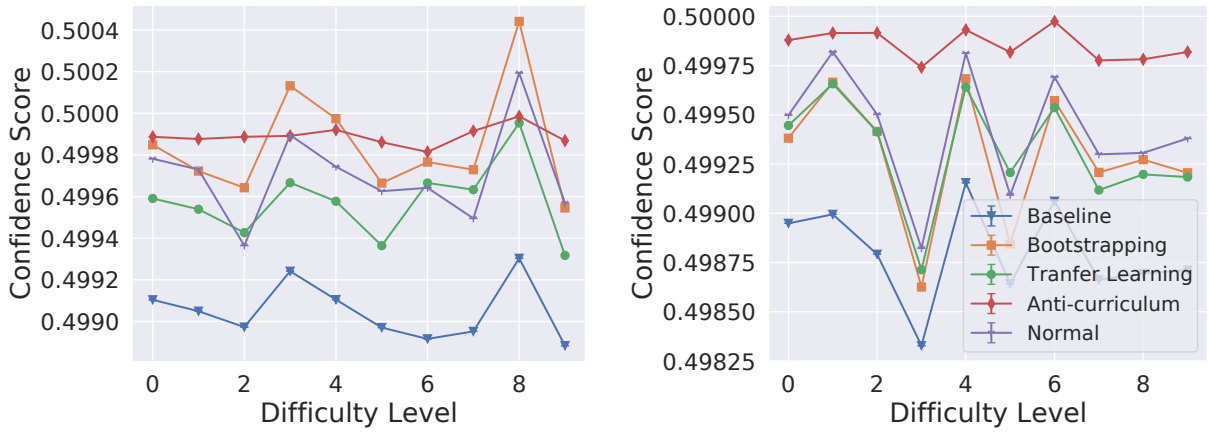
Table 5.7: The average accuracy of MIA (± standard deviation (STD)) on target model trained on CIFAR100 with ResNet-18 and different defense methods. All numbers are in percentage, entry without ± STD means the STD is less than 0.01%.

Table 5.7 demonstrates that DP-SGD is able to curb the MIA accuracy from 90.8% to 50.5%

in average, which is close to random guess (i.e., member or non-member), though at the cost of a significant drop in target model's classification accuracy (from 49.52% to 16.42% in average). This observation is consistent with previous works [149, 143]. We also found DP-SGD is effective against Diff-Cali (e.g., attack accuracy for normal and bootstrapping are dropped to 53.67% and 53.09%). For DP-SGD*, due to the introduced noise, the ranking given by its curriculum is less accurate, but Table 5.7 shows that such change does not impact the MIA accuracy, and the target model accuracy drops by only a small amount (i.e., 0.8% for bootstrapping and 0.7% for baseline) except for anti-curriculum. Due to the noise in ranking, the ranking for anti-curriculum is no longer strictly ordered from difficult to easy. Instead, it becomes more random, thus target accuracy of anti-curriculum is even closer to baseline or bootstrapping. In general, the result suggests using noisy ranking (DP-SGD*) as a defense might not be effective.

For MemGuard, due to its design, NN-based MIA accuracy is fixed to 50% when the defender knows what MIA method is performed by the attacker. In the meantime, the classification task of the target model is not impacted by MemGuard. However, it is not very effective towards label-only attack, as it does not change the label. Our evaluation shows that the overall label-only attack accuracy can still reach up tp 86% even with MemGuard deployed. MixupMMD decreases the MIA accuracy (e.g., 91.4% to 83.1% for bootstrapping), and interestingly, it increases the target model accuracy (e.g., from 51.4% to 54.4% for bootstrapping), which might be attributed to its new regularizer. AdvReg can also increase target accuracy (e.g., 51.4% to 54.2% for bootstrapping) but is less effective in mitigating MIA (e.g., MIA accuracy is even increased from 91.4% to 91.6% for bootstrapping). This observation concurs with a previous work [213].

Given that CL introduces disparate impact on samples under different difficulty groups, we further investigate the relation between difficulty groups and defenses, and we focus on DP-SGD. Figure 5.16 shows that DP-SGD is able to eliminate the disparate impact by CL,




(a) Member of CIFAR100

(b) Non-member of CIFAR100

Figure 5.16: Attack model's confidence score for member and non-member samples of CIFAR-100 trained on ResNet-18 with DP-SGD.

essentially making the difficult samples again hard to attack. We speculate the reason is that DP-SGD adds noise to gradient, which adds randomness to the optimization phase. CL, by introducing a teacher module, reinforces the learning by reducing the randomness. Ultimately, DP-SGD and CL are built on two opposite foundations. Thus, DP-SGD can eliminate the benefit from CL and achieve significant defense effect.

Overall, there is still room for improvement in defenses. Potential future work is to preserve certain properties brought by an ML technique (e.g., fast convergence and higher final performance by CL) and mitigate privacy risks generically.

Finding 6: None of the studied defenses can significantly drop the MIA accuracy while maintaining the target model accuracy. DP-SGD can reverse the impact of CL on MIA.

## 5.6 Discussion

**Limitations.** 1) The research on ML privacy has been growing strong in recent years, and numerous attacks, variations, and defenses have emerged. Admittedly, not all attack methods (e.g., adaptive attack [213] and LiRA [37]) and defense techniques (e.g., PATE [189]) have

been examined. Though LiRA is considered state of the art, it requires multiple shadow models while all other attacks on our paper need one. To fairly compare with LiRA, the current datasets need to be divided into much smaller subsets, which will lead to worse performance of all target models and shadow models. Thus, we didn't investigate LiRA in this work. However, we believe our key conclusions (e.g., the difficult samples become more vulnerable when trained with CL) hold generically, due to the fundamental designs of the curriculum. 2) We mainly evaluated the privacy attack on image and tabular datasets, with widely used models like ResNet and MLP. Admittedly, not all data types (e.g., text [28] and speech [252]) and models (e.g., VGG) are covered. 3) Not all ML privacy attacks are tested, such as model inversion attacks [75, 249], as we suspect they are less likely to be impacted by CL. In the end, we want to mention that our efforts are comparable to recent works that study the privacy of special ML settings like contrastive learning [99].

**Evaluation Metrics.** For privacy attacks like MIA, whether and how it is effective is determined by the evaluation metrics. Attack accuracy is the one adopted in the beginning and is still widely used today, but recent studies have suggested metrics have to be carefully selected to fully understand the results. Following Carlini et al. [37], we adopt TPR at low FPR as another metric. We also view the results under confidence scores to shed light on the divergent impacts of CL into samples, which reveal new insights that are not captured by other metrics. Other metrics like precision/recall [37] and disparate vulnerability [244] can be considered and we believe this research direction still needs new input.

## 5.7 Related Work

**Curriculum Learning (CL).** The idea of CL was first introduced by Bengio et. al [28]. Researchers have then developed many new designs such as predefined CL [121], self-paced CL [120], CL by transfer learning [239] and other automated CL [84]. CL is proved to be effective in the domain of reinforcement learning [165, 178, 74, 73], computer vision [28, 200,

59, 214], natural language processing [28, 216, 254, 87, 155], speech [252, 39, 160], etc. Note that the concept of self-paced[140] learning can often be confused with CL bootstrapping. They share a similar idea of using an iterative procedure to assign higher weights to training examples that have lower costs with respect to their chosen hypothesis. Bootstrapping differs in that the difficulty score is generated based on model accuracy rather than a hypothesis [89].

**Membership Inference Attack (MIA).** Section 5.4.2 has surveyed some representative works about MIA. Here we describe other notable works. On top of the original MIA [207], Salem et al. [203] proposed three more powerful attacks by relaxing the assumptions made by Shokri et al. [207]. Nasr et al. [181] investigated privacy risks in centralized and federated learning scenarios under both black-box and white-box settings. Recent works show that MIA can be further enhanced by adopting flexible thresholds [115], calibrated difficulty level [238], and loss trajectory [158]. Besides the general ML settings, recent works examined special settings like contrastive learning [99, 154], Generative Adversarial Networks (GAN) [103, 44, 46], and Graph Neural Networks (GNN) [96, 98, 240]. However, none of them investigated curriculum learning, and we aim to fill this knowledge gap. To mitigate MIA, researchers have proposed a few defensive mechanisms, like DP-SGD [13], MemGuard [118], MixupMMD [149], and AdvReg [180], as described in Section 5.4.5. PATE [189] uses teacher models to supervise the training of the student model and adds Laplacian noise to the teacher models' output. Salem et al. [203] leverage model stacking and dropout to reduce overfitting.

**Attribute Inference Attack (AIA).** AIA presents another notable threat to ML privacy. Section 5.4.4 surveyed the key works under AIA. In addition, He et al. [99] show that AIA is more vulnerable to models trained by contrastive learning. Recently, Song et al. [211] show that AIA is also effective against language models. Jayaraman et al. propose a new white-box AIA method that achieves better accuracy [114]. We focus on the black-box setting.

**Other Attacks Against ML Models.** MIA and AIA can be considered as attacks on the

data privacy of ML. Model privacy, integrity, and availability have also been investigated, resulting in numerous studies. Model stealing aims to learn the parameters [224, 188, 138, 126, 206] or hyperparameters [232, 186] of a target model, and model inversion, whose goal is to recover the training dataset [75, 249]. There also exists some works focus on protecting a model's ownership [150, 227, 15, 198, 117, 45, 54, 161] to defend against model stealing attacks and other attacks like network pruning and fine-tuning.

# Chapter 6

# Conclusion

**Summary.** In this dissertation, we identify the major gap between differential privacy (DP) theory and its practical applications. We expand DP application into non-standard settings like DNS and resource allocation. Furthermore, we look into the extremely popular domain of machine learning (ML) to explore if existing DP applications in ML domain apply to the emerging ML techniques.

First, we studied the issue of user tracking on DNS data. Based on our observation of the recent attack DSCorr [41], we designed our defense mechanism LDPResolve to make DNS sessions indistinguishable, using a generalized version of ULDP [174] and new constructions satisfying its requirements. We then evaluate the effectiveness of LDPResolve in different settings to prove its capability to protect users' privacy from tracking while preserving the utility for legitimate applications based on DNS data. Our study suggests the threats coming from DNS-based user tracking should be mitigated and it is feasible to protect users' privacy without damaging the utility of legitimate application.

Second, we studied the problem of privacy protection designated under resource allocation and systematically modeled it through the lens of differential privacy. Specifically, we iden-

tified the key issues of a prior system AKR [19]and propose to consider negative noise and mechanisms other than the standard Laplace noise. We designed four different mechanisms, CST, UNI, GEO, and DGEO, proved they all satisfy $\epsilon$-DP.

In both theoretical and empirical analysis, we found our mechanisms outperform AKR in utility ranging from 11% to 65% given a privacy budget $\epsilon$. Among the proposed mechanism, we recommend GEO, which has a good privacy-utility tradeoff and performs especially well when $\epsilon$ is small (e.g., less than 2). Ultimately, we hope to use this work to attract more attention to the privacy issues of resource allocation and encourage new privacy-preserving solutions to be designed.

Last, we performed the first quantitative study to understand how curriculum learning (CL), a widely used technique that accelerates model training, affected the privacy of the trained model. Specifically, we trained target models under 6 image datasets and 3 tabular datasets and performed membership inference attacks (MIA) and attribute inference attacks (AIA) against them to assess the privacy risk in CL. Our results showed that the target model became slightly more vulnerable to MIA but not so under AIA. We also found MIA had a significantly larger impact on samples with high difficulty levels. By exploiting the leakage from difficulty levels, we designed a new MIA, termed Diff-Cali, which achieved similar overall accuracy with much better TPR at low FPR and could infer difficulty samples from normal ML more accurately. Moreover, we evaluated the existing defenses DP-SGD, MemGuard, MixupMMD, and AdvReg in CL settings, and our results showed that they were still effective against the basic MIA.

**Future Directions.** Data provenance, which documents the origins and lifecycle of data, is critical across various fields including healthcare, finance, government, and academic research. Provenance records a comprehensive history of data, from its creation to manipulation and storage, essential for ensuring data integrity, conducting audits, and complying with regulatory standards. System provenance focuses on monitoring dependencies within

computer systems [130, 157], analyzing system call logs to track operations across processes on files and network sockets.

Recent advancements in machine learning (ML) have leveraged low-level system activities captured in provenance graphs to develop ML-based security models, enhancing security monitoring in sensitive networks [92, 93, 94, 170, 233]. However, as the detailed data within provenance graphs traverses through complex ML systems, the systems can inadvertently reveal sensitive information, which could lead to privacy challenges. Studies have shown that provenance-based ML detectors are vulnerable to adversarial attacks [83, 173].

Integrating DP into provenance graphs addresses the critical challenge of maintaining the utility of provenance data while protecting individual privacy. This integration is vital for several reasons:

- **Enhanced Security and Privacy of Provenance-based Applications:** Applying DP to provenance-based applications can prevent attackers from gleaning sensitive information even if they gain access to the models.
- **Regulatory Compliance:** Implementing DP in provenance graphs helps organizations adhere to strict data protection regulations like GDPR and HIPAA by providing a method to manage data that inherently safeguards privacy.
- **Trust and Transparency:** In settings like collaborative research or cross-sectoral analysis where data sharing is essential, using DP-enriched provenance graphs enables entities to share insights without exposing the underlying data, promoting a transparent operational model.

Thus, it is import keep embedding privacy features in provenance graphs as an important future direction. More specifically, exploring this direction will require future research on

1) analyzing privacy aspects in provenance-based applications, 2) defining privacy for provenance graph, and 3) developing techniques for generating private synthetic data.

## 6.1 Perspective

Within the context of our work on DP, we anticipate the following trends in the future: **Privacy in Artificial Intelligence (AI) .** The rapid growth of AI applications, such as GPT-4, has drawn significant attention to their privacy concerns. Recognizing their importance, the White House issued an Executive Order on October 30, 2023, to ensure the safe and responsible development of AI, emphasizing the power of DP [105]. Existing defense mechanisms, such as differentially private stochastic gradient descent (DP-SGD), show significant promise but still have limitations. For instance, the accuracy of models trained with DP-SGD often falls short of expectations. Although recent research suggests that it is feasible to train differentially private machine learning models with high accuracy, this typically involves substantial computational overhead. This overhead can be particularly prohibitive for users with limited resources, posing a significant barrier to the wider adoption of DP-SGD. Therefore, we expect a trend of more research studies in the area of privacy-preserving AI.

**Machine Learning for Privacy.** The trend in using machine learning for enhancing privacy focuses significantly on applications such as synthetic data generation, where ML algorithms are employed to create anonymized datasets that mimic the statistical properties of original data while ensuring individual privacy. This approach is particularly valuable for training ML models where access to real, sensitive data might be restricted or ethically problematic. Techniques like generative adversarial networks (GANs) have shown great promise in generating high-quality synthetic data that can be used for a wide range of purposes, including more secure data sharing and improving model accuracy without compromising personal data. Moreover, machine learning models are also being developed to improve

privacy-preserving techniques such as encryption and differential privacy by optimizing their parameters and making them more efficient and less resource-intensive. As the sophistication of privacy attacks increases, the role of ML in privacy applications is becoming more crucial, driving a substantial shift towards more secure and resilient privacy technologies.

# Bibliography


[1] dnsdist overview. https://dnsdist.org/.

[2] Google Public DNS. https://developers.google.com/speed/public-dns/.

[3] How to enable dns over https in google chrome. https://www.howtogeek.com/660088/how-to-enable-dns-over-https-in-google-chrome/.

[4] Public suffix list. https://publicsuffix.org/.

[5] S.1578 - do not track act. https://www.congress.gov/bill/116th-congress/senate-bill/1578/text.

[6] Welcome to powerdns. https://www.powerdns.com/documentation.html.

[7] Learning with privacy at scale. https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html, 2017.

[8] Alexa top sites. https://aws.amazon.com/alexa-top-sites/, 2019.

[9] Dns pai. http://www.dnspai.com/, 2019.

[10] Doh: (anti-)competitive and network neutrality aspects. https://blog.powerdns.com/2019/12/03/doh-anti-competitive-and-network-neutrality-aspects/, 2019.

[11] Security information exchange (sie) protects from cybercrime. https://www.farsightsecurity.com/solutions/security-information-exchange/, 2019.

[12] Easylist overview. https://easylist.to/, 2021.

[13] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[14] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 674–689, 2014.

[15] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security Symposium (USENIX Security)*, pages 1615–1631. USENIX, 2018.

[16] D. Agrawal and D. Kesdogan. Measuring anonymity: The disclosure attack. *IEEE Security & privacy*, 1(6):27–34, 2003.

[17] M. Almeida, A. Finamore, D. Perino, N. Vallina-Rodriguez, and M. Varvello. Dissecting dns stakeholders in mobile networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 28–34, 2017.

[18] S. Angel, H. Ballani, T. Karagiannis, G. O'Shea, and E. Thereska. End-to-end performance isolation through virtual datacenters. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pages 233–248, 2014.

[19] S. Angel, S. Kannan, and Z. Ratliff. Private resource allocators and their applications. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 372–391. IEEE, 2020.

[20] S. Angel, D. Lazar, and I. Tzialla. What's a little leakage between friends? In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, pages 104–108, 2018.

[21] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster. Keeping the smart home private with smart (er) iot traffic shaping. *Proceedings on Privacy Enhancing Technologies*, 2019(3):128–148, 2019.

[22] M. D. Ayenson, D. J. Wambach, A. Soltani, N. Good, and C. J. Hoofnagle. Flash cookies and privacy ii: Now with html5 and etag respawning. *Available at SSRN 1898390*, 2011.

[23] B. Balle, G. Barthe, and M. Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *Advances in Neural Information Processing Systems*, 31, 2018.

[24] B. Balle, J. Bell, A. Gascón, and K. Nissim. The privacy blanket of the shuffle model. In *Annual International Cryptology Conference*, 2019.

[25] J. Bater, X. He, W. Ehrich, A. Machanavajjhala, and J. Rogers. Shrinkwrap: efficient sql query processing in differentially private data federations. *Proceedings of the VLDB Endowment*, 12(3):307–320, 2018.

[26] A. Beams, S. Kannan, and S. Angel. Packet scheduling with optional client privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3415–3430, 2021.

[27] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[28] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[29] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *Ndss*, pages 1–17, 2011.

[30] J. Black. Learn How Trillions of DNS Requests Help Improve Security. `https://blogs.akamai.com/2018/05/learn-how-trillions-of-dns-requests-help-improve-security.html`.

[31] K. Borgolte, T. Chattopadhyay, N. Feamster, M. Kshirsagar, J. Holland, A. Hounsel, and P. Schmitt. How dns over https is reshaping privacy, performance, and policy in the internet ecosystem. *Performance, and Policy in the Internet Ecosystem (July 27, 2019)*, 2019.

[32] F. Brandt. How to obtain full privacy in auctions. *International Journal of Information Security*, 5(4):201–216, 2006.

[33] Z. Bu, Y.-X. Wang, S. Zha, and G. Karypis. Automatic clipping: Differentially private deep learning made easier and stronger. *arXiv preprint arXiv:2206.07136*, 2022.

[34] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros. Web tracking: Mechanisms, implications, and defenses. *arXiv preprint arXiv:1507.07872*, 2015.

[35] J. Bushart and C. Rossow. Padding ain't enough: Assessing the privacy guarantees of encrypted dns. *arXiv preprint arXiv:1907.01317*, 2019.

[36] C. L. Canonne, G. Kamath, and T. Steinke. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.

[37] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[38] S. Castillo-Perez and J. Garcia-Alfaro. Evaluation of two privacy-preserving protocols for the dns. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 411–416. IEEE, 2009.

[39] A. Caubrière, N. Tomashenko, A. Laurent, E. Morin, N. Camelin, and Y. Estève. Curriculum-based transfer learning for an effective end-to-end spoken language understanding and domain portability. *arXiv preprint arXiv:1906.07601*, 2019.

[40] T. H. Chan, K.-M. Chung, B. M. Maggs, and E. Shi. Foundations of differentially oblivious algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2448–2467. SIAM, 2019.

[41] D. Chang, J. Q. Chen, Z. Li, and X. Li. Hide and seek: Revisiting dns-based user tracking. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 188–205. IEEE, 2022.

[42] H. Chang and R. Shokri. On the privacy risks of algorithmic fairness. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 292–303. IEEE, 2021.

[43] C.-Y. Chen, D. Sanyal, and S. Mohan. Indistinguishability prevents scheduler side channels in real-time systems. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 666–684, 2021.

[44] D. Chen, N. Yu, Y. Zhang, and M. Fritz. GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 343–362. ACM, 2020.

[45] J. Chen, J. Wang, T. Peng, Y. Sun, P. Cheng, S. Ji, X. Ma, B. Li, and D. Song. Copy, Right? A Testing Framework for Copyright Protection of Deep Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.

[46] J. Chen, W. H. Wang, H. Gao, and X. Shi. PAR-GAN: Improving the Generalization of Generative Adversarial Networks Against Membership Inference Attacks. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 127–137. ACM, 2021.

[47] J. Q. Chen. Code of this project. `https://github.com/dpra-dp/dpra`, 2023.

[48] J. Q. Chen, X. He, Z. Li, Y. Zhang, and Z. Li. A comprehensive study of privacy risks in curriculum learning. *arXiv preprint arXiv:2310.10124*, 2023.

[49] J. Q. Chen, T. Wang, Z. Zhang, Y. Zhang, S. Jha, and Z. Li. Differentially private resource allocation. In *Proceedings of the 39th Annual Computer Security Applications Conference*, pages 772–786, 2023.

[50] Y. Chen, M. Antonakakis, R. Perdisci, Y. Nadji, D. Dagon, and W. Lee. Dns noise: Measuring the pervasiveness of disposable domains in modern dns traffic. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 598–609. IEEE, 2014.

[51] H. Choi and H. Lee. Identifying botnets by capturing group activities in dns traffic. *Computer Networks*, 56(1):20–33, 2012.

[52] H. Choi, H. Lee, H. Lee, and H. Kim. Botnet detection by monitoring group activities in dns traffic. In *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, pages 715–720. IEEE, 2007.

[53] C. A. C. Choo, F. Tramèr, N. Carlini, and N. Papernot. Label-Only Membership Inference Attacks. In *International Conference on Machine Learning (ICML)*, pages 1964–1974. PMLR, 2021.

[54] T. Cong, X. He, and Y. Zhang. SSLGuard: A Watermarking Scheme for Self-supervised Learning Pre-trained Encoders. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 579–593. ACM, 2022.

[55] R. Cummings, M. Kearns, A. Roth, and Z. S. Wu. Privacy and truthful equilibrium selection for aggregative games. In *International Conference on Web and Internet Economics*, pages 286–299. Springer, 2015.

[56] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[57] D. Desfontaines. A bottom-up approach to making differential privacy ubiquitous. `https://desfontain.es/privacy/bottom-up-differential-privacy.html`, 03 2022. Ted is writing things (personal blog), Accessed: Oct, 2023.

[58] D. Desfontaines and B. Pejó. Sok: differential privacies. *Proceedings on privacy enhancing technologies*, 2020(2):288–313, 2020.

[59] Y. Duan, H. Zhu, H. Wang, L. Yi, R. Nevatia, and L. J. Guibas. Curriculum deepsdf. In *European Conference on Computer Vision*, pages 51–67. Springer, 2020.

[60] C. Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[61] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[62] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724, 2010.

[63] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[64] P. Eckersley. How unique is your web browser? In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–18. Springer, 2010.

[65] S. Englehardt and A. Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1388–1401, 2016.

[66] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. *arXiv preprint arXiv:1811.12469*, 2018.

[67] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[68] Ú. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[69] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

[70] V. Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[71] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 521–532. IEEE, 2018.

[72] V. Feldman and C. Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

[73] F. Foglino, M. Leonetti, S. Sagratella, and R. Seccia. A gray-box approach for curriculum learning. In *World Congress on Global Optimization*, pages 720–729. Springer, 2019.

[74] P. Fournier, C. Colas, M. Chetouani, and O. Sigaud. Clic: Curriculum learning and imitation for object control in non-rewarding environments. *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

[75] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[76] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 619–633. ACM, 2018.

[77] H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, and H. Duan. An empirical reexamination of global dns behavior. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 267–278, 2013.

[78] P. Garraghan, P. Townend, and J. Xu. An analysis of the server characteristics and resource utilization in google cloud. In *2013 IEEE International Conference on Cloud Engineering (IC2E)*, pages 124–131. IEEE, 2013.

[79] Q. Geng and P. Viswanath. Optimal noise adding mechanisms for approximate differential privacy. *IEEE Transactions on Information Theory*, 62(2):952–969, 2015.

[80] A. Ghorbani and J. Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

[81] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[82] S. Gopi, Y. T. Lee, and L. Wutschitz. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34:11631–11642, 2021.

[83] A. Goyal, X. Han, G. Wang, and A. Bates. Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems. In *30th Network and Distributed System Security Symposium*, 2023.

[84] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. PMLR, 2017.

[85] B. Greschbach, T. Pulls, L. M. Roberts, P. Winter, and N. Feamster. The effect of dns on tor's anonymity. *arXiv preprint arXiv:1609.08187*, 2016.

[86] C. Grier, L. Ballard, J. Caballero, N. Chachra, C. J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis, N. Provos, M. Z. Rafique, M. A. Rajab, C. Rossow, K. Thomas, V. Paxson, S. Savage, and G. M. Voelker. Manufacturing compromise: the emergence of exploit-as-a-service. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 821–832, 2012.

[87] J. Guo, X. Tan, L. Xu, T. Qin, E. Chen, and T.-Y. Liu. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7839–7846, 2020.

[88] G. Hacohen. `https://github.com/GuyHacohen/curriculum_learning`, 2019.

[89] G. Hacohen and D. Weinshall. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*, pages 2535–2544. PMLR, 2019.

[90] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, et al. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, 98(7):751–774, 2016.

[91] Z. Hammoudeh and D. Lowd. Training data influence analysis and estimation: A survey. *arXiv preprint arXiv:2212.04612*, 2022.

[92] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525*, 2020.

[93] X. Han, X. Yu, T. Pasquier, D. Li, J. Rhee, J. Mickens, M. Seltzer, and H. Chen. {SIGL}: Securing software installations through deep graph learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2345–2362, 2021.

[94] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *network and distributed systems security symposium*, 2019.

[95] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[96] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang. Stealing Links from Graph Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pages 2669–2686. USENIX, 2021.

[97] X. He, A. Machanavajjhala, C. Flynn, and D. Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1389–1406. ACM, 2017.

[98] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang. Node-Level Membership Inference Attacks Against Graph Neural Networks. *CoRR abs/2102.05429*, 2021.

[99] X. He and Y. Zhang. Quantifying and Mitigating Privacy Risks of Contrastive Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 845–863. ACM, 2021.

[100] D. Herrmann, C. Banse, and H. Federrath. Behavior-based tracking: Exploiting characteristic patterns in dns traffic. *Computers & Security*, 39:17–33, 2013.

[101] D. Herrmann, C. Gerber, C. Banse, and H. Federrath. Analyzing characteristic host access patterns for re-identification of web user sessions. In *Nordic Conference on Secure IT Systems*, pages 136–154. Springer, 2010.

[102] D. Herrmann, M. Kirchler, J. Lindemann, and M. Kloft. Behavior-based tracking of internet users with semi-supervised learning. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 596–599. IEEE, 2016.

[103] B. Hilprecht, M. Härterich, and D. Bernau. Monte Carlo and Reconstruction Membership Inference Attacks against Generative Models. *Privacy Enhancing Technologies Symposium*, 2019.

[104] N. P. Hoang, I. Lin, S. Ghavamnia, and M. Polychronakis. K-resolver: Towards decentralizing encrypted dns resolution. *arXiv preprint arXiv:2001.08901*, 2020.

[105] W. House. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence, 2023. Accessed: Oct 31, 2023.

[106] R. Houser, Z. Li, C. Cotton, and H. Wang. An investigation on information leakage of dns over tls. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, pages 123–137, 2019.

[107] J. Hsu, Z. Huang, A. Roth, T. Roughgarden, and Z. S. Wu. Private matchings and allocations. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 21–30, 2014.

[108] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for DNS over transport layer security (TLS). Technical report, 2016.

[109] B. Hubert. Centralised doh is bad for privacy, in 2019 and beyond. `https://labs.ripe.net/Members/bert_hubert/centralised-doh-is-bad-for-privacy-in-2019-and-beyond`, 2019.

[110] P. Huffman and P. McManus. Dns queries over https (doh). Technical report, 2018.

[111] H. Hussain, S. U. R. Malik, A. Hameed, S. U. Khan, G. Bickler, N. Min-Allah, M. B. Qureshi, L. Zhang, W. Yongji, N. Ghani, et al. A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11):709–736, 2013.

[112] B. Imana, A. Korolova, and J. Heidemann. Institutional privacy risks in sharing dns data. In *Proceedings of the Applied Networking Research Workshop*, ANRW '21, page 69–75, New York, NY, USA, 2021. Association for Computing Machinery.

[113] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, 2019.

[114] B. Jayaraman and D. Evans. Are attribute inference attacks just imputation? In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1569–1582, 2022.

[115] B. Jayaraman, L. Wang, K. Knipmeyer, Q. Gu, and D. Evans. Revisiting membership inference under realistic assumptions. *arXiv preprint arXiv:2005.10881*, 2020.

[116] B. Jia, H. Hu, Y. Zeng, T. Xu, and Y. Yang. Double-matching resource allocation strategy in fog computing networks based on cost efficiency. *Journal of Communications and Networks*, 20(3):237–246, 2018.

[117] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot. Entangled Watermarks as a Defense against Model Extraction. In *USENIX Security Symposium (USENIX Security)*, pages 1937–1954. USENIX, 2021.

[118] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 259–274. ACM, 2019.

[119] R. Jia, F. Wu, X. Sun, J. Xu, D. Dao, B. Kailkhura, C. Zhang, B. Li, and D. Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8239–8247, 2021.

[120] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[121] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2018.

[122] J. Jin, E. McMurtry, B. I. Rubinstein, and O. Ohrimenko. Are we there yet? timing and floating-point attacks on differential privacy systems. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 473–488. IEEE, 2022.

[123] R. Johari and J. N. Tsitsiklis. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research*, 29(3):407–435, 2004.

[124] L. Kang and D. C. Parkes. Passive verification of the strategyproofness of mechanisms in open environments. In *Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet*, pages 19–30, 2006.

[125] S. Kannan, J. Morgenstern, A. Roth, and Z. S. Wu. Approximately stable, school optimal, and student-truthful many-to-one matchings (via differential privacy). In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1890–1903. SIAM, 2014.

[126] S. Kariyappa, A. Prakash, and M. K. Qureshi. MAZE: Data-Free Model Stealing Attack Using Zeroth-Order Gradient Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13814–13823. IEEE, 2021.

[127] M. Kearns, M. Pai, A. Roth, and J. Ullman. Mechanism design in large games: Incentives and privacy. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 403–410, 2014.

[128] D. W. Kim and J. Zhang. You are how you query: Deriving behavioral fingerprints from dns traffic. In *International Conference on Security and Privacy in Communication Systems*, pages 348–366. Springer, 2015.

[129] D. W. Kim and J. Zhang. Deriving and measuring dns-based fingerprints. *Journal of Information Security and Applications*, 36:32–42, 2017.

[130] S. T. King and P. M. Chen. Backtracking intrusions. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 223–236, 2003.

[131] E. Kinnear, T. Pauly, and C. Wood. Adaptive dns: Improving privacy of name resolution. Technical report, Technical report, 2019.

[132] M. Kirchler, D. Herrmann, J. Lindemann, and M. Kloft. Tracked without a trace: linking sessions of users by unsupervised learning of patterns in their dns traffic. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, pages 23–34, 2016.

[133] A. Klein and B. Pinkas. Dns cache-based user tracking. In *Network and Distributed System Security Symposium (NDSS'19). San Diego, CA, USA (Feb 2019)*, 2019.

[134] K. C. Knowlton. A fast storage allocator. *Communications of the ACM*, 8(10), 1965.

[135] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[136] I. Komargodski and E. Shi. Differentially oblivious turing machines. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[137] J. K. Konjaang, J. Maipan-uku, and K. K. Kubuga. An efficient max-min resource allocator and task scheduling algorithm in cloud computing environment. *arXiv preprint arXiv:1611.08864*, 2016.

[138] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer. Thieves on Sesame Street! Model Extraction of BERT-based APIs. In *International Conference on Learning Representations (ICLR)*, 2020.

[139] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[140] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*, volume 1, page 2, 2010.

[141] M. Kumpošt and V. Matyáš. User profiling and re-identification: case of university-wide network analysis. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 1–10. Springer, 2009.

[142] Y.-H. Kuo, C.-C. Chiu, D. Kifer, M. Hay, and A. Machanavajjhala. Differentially private hierarchical count-of-counts histograms. *arXiv preprint arXiv:1804.00370*, 2018.

[143] A. Kurakin, S. Chien, S. Song, R. Geambasu, A. Terzis, and A. Thakurta. Toward training at imagenet scale with differential privacy. *arXiv preprint arXiv:2201.12328*, 2022.

[144] D. Lazar, Y. Gilad, and N. Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 711–725, 2018.

[145] D. Lazar and N. Zeldovich. Alpenhorn: Bootstrapping secure communication without leaking metadata. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 571–586, 2016.

[146] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[147] J. Lee, J. Kwon, H.-J. Shin, and H. Lee. Tracking multiple c&c botnets by analyzing dns traffic. In *2010 6th IEEE Workshop on Secure Network Protocols*, pages 67–72. IEEE, 2010.

[148] D. Li, M. Rhu, D. R. Johnson, M. O'Connor, M. Erez, D. Burger, D. S. Fussell, and S. W. Redder. Priority-based cache allocation in throughput processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 89–100. IEEE, 2015.

[149] J. Li, N. Li, and B. Ribeiro. Membership inference attacks and defenses in classification models. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 5–16, 2021.

[150] Z. Li, C. Hu, Y. Zhang, and S. Guo. How to Prove Your Model Belongs to You: A Blind-Watermark based Framework to Protect Intellectual Property of DNN. In *Annual Computer Security Applications Conference (ACSAC)*, pages 126–137. ACM, 2019.

[151] Z. Li and Y. Zhang. Membership Leakage in Label-Only Exposures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 880–895. ACM, 2021.

[152] B. Liu, Z. Li, P. Zong, C. Lu, H. Duan, Y. Liu, S. Alrwais, X. Wang, S. Hao, Y. Jia, Y. Zhang, K. Chen, and Z. Zhang. Traffickstop: Detecting and measuring illicit traffic monetization through large-scale dns analysis. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 560–575. IEEE, 2019.

[153] D. Liu, Z. Li, K. Du, H. Wang, B. Liu, and H. Duan. Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 537–552, 2017.

[154] H. Liu, J. Jia, W. Qu, and N. Z. Gong. EncoderMI: Membership Inference against Pre-trained Encoders in Contrastive Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021.

[155] J. Liu, Y. Ren, X. Tan, C. Zhang, T. Qin, Z. Zhao, and T.-Y. Liu. Task-level curriculum learning for non-autoregressive neural machine translation. *arXiv preprint arXiv:2007.08772*, 2020.

[156] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876, 2021.

[157] Y. Liu, M. Zhang, D. Li, K. Jee, Z. Li, Z. Wu, J. Rhee, and P. Mittal. Towards a timely causality analysis for enterprise security. In *NDSS*, 2018.

[158] Y. Liu, Z. Zhao, M. Backes, and Y. Zhang. Membership Inference Attacks by Exploiting Loss Trajectory. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2085–2098. ACM, 2022.

[159] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen. Understanding Membership Inferences on Well-Generalized Learning Models. *CoRR abs/1802.04889*, 2018.

[160] R. Lotfian and C. Busso. Curriculum learning for speech emotion recognition from crowdsourced labels. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4):815–826, 2019.

[161] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum. SoK: How Robust is Image Classification Deep Neural Network Watermarking? In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.

[162] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM Computer Communication Review*, 32(3):62–73, 2002.

[163] N. Mallesh and M. Wright. The reverse statistical disclosure attack. In *International Workshop on Information Hiding*, pages 221–234. Springer, 2010.

[164] C. Mastroianni, M. Meo, and G. Papuzzo. Self-economy in cloud data centers: Statistical assignment and migration of virtual machines. In *European Conference on Parallel Processing*, pages 407–418. Springer, 2011.

[165] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[166] S. Mazloom and S. D. Gordon. Secure computation with differentially private access patterns. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 490–507, 2018.

[167] K. McCarthy. Internet overseer icann loses a third time in whois gdpr legal war. `https://www.theregister.co.uk/2018/08/07/icann_whois_gdpr/`, 2018.

[168] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 497–512. IEEE, 2019.

[169] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[170] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1137–1152. IEEE, 2019.

[171] F. Mireshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and H. Esmaeilzadeh. Privacy in deep learning: A survey. *arXiv preprint arXiv:2004.12254*, 2020.

[172] I. Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[173] K. Mukherjee, J. Wiedemeier, T. Wang, J. Wei, F. Chen, M. Kim, M. Kantarcioglu, and K. Jee. Evading {Provenance-Based}{ML} detectors with adversarial system actions. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1199–1216, 2023.

[174] T. Murakami and Y. Kawamoto. Utility-optimized local differential privacy mechanisms for distribution estimation. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1877–1894, 2019.

[175] A. Nahir, A. Orda, and D. Raz. Resource allocation and management in cloud computing. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1078–1084. IEEE, 2015.

[176] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139, 1999.

[177] A. Narayan and A. Haeberlen. Djoin: Differentially private join queries over distributed databases. In *10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)*, pages 149–162, 2012.

[178] S. Narvekar and P. Stone. Learning curriculum policies for reinforcement learning. *arXiv preprint arXiv:1812.00285*, 2018.

[179] M. Nasr, A. Bahramali, and A. Houmansadr. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1962–1976, 2018.

[180] M. Nasr, R. Shokri, and A. Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 634–646. ACM, 2018.

[181] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1021–1035. IEEE, 2019.

[182] M. Nasr, S. Songi, A. Thakurta, N. Papernot, and N. Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*. IEEE, 2021.

[183] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[184] T. T. Nguyên, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin. Collecting and analyzing data from smart device users with local differential privacy. *arXiv preprint arXiv:1606.05053*, 2016.

[185] J. Oberheide, M. Karir, and Z. M. Mao. Characterizing dark dns behavior. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 140–156. Springer, 2007.

[186] S. J. Oh, M. Augustin, B. Schiele, and M. Fritz. Towards Reverse-Engineering Black-Box Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[187] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais. Detection of early-stage enterprise infection by mining large-scale log data. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 45–56. IEEE, 2015.

[188] T. Orekondy, B. Schiele, and M. Fritz. Knockoff Nets: Stealing Functionality of Black-Box Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4954–4963. IEEE, 2019.

[189] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable Private Learning with PATE. In *International Conference on Learning Representations (ICLR)*, 2018.

[190] V. Paxson, M. Christodorescu, M. Javed, J. Rao, R. Sailer, D. L. Schales, M. Stoecklin, K. Thomas, W. Venema, and N. Weaver. Practical comprehensive bounds on surreptitious communication over {DNS}. In *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, pages 17–32, 2013.

[191] Pytorch. Opacus RDP. `https://github.com/pytorch/opacus/blob/main/opacus/accountants/rdp.py`, 2022.

[192] L. Qin, R. Jayaram, E. Shi, Z. Song, D. Zhuo, and S. Chu. Adore: Differentially oblivious relational database operators. *Proceedings of the VLDB Endowment*, 16(4):842–855, 2022.

[193] R. Radu and M. Hausding. Consolidation in the dns resolver market – how much, how fast, how dangerous? *Journal of Cyber Policy*, 5(1):46–64, 2020.

[194] B. Rahbarinia, R. Perdisci, and M. Antonakakis. Segugio: Efficient behavior-based tracking of malware-control domains in large isp networks. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 403–414. IEEE, 2015.

[195] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 155–168, 2012.

[196] R. M. Rogers and A. Roth. Asymptotically truthful equilibrium selection in large congestion games. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 771–782, 2014.

[197] A. Roth. Differential privacy, equilibrium, and efficient allocation of resources. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1593–1597. IEEE, 2013.

[198] B. D. Rouhani, H. Chen, and F. Koushanfar. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *CoRR abs/1804.00750*, 2018.

[199] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[200] C. Sakaridis, D. Dai, and L. V. Gool. Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7374–7383, 2019.

[201] N. Saleheen, S. Chakraborty, N. Ali, M. M. Rahman, S. M. Hossain, R. Bari, E. Buder, M. Srivastava, and S. Kumar. msieve: differential behavioral privacy in time series of mobile sensor data. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 706–717, 2016.

[202] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang. Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning. In *USENIX Security Symposium (USENIX Security)*, pages 1291–1308. USENIX, 2020.

[203] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[204] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520. IEEE, 2018.

[205] P. Schmitt, A. Edmundson, A. Mankin, and N. Feamster. Oblivious dns: Practical privacy for dns queries. *Proceedings on Privacy Enhancing Technologies*, 2019(2):228–244, 2019.

[206] Y. Shen, X. He, Y. Han, and Y. Zhang. Model Stealing Attacks Against Inductive Graph Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1175–1192. IEEE, 2022.

[207] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*, pages 3–18. IEEE, 2017.

[208] I. Shumailov, Z. Shumaylov, D. Kazhdan, Y. Zhao, N. Papernot, M. A. Erdogdu, and R. J. Anderson. Manipulating sgd with data ordering attacks. *Advances in Neural Information Processing Systems*, 34:18021–18032, 2021.

[209] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso. Encrypted dns–¿ privacy? a traffic analysis perspective. *arXiv preprint arXiv:1906.09682*, 2019.

[210] R. Singel. Netflix cancels recommendation contest after privacy lawsuit. *Retrieved March*, 29:2018, 2010.

[211] C. Song and A. Raghunathan. Information Leakage in Embedding Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 377–390. ACM, 2020.

[212] C. Song and V. Shmatikov. Overlearning Reveals Sensitive Attributes. In *International Conference on Learning Representations (ICLR)*, 2020.

[213] L. Song and P. Mittal. Systematic Evaluation of Privacy Risks of Machine Learning Models. In *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.

[214] P. Soviany, C. Ardei, R. T. Ionescu, and M. Leordeanu. Image difficulty curriculum for generative adversarial networks (cugan). In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3463–3472, 2020.

[215] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe. Curriculum learning: A survey. *arXiv preprint arXiv:2101.10382*, 2021.

[216] V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. Baby steps: How "less is more" in unsupervised dependency parsing. 2009.

[217] J. M. Spring and C. L. Huth. The impact of passive dns collection on end-user privacy. 2012.

[218] T. Steinke and J. Ullman. Open problem - avoiding the union bound for multiple queries. `https://differentialprivacy.org/open-problem-avoid-union/`, 2021.

[219] M. Sun, G. Xu, J. Zhang, and D. W. Kim. Tracking you through dns traffic: Linking user sessions by clustering with dirichlet mixture model. In *Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, pages 303–310, 2017.

[220] Y. Sun, K. Jee, S. Sivakorn, Z. Li, C. Lumezanu, L. Korts-Parn, Z. Wu, J. Rhee, C. H. Kim, M. Chiang, and P. Mittal. Detecting malware injection with program-dns behavior. In *2020 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 552–568, 2020.

[221] R. Sundar. https://github.com/rsundar96/curriculum-learning-acceleration, 2020.

[222] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[223] P. Thaker, M. Budiu, P. Gopalan, U. Wieder, and M. Zaharia. Overlook: Differentially private exploratory visualization for big data. *arXiv preprint arXiv:2006.12018*, 2020.

[224] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium (USENIX Security)*, pages 601–618. USENIX, 2016.

[225] Trusted-AI. https://github.com/Trusted-AI/adversarial-robustness-toolbox, 2023.

[226] N. Tyagi, Y. Gilad, D. Leung, M. Zaharia, and N. Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 423–440, 2017.

[227] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh. Embedding Watermarks into Deep Neural Networks. In *International Conference on Multimedia Retrieval (ICMR)*, pages 269–277. ACM, 2017.

[228] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[229] N. V. Verde, G. Ateniese, E. Gabrielli, L. V. Mancini, and A. Spognardi. No nat'd user left behind: Fingerprinting users behind nat from netflow records alone. In *2014 IEEE 34th International Conference on Distributed Computing Systems*, pages 218–227. IEEE, 2014.

[230] VMware. Managing resource pools. https://docs.vmware.com/en/VMware-vSphere/8.0/vsphere-resource-management/GUID-60077B40-66FF-4625-934A-641703ED7601.html, 2019.

[231] S. Wagh, P. Cuff, and P. Mittal. Differentially private oblivious ram. *Proceedings on Privacy Enhancing Technologies*, 4:64–84, 2018.

[232] B. Wang and N. Z. Gong. Stealing Hyperparameters in Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 36–52. IEEE, 2018.

[233] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter, et al. You are what you do: Hunting stealthy malware via data provenance analysis. In *NDSS*, 2020.

[234] T. Wang. High precision open-world website fingerprinting. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 152–167. IEEE, 2020.

[235] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 729–745, 2017.

[236] X. Wang, Y. Chen, and W. Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[237] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[238] L. Watson, C. Guo, G. Cormode, and A. Sablayrolles. On the importance of difficulty calibration in membership inference attacks. In *International Conference on Learning Representations (ICLR)*, 2022.

[239] D. Weinshall, G. Cohen, and D. Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*, pages 5238–5246. PMLR, 2018.

[240] F. Wu, Y. Long, C. Zhang, and B. Li. LinkTeller: Recovering Private Edges from Graph Neural Networks via Influence Analysis. In *IEEE Symposium on Security and Privacy (S&P)*, pages 2005–2024. IEEE, 2022.

[241] X. Wu, E. Dyer, and B. Neyshabur. When do curricula work? In *International Conference on Learning Representations*, 2021.

[242] Q. Xiao, M. K. Reiter, and Y. Zhang. Mitigating storage side channels using statistical privacy mechanisms. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1582–1594, 2015.

[243] M. Xu, A. Papadimitriou, A. Feldman, and A. Haeberlen. Using differential privacy to efficiently mitigate side channels in distributed analytics. In *Proceedings of the 11th European Workshop on Systems Security*, pages 1–6, 2018.

[244] M. Yaghini, B. Kulynych, and C. Troncoso. Disparate Vulnerability: on the Unfairness of Privacy Attacks Against Machine Learning. *CoRR abs/1906.00389*, 2019.

[245] D. Yang, D. Zhang, and B. Qu. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(3):1–23, 2016.

[246] Y. Yarom and K. Falkner. Flush+ reload: A high resolution, low noise, l3 cache side-channel attack. In *USENIX Security Symposium 2014*, 2014.

[247] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 199–208, 2013.

[248] X. Zhang, J. Hamm, M. K. Reiter, and Y. Zhang. Statistical privacy for streaming traffic. In *Proceedings of the 26th ISOC Symposium on Network and Distributed System Security*, 2019.

[249] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 253–261, 2020.

[250] Z. Zhang, Y. Song, and H. Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017.

[251] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier. A survey on malicious domains detection through dns data analysis. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.

[252] S. Zheng, G. Liu, H. Suo, and Y. Lei. Autoencoder-based semi-supervised curriculum learning for out-of-domain speaker verification. *System*, 3:98, 2019.

[253] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[254] Y. Zhou, B. Yang, D. F. Wong, Y. Wan, and L. S. Chao. Uncertainty-aware curriculum learning for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6934–6944, 2020.

[255] S. Zimmeck, J. S. Li, H. Kim, S. M. Bellovin, and T. Jebara. A privacy analysis of cross-device tracking. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1391–1408, 2017.