UNIVERSITY OF CALIFORNIA,
IRVINE

Federated Learning for Mobile Data Privacy

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Networked Systems

by

Evita Bakopoulou

Dissertation Committee:
Professor and Chancellor's Fellow Athina Markopoulou, Chair
Chancellor's Professor Carter T. Butts
Assistant Professor Yanning Shen

2021

DEDICATION

To my loving husband, Konstantinos Pieros, who put his own profession and dreams on hold to help me achieve mine. You have been the best source of constant support, patience, and encouragement through the challenges of my PhD studies.
To my family for always believing in me.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

ACKNOWLEDGMENTS

I would like to thank those who made this thesis possible. First and foremost, I am deeply grateful to my advisor, Professor Athina Markopoulou, for providing me this life-changing opportunity. I thank her for the continuous support, guidance and mentorship during my PhD journey.

Second, I would like to extend my sincere thanks to the members of my doctoral committee - Professor Carter T. Butts and Professor Yanning Shen, and for their insightful comments and suggestions during my PhD studies. I would also like to thank Professors Gene Tsudik, Scott Jordan and Sameer Singh for serving on my candidacy committee. I would like to thank my co-authors: Dr. Anastasia Shuba, Dr. Balint Tillman, Professor Konstantinos Psounis, Justin Ley, and Jiang Zhang. I would also like to thank my labmates who became dear friends during my years at UCI: Dr. Balint Tillman, Dr. Anastasia Shuba, Dr. Emmanouil Alimpertis, Milad Asgari, Janus Varmarken, Hieu Le, for their support and all the brainstorming sessions. A special thanks goes to Dr. Tillman for his friendship and mentorship: his guidance was key in the early stages of the FedPacket project. Another special thanks goes to Dr. Alimpertis for his friendship and mentorship on the signal maps problem: without him introducing me to the problem, the Federated Signal Maps project would not be possible. I also thank the newest members of our lab: Dr. Rahmadi Trimananda, Olivia Figueira, Jad Aaraj, Hao Cui and Mengwei Yang. A thank you is in order to Professor Vana Kalogeraki for introducing me to research and for advising me during my Bachelor's and Master's theses at Athens University of Economics and Business. I also thank my internship mentors: Dr. Paul Rigor, Dr. Harkeerat Bedi, Alejo Grigera Sutro, and Jessica Wong.

I have greatly appreciated all the funding support from the University of California, Irvine through the Networked Systems Fellowship, Henry Samueli Fellowship, a UCI Proof-of-Concept Award in 2017; the Broadcom Foundation Fellowship; Gerondelis Foundation Graduate Fellowship; the National Science Foundation Awards 1649372, 1815666, 1900654, 1649372 and 1526736, and Bell Labs, Oath/Verizon Digital Media Services and Google for the summer internships. I would like to thank E. Alimpertis, A. Shuba, and M. Gjoka, former members of our group, for the UCI Campus LTE, NoMoAds and in-house datasets, used for evaluation in this thesis. I also thank IEEE for granting me permission to use previously published work in this dissertation. Portion of this dissertation's text is a reprint of the material as it appears in Evita Bakopoulou, Balint Tillman, and Athina Markopoulou: "FedPacket: A Federated Learning Approach to Mobile Packet Classification", IEEE Transactions on Mobile Computing, 2021.

I am thankful to my family and friends for believing in me and providing me support and encouragement during challenging times. Thank you Maria Oikonomou, you have always supported me since high school. Lyn Lee, I am so glad we met at the Broadcom workshop and I can call you my friend ever since. Thank you for always cheering me up. A thank you is in order to Dr. Georgios Detorakis, for being always there for me as a friend and a mentor, and for our discussions on machine learning, science, cats and everything in between. I owe another special thanks to Konstantinos Pieros for believing in me and being my rock for more than 11 years and counting. Finally, I would like to thank my non-humans friends, my cats Besito, Scratch and late Otisep, for their emotional support and calming purring in the last 10 years.

VITA

Evita Bakopoulou

EDUCATION

Doctor of Philosophy in Networked Systems	**2021**
University of California, Irvine	*Irvine, CA, USA*
Master of Science in Networked Systems	**2021**
University of California, Irvine	*Irvine, CA, USA*
Master of Science in Computer Science	**2016**
Athens University of Economics and Business	*Athens, Greece*
Bachelor of Science in Computer Science	**2014**
Athens University of Economics and Business	*Athens, Greece*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2016–2021**
University of California, Irvine	*Irvine, CA, USA*
Research Assistant	**06/2016–07/2016**
Athens University of Economics and Business	*Athens, Greece*

TEACHING EXPERIENCE

Teaching Assistant	**Fall 2017**
University of California Irvine	*Irvine, CA, USA*
Teaching Assistant	**Spring 2015**
Athens University of Economics and Business	*Athens, Greece*

INTERNSHIPS

Privacy Engineer Intern	**Summer 2020**
Google	*Sunnyvale, CA, USA*
Research Intern, Security	**Summer 2018**
Oath/Verizon Digital Media Services	*Los Angeles, CA, USA*
Summer Intern	**Summer 2017**
Bell Labs	*Sunnyvale, CA, USA*

REFEREED JOURNAL PUBLICATIONS

E. Bakopoulou, B. Tillman, and A. Markopoulou. “FedPacket: A Federated Learning Approach to Mobile Packet Classification”. *IEEE Transactions on Mobile Computing*, 2021.

REFEREED CONFERENCE PUBLICATIONS

A. Shuba, E. Bakopoulou, and A. Markopoulou. “Privacy Leak Classification on Mobile Devices”. *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018.

REFEREED POSTER PRESENTATIONS

E. Bakopoulou, P. Thanopoulos, I. Boutsis, and V. Kalogeraki. “ICU: A Tool for Intent Filtering on Android devices”. *9th International ACM Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, 2016.

TECHNICAL REPORTS

E. Bakopoulou, J. Zhang, J. Ley, K. Psounis, and A. Markopoulou, “Location Leakage in Federated Signal Maps”. *in preparation*, 2021.

E. Bakopoulou, A. Shuba, and A. Markopoulou, “Exposures Exposed: A Measurement and User Study to Assess Mobile Data Privacy in Context”. *arXiv preprint arXiv:2008.08973*, 2020.

E. Bakopoulou, B. Tillman, and A. Markopoulou, “A Federated Learning Approach for Mobile Packet Classification.” *arXiv preprint arXiv:1907.13113*, 2019.

ABSTRACT OF THE DISSERTATION

Federated Learning for Mobile Data Privacy

By

Evita Bakopoulou

Doctor of Philosophy in Networked Systems

University of California, Irvine, 2021

Professor and Chancellor's Fellow Athina Markopoulou, Chair

Mobile devices have access to rich personal, potentially sensitive data, from online activity and multiple sensors that include personally identifiable information (PII), such as user identifiers, device identifiers, location, health data etc. Mobile crowdsourcing (MCS) is a prevalent practice today: a large number of mobile devices upload measurements to a server, often including the location where they were collected. This data is used to provide various services (including spatiotemporal maps of cellular/WiFi coverage, sentiment, occupancy, COVID-related information etc.) but also poses privacy threats due to untrusted servers and/or third party sharing. In this thesis, first, we design and launch a user study in order to better understand of not just the extent of personally identifiable information (PII) exposure, but also its context (i.e., functionality of the app, destination server, encryption used, etc.) and the risk perceived by mobile users today on a real-world dataset from 400 apps. Next, we consider two applications of learning from mobile crowdsourced data, using the Federated Learning (FL) framework: FL improves privacy by allowing mobile devices to collaboratively train a global model, while keeping their training data local and only exchanging model parameters with the server.

First, we consider training classifiers that predict PII exposures and advertising requests in mobile data packets, and use them to block those packets on mobile devices. While such classifiers have been previously trained in a centralized way, we propose, for the first time, a federated packet classification framework and we demonstrate its effectiveness in terms of classification performance, communication and computation cost via evaluation on three real-world datasets. Methodological challenges include model and feature selection, and tuning the federated learning parameters. We also design, for the first time, two privacy attacks based on HTTP features for an honest-but-curious server and demonstrate one mitigation approach, where the aggregation of sufficient users can limit the attack's effectiveness.

Second, we consider training mobile signal strength maps based on crowdsourced measurements. State-of-the-art trained centralized models using location and other features to predict signal strength. In this work, we apply online federated learning to this problem, since mobile users move around and collect measurements over time. We consider a Deep Leakage from Gradients (DLG) attack, where an honest-but-curious server can infer information about an individual user's trajectory based on its gradient updates. Such DLG attacks have been studied before only for image/text data and are applied for the first time in this setting. We evaluate the effect of various FL parameters, we show that averaging of gradients provides some protection against such DLG attacks in our setting, and we propose an algorithm that can further improve the privacy-utility tradeoff by selecting which data to include in a batch and use for local training.

Chapter 1

Introduction

1.1 Motivation

There is recently increased public awareness and concern about how sensitive information available on mobile devices is shared and tracked. In particular, mobile apps and third party libraries (including developer, tracking and advertising libraries) routinely send such information (*i.e.,* personally identifiable information or “PII”, sensory data, user activity) to servers, typically without the user being aware or in control of this information flow. Users often share their mobile data, e.g., via crowdsourcing, which is a common practice and useful in many contexts, *i.e.,* gaining insights based on mobility patterns, etc. Explicitly or implicitly obtaining users’ mobile data introduces privacy risks due to the sensitive nature of such data. In explicit mobile tracking, a large number of mobile applications and third-party libraries collect information on the mobile device and transmit it over the network to remote servers (examples include app developer servers, advertising and third party servers). Tracking might be necessary for the functionality of the apps or may be happening for monetization of user data, for example through targeted advertising. Efforts to increase online data transparency include landmark privacy laws (such as GDPR [9] and CCPA

[7]) as well as technical approaches (*e.g.,* permissions [101], static and dynamic analysis [89, 57], and network-based approaches [111, 115, 122, 121]).

In this thesis, first, we design and launch a user study in order to better understand of not just the extent of PII exposure, but also its context (i.e., functionality of the app, destination server, encryption used, etc.) and the risk perceived by mobile users today on a real-world dataset from 400 apps. In implicit tracking, a common example is malicious actors getting access to crowdsourced locations from devices and obtaining the whereabouts of users and thus, breaking their privacy. In order to limit mobile tracking, the devices can stop sharing their data but this negatively impacts the utility or they can use privacy-preserving techniques.

Mobile crowdsourcing is widely used to collect data from mobile devices at a large scale, which is used to train machine learning models for properties of interest, such as cellular/WiFi. Leveraging data from mobile devices in order to train machine learning models for a task of interest increases utility but introduces privacy risk due to the sensitive nature of the data that can enable tracking. On one hand, the mobile devices could train local models without sharing their data, but the local models will have low utility due to limited data available on the device. On the other hand, the devices can share their data with a server who collects it and trains a more powerful centralized model at the expense of the privacy of mobile devices. Federated learning (FL) is an approach to collaboratively train machine learning models and it combines the best from both worlds. It increases utility compared to local training and it increases privacy compared to centralized training, while maintaining utility by allowing to keep data on the devices; the devices and server exchange only model parameters and not raw training data. Although FL raises the privacy bar, the gradients of model parameters can leak information about the local data of devices that participate in training, which enables tracking via inference. In this thesis we consider two applications of FL for mobile data and we evaluate the corresponding privacy attacks based on gradients (originally developed to reconstruct training data of DNN image classifiers) for an honest-but-curious server who observes gradients and aims to reconstruct the private data of a participating device.

First, we propose for the first time, a federated learning approach for two packet classification tasks (*i.e.,* to predict PII exposure or ad request in individual packets). In order to limit explicit mobile tracking, machine learning approaches have been proposed for PII detection in mobile packets. State-of-the-art approaches use features extracted from HTTP packets and train packet classifiers in a centralized way: users collect and label network packets on their mobile devices, then upload data to a central server; the server uses the data contributed by all users to train a packet classifier. We propose FedPacket, a federated learning framework for mobile packet classification that raises the privacy bar and demonstrate its effectiveness in terms of classification performance, communication and computation cost via evaluation on three real-world datasets. Moreover, we design and launch, for the first time, two privacy attacks based on HTTP features and we show that it is possible to infer the browsing history of a user based on HTTP features leakage.

The second application of FL is on the cellular signal strength prediction problem, where the users collaboratively train a regression Deep Neural Network (DNN) that predicts the cellular signal strength given location features without sharing their location data. In this application, users may not want to share their measurements to enable centralized training, as they contain sensitive information like device information, trajectories etc. To that end, we propose online federated signal maps, where the data on each device becomes available in online fashion and we evaluate the leakage from gradients specifically on this problem. Such Deep Leakage from Gradients (DLG) attacks have been demonstrated so far only on image/text data with limited evaluation on FL parameters, and we demonstrate their effectiveness for the first time in online federated signal maps where it is possible to reconstruct the users' locations over time via observing gradients. Finally, we propose a mitigation algorithm that chooses which locations to put in a batch without hurting utility.

1.2 Contributions

In this thesis, we make the following contributions.

1.2.1 Exposures Exposed: A Measurement and User Study to Assess Mobile Data Privacy in Context

In Chapter 3 and in [30], we are interested in better understanding not only the extent of personally identifiable information (PII) exposure, but also its *context* (*i.e.,* functionality of the app, destination server, encryption used, *etc.*) and the risk perceived by mobile users today. To that end, we take two steps. First, we perform a *measurement study*: we analyze the AntShield dataset that was generated via manual and automatic testing and captured the exposure of 16 PII types from 400 most popular Android apps. We analyze these exposures and provide insights into the extent and patterns of mobile apps sharing PII, which can be later used for prediction and prevention. Second, we perform a *user study* with 220 participants on Amazon Mechanical Turk: we summarize the results of the measurement study in categories, present them in a realistic context, and assess users' understanding, concern, and willingness to take action. To the best of our knowledge, our user study is the first to collect and analyze user input in such fine granularity and on actual (not just potential or permitted) privacy exposures on mobile devices. Although many users did not initially understand the full implications of their PII being exposed, after being better informed through the study, they became appreciative and interested in better privacy practices.

1.2.2 FedPacket: Federated Packet Classification

In Chapter 4 and in [32], we propose a federated learning approach to mobile packet classification, which enables devices to collaboratively train a global model, without uploading the training data collected on devices. We apply our framework to two packet classification tasks (*i.e.,* to predict PII exposure or ad requests in individual packets) and we demonstrate its effectiveness in terms of classification performance, communication and computation cost, using three real-world datasets. Methodological challenges we address in the process include model and feature selection, as well as tuning the federated learning parameters specifically for our packet classification tasks. We extend FedPacket further and demonstrate for the first time two privacy attacks for an honest-but-curious server, based on gradients that reveal information about user data (HTTP(s) packet traces). First, we show the attack that reconstructs HTTP features of the user. Second, we shown how the recovered features can reveal user's visited domains or browsing history. Finally, we demonstrate one mitigation approach via aggregation of multiple users gradients and how the aggregation of sufficient users can be used to limit the attack's effectiveness [32].

1.2.3 Federated Signal Maps and Location Leakage

In Chapter 5 and in [33], we present our second application of FL to mobile data. We consider the problem of signal maps prediction (i.e., training a machine learning model that predicts cellular performance based on location and other features) using measurements collected from different mobile devices. We formulate the problem within the online federated learning framework: (i) federated learning enables users to collaboratively train a model, while keeping their training data on their devices; (ii) measurements are collected as users move around over time and are used for local training in an online fashion. Without additional privacy enhancing mechanisms, federated learning is well-known to be vulnerable to privacy attacks. We consider an honest-but-curious server, who observes the updates from target users participating in federated learning and infers their location using a deep leakage from gradients (DLG) type-of-attack, originally developed to reconstruct training data of DNN image classifiers. We make the key observation that a DLG attack, applied to our setting, infers the average location of a batch of local data, and can thus be used to reconstruct the target users' trajectory at a coarse granularity. We show that a moderate level of privacy protection is already offered by the averaging of gradients, which is inherent in FedAvg and controlled by parameters such as the batch and mini-batch size, the number of local epochs etc. Furthermore, we propose an algorithm that devices can apply to their local data to carefully curate the batches used for local updates, so as to effectively protect their location privacy without hurting utility. Finally, we show that the effect of multiple users participating in federated learning depends on the similarity of their trajectories. To the best of our knowledge, this is the first study of DLG attacks in the setting of federated learning from crowdsourced spatio-temporal data.

1.3 Thesis Outline

The structure of this thesis is as follows. Chapter 2 discusses related work and background. Chapter 3 presents a measurement and user study to assess mobile data privacy in context. Chapter 4 presents the first application of federated learning to mobile data: FedPacket, a federated learning approach for mobile data privacy, which is evaluated in various scenarios. It also demonstrates and discusses privacy attacks on the feature space. Chapter 5 presents the second application of federated learning on mobile data: signal strength prediction and the corresponding privacy attacks. Chapter 6 concludes the thesis.

Chapter 2

Related Work & Background

2.1 Mobile Tracking & Packet Classification

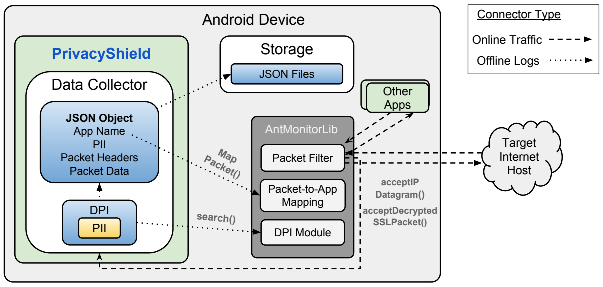
There is increasing interest in understanding and controlling Personally Identifiable Information (PII) exposure and user tracking on mobile devices. Proposed approaches include: permissions [101], static analysis [89], dynamic analysis [57], and network-based approaches [109, 114, 131]. Chapters 3 and 4 fall within the latter which inspect mobile traffic for PII exposure, or other information *i.e.,* tracking, malware, advertising. State-of-the-art tools for network traffic interception and inspection include Haystack/Lumen [111], AntMonitor [122] which use string matching to detect PII in outgoing packets sent from apps to remote destinations. The interception of mobile traffic is not part of our contribution, although mobile data collected using AntMonitor [122, 121] and its extension to signal maps [23] are used for Chapter 3 and 4 respectively.

2.1.1 Mobile Packet Collection via AntMonitor/AntShield

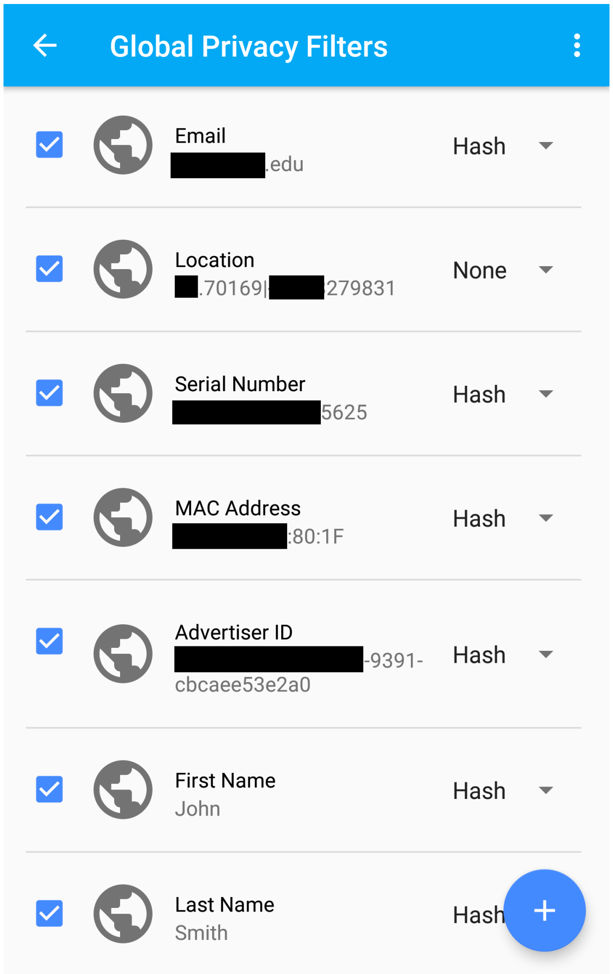
We are interested in understanding real-world cases of PII exposure in packet traces of outgoing packets as captured in AntShield dataset [121, 120] via AntMonitor [122]. We focus on PII that have been previously defined in related work that includes the following list:

- Device Identifiers: IMEI, IMSI, Android ID, phone number, serial number, ICCID, MAC Address, available through Android APIs.
- User identifiers: usernames and passwords used to login to various apps (unavailable through Android APIs); Advertiser ID, email (available through Android).
- User demographic: first and last name, gender, zipcode, city, *etc.* - unavailable through Android APIs.
- Location: latitude and longitude coordinates, available through Android APIs.

The AntShield [121, 120] dataset collected real cases of when the PII defined above are exposed, via AntMonitor, an app that intercepts all network traffic from the device without requiring rooting. AntMonitor is a representative VPN-based tool for privacy protection and it is easy to extend it for data collection. The traffic interception, along with several utility functions of AntMonitor have been made available as a library, and we will refer to it as the AntMonitor Library [122]. We will refer to the data collection app that extends the AntMonitor Library, as AntShield [121, 120]. The design and performance evaluation of the AntShield system is a contribution on its own. However, we consider it out of the scope of this dissertation. AntShield was used as a tool to collect the packet traces in the AntShield dataset which is the starting point of our measurement and user studies in Chapter 3. As shown in Fig. 2.1a, AntShield receives outgoing packets via the PacketFilter interface provided by the AntMonitor Library. Note that the AntMonitor Library also implements a TLS proxy, which allows it to decrypt SSL/TLS traffic of applications that do not use certificate pinning (see [122] for



(a) AntShield Architecture. Data collection consists of the following steps: each packet is intercepted by AntMonitor Library, searched for any PII, and mapped to an app.



Global Privacy Filters

☑ Email
[redacted].edu
Hash

☑ Location
[redacted].70169|[redacted]279831
None

☑ Serial Number
[redacted]5625
Hash

☑ MAC Address
[redacted]:80:1F
Hash

☑ Advertiser ID
[redacted]-9391-cbcaee53e2a0
Hash

☑ First Name
John
Hash

☑ Last Name
Smith
Hash

(b) PII, including those manually entered (e.g., First/Last Name).

Figure 2.1: AntShield System used for Data Collection: Architecture and Screenshot. AntShield was developed by A. Shuba in [121] and used for collecting the dataset that is the basis to the user study here (Chapter 3) and in [30]. It is repeated here for completeness.

details). These decrypted packets are also passed to AntShield via the PacketFilter interface. Each intercepted packet (unencrypted or decrypted) is searched for PII using the AntMonitor Library's Deep Packet Inspection (DPI) module. Some of the PII defined above is available to all apps via Android APIs and is thus easy to find in packets. To find PII that is unavailable through APIs, we add it to the list of strings to search for using AntShield's GUI – see Fig. 2.1b. Note that this methodology may miss PII that is obfuscated by applications prior to transmission, but as shown in [49] such behavior is rare. After DPI, the AntMonitor Library's mapPacket API call is used to note which app was responsible for generating the outgoing packet in question.

2.1.2 User Studies on Mobile Data Privacy

Several experimental studies have analyzed mobile app behavior and several users studies have analyzed user interactions and mobile usage (*e.g.,* Mehrotra et al. [95], Tian et al. [127], EarlyBird [136] and Xu et al. [140]). Most closely related to this dissertation, are user studies that focus specifically on privacy. Permissions and how users interact with them have been extensively studied in [74, 75, 135, 28, 87, 88]. Almuhimedi et al. [28] studied how sending users privacy nudges affected their permission settings. Wang et al. [135] studied user decisions when presented with permission settings that are separated between apps and ad libraries. More recently, Ismail et al. [74] showed that it is possible to maintain app usability even when disallowing certain permissions. Chitkara et al. proposed a retrofitted Android system, ProtectMyPrivacy [48], that allows users to make fewer privacy decisions by setting permissions based on third-party libraries instead of applications. Taking a different approach, PrivacyStreams [86] proposes and evaluates (with a user study) a tool for developers to write code in a more privacy-preserving way. In the web ecosystem, the work in [107] studies online privacy in websites to identify mismatched user expectations and the factors that impact these mismatches. The work by Kleek et al. [132] is closest to our work in that they use information captured from network monitoring to see if it influences users' decisions to install apps. Unlike their work, however, we are interested in learning what users would do if given more fine-grained control over their data.

In **Chapter 3**, we analyze packet traces from the AntShield dataset and the privacy exposures found therein. This approach has the advantage that it analyzes actual real-world privacy exposures, as opposed to *e.g.,* potential exposures described by permissions. In addition, we compile the large volume of information from the packet traces and present it to users in a way that they can process and assess. The combination of a measurement study (volume and coverage of packet traces obtained through extensive and systematic experiments) with a user study (summarizing the information into categories, defining context, and obtaining fine-granularity feedback from users) is one of the contributions of this work, in addition to the detailed findings of both studies.

2.1.3 Packet Classification

There are many approaches based on manually curated blacklists [8, 11, 1] of domains on which they decide to block the whole packet destined to such domains or cookies from such domains [63]. Since blacklists are hard to maintain due to the ever-changing advertising ecosystem, additional graph analysis [131], or machine learning (ML) [67, 124] are used. NoMoAds [124] and NoMoATS [123] are state-of-the-art approaches for detecting Advertising and Tracking requests, respectively. They train classifiers, based on URL requests labeled by blacklists [8], to detect advertising and tracking; conversely, the classifiers trained this way can generalize, complement and enhance blacklists' manual curation. Recon [115] and AntShield [121] use ML to detect PII exposure in outgoing HTTP packets: they train (offline, and in a centralized fashion) per-app/domain Decision Trees to detect PII exposures, based on features extracted from HTTP packets. We build on top of these ML approaches to introduce mobile packet classification learning in a distributed way. A step towards a more privacy-preserving PII detection is PrivacyProxy [126], which processes user data locally and sends only hashed data to a server, however it has to wait for enough data to be collected from other users in order to detect PII. All these approaches are Centralized, as they do not consider collaboration between users to leverage diverse app usage behaviors that can generate PII or Ad requests. In this work, we focus on two classification tasks: PII exposure and Ad request detection because of the availability of labeled datasets that support these per-packet classification tasks, but our federated mobile classification approach can be used towards predicting other tasks, *i.e.,* fingerprinting [148], or tracking [123] detection.

In **Chapter 4** our focus is on how to adapt and evaluate the FL framework specifically for mobile packet classification (as opposed to the image/text classification that is most commonly used for). We showed in [120] showed that systematic crowdsourcing where users collaborate with each other via data sharing helped to train better classifiers to detect PII exposures. To leverage crowdsourcing in a more privacy-preserving way, we considered FL for federated packet classification. In contrast to related work in the field that is using image classification or next word prediction via word/character embeddings [93], we focus on a problem where pre-trained word embeddings (*i.e.,* Word2Vec [15]) are not applicable due to non-dictionary words present in HTTP packets. We apply FL in a setting where shallow models' (*i.e.,* SVM) performance is comparable to state-of-the-art methods, this means that deep learning architectures (*e.g.,* Convolutional Neural Networks [93]) are unnecessary. In terms of privacy, we demonstrate a privacy attack by an honest-but-curious server that uses non-zero gradients to recover the features (shallow leakage) of a target user. In Sec. 4.5, we evaluate the sensitivity of the attack to various FL parameters. More specifically, we quantify the success of the attack in terms of percentage of total features recovered in each FL round. Moreover, we show what sensitive information can be inferred about a target user, once the server has reconstructed all their local data based on two different feature spaces (our proposed one and one baseline). To the best of our knowledge, this is the first work to show what information can be inferred in case of "leaked" HTTP packet traces in the context of mobile packet classification, as prior works focused on reconstruction of input images [147, 146, 64, 138] or text sentences [147] based on gradients.

2.2 Federated Learning

The Federated Learning Framework. An early version of distributed learning was proposed in [119], where users trained models locally and shared the Stochastic Gradient Descent (SGD) updates of certain parameters with a server, which then updated the global model. However, [119] had no averaging mechanism and the evaluation was limited. The paper that coined the term

"Federated Learning" (FL) was introduced in [93], in order to train text and image classifiers using training data available on a large number of mobile devices. The idea is that devices train SGD-based classifiers based on their local data and send updates (model parameters) to a trusted server, which aggregates them to update a global model. The main advantage of FL is that the training data does not leave the device and thus, it is more privacy-preserving than a centralized model. A secondary advantage is that exchanging model parameters requires less communication (assuming fast convergence) than exchanging the raw training data, but this communication saving comes at some computational cost imposed on the devices to train models locally. Subsequent papers introduced optimizations in terms of communication efficiency, scalability and convergence [78, 45, 68, 37, 69].

This scheme, also called Federated Averaging (FedAvg), provides more privacy in contrast to fully centralized training where the devices must send their data to a server and more utility in contrast to fully decentralized learning where each device trains its own local model. A similar scheme, Federated SGD (FedSGD), has a significant difference from FedAvg: in the former each device performs a single SGD step on their local data and sends the gradient to the server, while in FedAvg each device performs multiple SGD steps on the local training data before sending a gradient to the server, depending on the minibatch size B and local epochs E parameters. FL has the following parameters: R number of FL rounds, B local minibatch size (B=∞ means the whole dataset is treated as a single batch), E number of local epochs, C fraction of users that participate in an FL round. These parameters control the client computation and communication between the server and clients. An overview of the algorithm is shown in Algorithm 1.

Federated Learning & Privacy. Several security and privacy attacks are known for machine learning systems; *e.g.,* [21, 71, 96, 116, 35] which include membership inference attacks, model inversion/extraction and model poisoning via malicious clients/server. Although FL protects the training data of each device and shares only model parameter updates, these updates may themselves leak information, due to privacy vulnerabilities of SGD [96, 98]. Examples of leakage

Algorithm 1: Federated Averaging Algorithm [93].

```
Given $K$ clients (indexed by $k$); $B$ local minibatch size; $E$ number of local epochs; $R$ number of
    global rounds; $C$ fraction of clients; $n_k$ is the training data size of client $k$; $n$ is the total data size
    from all users and $\eta$ is learning rate.
Server executes:
  Initialize $w_0$
  for each round t = 1,2, ... $R$ do
      $m \leftarrow max(C \cdot K, 1)$
      $S_t \leftarrow$ (random set of $m$ clients)
      for each client $k \in S_t$ in parallel do
          $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
      $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate($k, w$):
  $B_k \leftarrow$ (split of local data into batches of size $B$)
  for each local epoch i from 1 to $E$ do
      for batch $b \in B_k$ do
          $w \leftarrow w - \eta \nabla \ell(w; b)$
  return $w$ to server
```

include membership inference of data points [96], inference of certain property of training data [96] or reconstruction of training data [71, 137, 147, 146, 64, 138]. Reconstruction of local data usually requires either training auxiliary models (GANs) [71, 137] and/or having access to an auxiliary dataset [96], however, recent works called “Deep Leakage from Gradients” or DLG attacks [147, 146, 64, 138] introduced new methods for reconstructing local training data based on gradient updates without the need of auxiliary data. To prevent privacy attacks, additional mechanisms have been proposed on top of FL, most notably, Secure Aggregation based on secure Multi-Party Computation (MPC) [38] or Differential Privacy [53] to offer privacy guarantees [94, 65, 36] or both [128], but tuning these add-ons are application specific and utility is not always maintained.

2.3 Data reconstruction based on gradients

Federated learning raises the privacy bar but it can still leak information about local data based on the exchanged gradients of model parameters. The first work that showed possible reconstruction of training data (and their corresponding labels) based on shared gradients in FL, without the need of additional models (e.g., GANS [71]) or side information, is DLG [147]. However, they considered an attack scenario where the attacker observes a single gradient after the user performs a single SGD step on their local data before sending their model parameters to the server, which is the case of FedSGD. A subsequent paper called iDLG [146] optimized the way the target label is reconstructed and thus, improved the convergence speed of the attack. However, the iDLG method is only suitable for classification tasks, as the reconstruction of the target labels is obtained analytically in that case. A more recent work [64] showed for the first time the performance of the DLG attack with FedAvg and impact of local epochs on the attack and improved its convergence by proposing a cosine based distance instead of Euclidean.

2.4 Datasets

In this thesis, we use several existing mobile datasets for evaluation.

AntShield [120, 121, 4]. The tool AntShield was developed by A. Shuba et al. in [121] and was used to collect the dataset with the same name, we use in this thesis. It consists of HTTP(S) packets labeled to indicate if they contain a PII exposure or not. The data was generated with manual and automated testing in 2017, which we combine to a single dataset and consider the 297 apps out of 400, that generated HTTP/HTTPS traffic. Using AntShield's packet capturing ability, we interacted manually or via Monkey [18] and collected packet traces from 100 and 400 most popular free Android apps respectively, based on rankings in *App Annie* [5]. In addition to being used to generate survey questions in our user study in Chapter 3, the AntShield datasets are also used for training the classifiers in the FedPacket approach in Chapter 4.

	Auto	**Manual**
# of Apps	414	149
# of packets	21887	25189
# of destination domains	597	379
# of exposures detected	4760	3819
# of exposures in encrypted traffic	1513	1526
# of background exposures	2289	639
# of HTTP packets	13694	13648
# of HTTPS packets	6830	8103
# of TCP packets	867	2264
# of exposures in TCP (other ports)	38	7
# of UDP packets	496	1174
# of exposures in UDP	17	12

Table 2.1: Summary of Manual and Auto AntShield datasets collected on the device.

NoMoAds dataset [124, 13]. The tool NoMoAds was developed by A. Shuba et al. in [124] and was used to collect the dataset with the same name, we use in this thesis. It consists of HTTP and unencrypted HTTPS packets, labeled with Ad requests and PII exposures they may contain, from 50 most popular apps in the Google Play Store. It was generated in 2017 via manual testing (interacting with each app for 5 minutes) with test accounts (no human subjects were involved) similarly to the AntShield data. It is similar to the AntShield dataset in terms of PII labeled packets, but it also contains state-of-the-art labels for advertising; whether a packet contains an Ad request or not. This dataset is used in the FedPacket approach in Chapter 4.

In-house datasets with real users. This dataset was collected by M. Gjoka, from 10 real users from our lab, in 2015 who contributed their packet traces for a period of 7 months. IRB review was not required as the proposed activity was deemed as non-human subjects research by the IRB at UC Irvine. The packet traces were collected by running AntMonitor [122] which intercepts outgoing network traffic generated from each mobile app. These users installed AntMonitor on their personal phones for 7 months and continued to use their phone as usual – no restrictions there. To evaluate the FedPacket approach in Chapter 4, we consider the two most popular apps across all users: Chrome (8 users) and Facebook (10 users) and treat them as separate datasets.

UCI Campus LTE Dataset [25]. This dataset was collected by E. Alimpertis by running a modification to AntMonitor. The dataset has been used as the basis of many papers since [24, 22, 33]. It consists of 180,000 measurements of cellular signal strength, as opposed to measurements of app network traffic, across 7 different devices from members in our lab collected during a five month period. IRB review was not required as the proposed activity was deemed as non-human subjects research by the IRB at UC Irvine. The UCI campus dataset, although it is small in terms of geographical area (approx. 3 km^2), it is very dense. This dataset is used for evaluation in Chapter 5, where the regression problem is signal strength (RSRP). In particular, we consider the top three (out of 25) cell tower data with the most measurements, where the pseudo-IDs are used as proxies for user splits under federated learning.

Signal Maps Radiocells Data [16]. In Chapter 5, we also use the large-scale real-world Radiocells dataset publicly available in [16], and used in several papers [43] before us. It contains multiple upload files and each file contains RSRP values, location, timestamp, device information (make, model). It does not contain user ids, but each upload file corresponds to a single device. We focus on data from 2017 and the area of London, UK, which had the most measurements and approx. 3500 upload files and use heuristics in order to obtain large users by concatenating multiple upload files.

Chapter 3

Exposures Exposed: A Measurement and User Study to Assess Mobile Data Privacy in Context

3.1 Overview

Applications and third party libraries routinely transmit user data to remote servers, including application servers but also ad servers and trackers, and users have typically limited visibility and understanding of what part of their personal data is shared, with whom, and for what purpose. With the increased interest in online privacy, there are several bodies of related work. On one hand, a number of systems have been proposed that improve data transparency and protect personally identifiable information (PII). In general, these systems fall into three categories: (i) static analysis and application re-writing [97, 66, 54], (ii) dynamic analysis with a modified or rooted OS [56, 14, 130, 58, 139], and (iii) VPN-based network monitoring [106, 112, 125, 115, 122, 121]. While these tools provide more fine-grained control over sensitive data (as opposed to just permissions), the way users engage with these privacy-preserving systems is less well studied. On the other hand, in the human-computer interaction (HCI) community, researchers have extensively studied how different designs for app permissions affect users' decisions on which apps to install and which permission requests are considered legitimate by users [74, 75, 135, 28, 87].

In this chapter, we are interested in understanding *privacy exposures*, which we define as PII transmitted by a mobile app (or third party library used by the app) on the device, over the network interface, towards a remote server. Our goal is to understand not only the extent and mechanisms of PII exposure, but also its context (*i.e.,* functionality of the app, destination server, encryption used, frequency, etc.) and the risk perceived by mobile users today. For example, location needs to be shared for a navigation app to perform its intended and legitimate function, and should not be of concern to the user. In contrast, if the same navigation app uses a library that shares device ids with a third-party server, this is more likely a privacy *leak*[1] and should be of concern to the user. We are also interested in PII actually exposed in real network traffic, as opposed to potential privacy exposures as captured by permissions. To that end, we make the following two contributions.

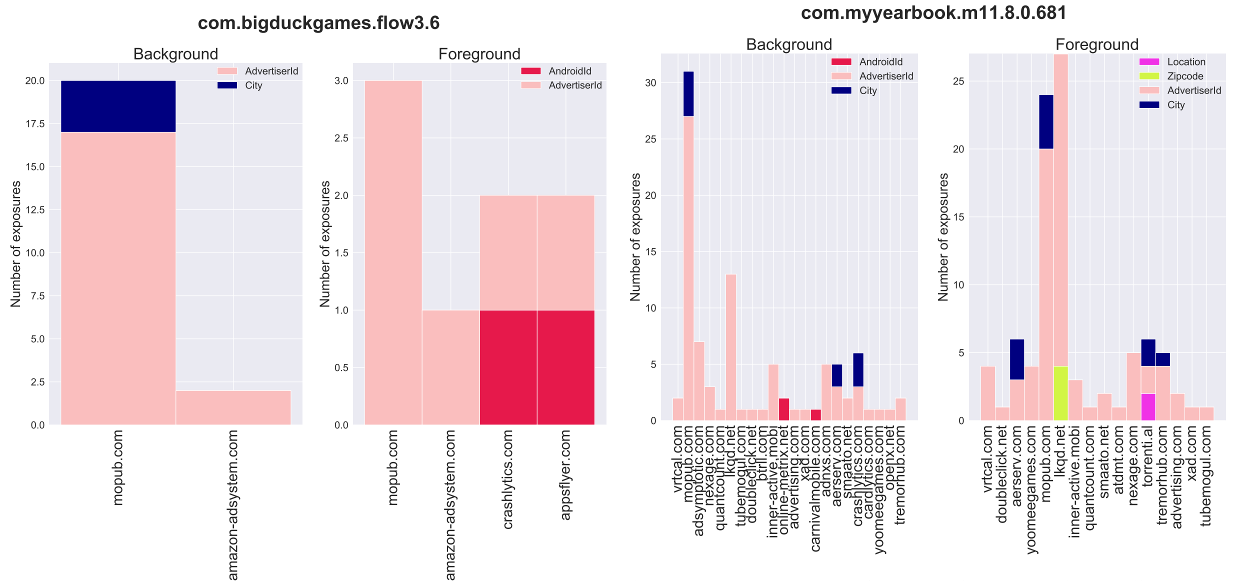
Measurement Study. We use the AntShield datasets [120, 121, 4] which contains 47,076 outgoing packets in total and we identify whether these packets contain any of 16 predefined types of PII (as defined in Section 2.1.1), together with related information, which we collectively refer to as *context*. We define as *context* the following information: the destination server/domain (*i.e.,* whether it is an App Developer server or a third-party Advertisers & Analytics server), the app category (games, shopping, navigation etc.) which reveals the intended functionality of the app, whether the PII is exposed in clear text or is encrypted, and whether the app runs in the background or foreground. The AntShield datasets partly confirm findings of previous measurement studies of mobile devices but are richer: *e.g.,* they contain previously unseen exposures over plain TCP and

[1] Most prior work [115, 110, 121] refers to PII found in outgoing packets as a "privacy leak", because PII is by definition private information and an outgoing packets indicates exfiltration or a "leak". However, we purposely distinguish between "privacy exposure" (a PII contained in an outgoing packet) and "privacy leak" (which is an exposure that is not necessary for the intended functionality of the app, and/or goes to a third party server, or happens in clear text). This distinction, between a PII exposure and an actual leak, can only be made based on the *context*, which is one of the main aspects we investigate in this chapter.

UDP, exposures while the app is in the background, and malicious scanning for rooted devices. We analyze the datasets and provide insights into the extent and nature of how PII is exposed today. We also identify behavioral patterns, such as communities of domains and mobile apps involved in exposing private information. These patterns can be used in the future to design automated prediction and prevention methods. We plan to make the datasets available to the community.

User Study. Second, we perform a user study on Amazon Mechanical Turk (MTurk) with 220 users. We summarize the results of the measurement study in categories, present participants with real-world scenarios of private information exposure in context (type of PII, whether it is shared with the application or a third party, use of encryption, etc.) and we ask them to assess the legitimacy (*i.e.,* whether the information is needed for the app's functionality) and privacy risk posed. We also educate the participants on how a single piece of PII can lead to even more information being discovered when combined with data fetched from a data broker. Finally, we ask users before and after the survey what actions they would be willing to take to protect their privacy, including using free/paid privacy-enhancing tools and contributing their data to crowdsourcing. To the best of our knowledge, our user study is the first to collect and analyze user input in such *fine granularity* (*i.e.,* taking context into account) and on *actual* (not just potential or permitted) privacy exposures from mobile devices. We found that (i) many users did not initially understand the full implications of their PII being exposed but (ii) after being better informed through the study, they became appreciative and interested in better privacy practices. The insights gained by the study can inform the design of fine-grained data transparency and privacy preserving tools such as AntMonitor [122].

The structure of the rest of this chapter is as follows. Section 3.2 presents the data analysis of PII exposures found in the datasets. Section 3.3 presents the Amazon Mechanical Turk study and findings based on our datasets. Section 3.4 concludes the chapter.



(a) *Flow Free*, a puzzle game, exposes the user's city to an advertising server when the application is in the background.

(b) *MeetMe*, an app for meeting people on-line contacts different domains with different PII types depending on whether it is in the background or the foreground.

Figure 3.1: Application behavior exposing PII, while running in the background vs. toreground

3.2 PII Exposures Found in the Datasets

The AntShield datasets provide us with insights into the current state of privacy exposures in the Android ecosystem. Some of the captured patterns were previously unknown, and are revealed for the first time here. For example, we were able to detect exposures happening in the background, exposures in plain TCP and UDP (not belonging to HTTP(S) flows), and malicious scanning for rooted devices.

Background Exposures. AntShield is in a unique position to capture exposures that happen in the background vs. foreground, and other contextual information that is only available on the device. Table 2.1 shows that there is a substantial number of background exposures (*e.g.,* half of all exposures in the automatic dataset) that should be brought to users' attention. Digging deeper, we found several interesting patterns in apps that expose PII both in the background and in the foreground. Fig. 3.1a shows how *Flow Free*, a puzzle game behaves differently in the background

App Name	Leak Types	Port
System	IMEI, IMSI, AndroidId	8080
com.jb.gosms	AndroidId	10086
com.jiubang.go.music	AndroidId	10086
air.com.hypah.io.slither	Username	10086
com.jb.emoji.gokeyboard	AndroidId	10086
com.gau.go.launcherex	AndroidId	10086
com.steam.photoeditor	AndroidId	10086
com.jb.zcamera	AndroidId	10086
com.flashlight.brightestflashlightpro	AndroidId	10086

Domain Name	Leak Types	Port
206.191.155.105	Username	454
206.191.154.41	Username	454
23.236.120.208	AndroidId	10086
3g.cn	IMEI, IMSI, AndroidId	8080
23.236.120.220	AndroidId	10086

Table 3.1: TCP packets (non HTTP/S) sending PII over ports other than 80, 443, 53

vs. the foreground: in the foreground several device identifiers are sent to ad and analytics servers, and in the background, one of the ad servers (mopub.com) also collects the user's city. Perhaps this information is needed to serve personalized ads based on the user's location, but it is unclear why it is needed when the application is in the background and no ads are being shown. Another example is *MeetMe*, an app for meeting people on-line, whose behavior is shown in Fig. 3.1b. In this case, the app collects less PII in the background, but is contacting more ad servers. Such findings are concerning, since apps are causing users data usage and are posing privacy risks even when the user is not interacting with the app.

Non-HTTP Exposures. Prior state-of-the-art datasets [115, 113] reported only HTTP(S) exposures. Table 2.1 reports, for the first time, exposures in non-HTTP(S), including plain TCP or UDP packets. The AntShield dataset contains 29 UDP exposures, all of which were exposing Advertiser ID and Location. As shown in Table 3.1, we also found some apps (mostly games and photo-editing apps) that exposed the Android ID over non-standard (80, 443, 53) TCP ports, such as 8080 or 10086 (a port known to be used by trojans, Syphillis and other threats [17]). The destination IPs could not be resolved by DNS, indicating that the application may have hard-coded

App Name	PII Types	# Exposures
com.ss.android.article.master	City, Adid, Location, AndroidId, IMEI	752
com.cleanmaster.security	Adid, AndroidId	174
com.paypal.android.p2pmobile	City, FirstName, LastName, Zipcode, Adid, SerialNumber, AndroidId, Password, Email	131
com.offerup	Adid, Username, FirstName, Location, Zipcode, AndroidId	114
com.cmcm.live	Adid, AndroidId, Location, IMEI, SerialNumber, IMSI	114
me.lyft.android	City, FirstName, LastName, SerialNumber, Zipcode, PhoneNumber, Location, AndroidId	112
com.pinterest	Adid, AndroidId	111
com.weather.Weather	Adid, Location	110
com.qisiemoji.inputmethod	Adid, IMEI, AndroidId	83
...	...	...
All	All	3039

Domain Name	Leak Types	# Exposures
mopub.com	Adid	2380
isnssdk	AndroidId, IMEI	805
roblox.com	Location	679
applovin.com	Adid	566
rbxcdn.com	Location	561
appsflyer.com	Adid	549
facebook.com	Adid	391
bitmango.com	Adid	371
goforandroid.com	AndroidId	262
ihrhls.com	Adid	219
pocketgems.com	AndroidId	211
ksmobile.net	SerialNumber, Location, AndroidId	159
tapjoy.com	Adid, AndroidId	151
tapjoyads.com	IMEI, AndroidId	147
wish.com	Adid	139
paypal.com	AndroidId	131
...	...	...
All	All	3039

Table 3.2: Summary of applications and domain names with HTTPS exposures in our dataset (manual and auto).

those IPs.

HTTPS Exposures. Since the usage of encryption is increasing, we also collected and analyzed PII sent over HTTPS. Table 3.2 summarizes the exposures we discovered in HTTPS traffic. The top app com.ss.android.article.master is a news app, thus it makes sense for it to query the user's city, perhaps to fetch localized news. However, it is unclear why the app needs the user's IMEI (when it already has the Advertiser ID) and the specific longitude and latitude coordinates of the user. In this case, the city is needed by the app, but the IMEI and location coordinates are potentially privacy exposures. Another example is com.cmcm.live - it exposes 5 different device identifiers for no apparent reason. Hence, although well-behaving apps should use HTTPS, they should also be inspected for potential privacy exposures as not all information that they gather is necessary for their functionality. We also found that the majority of top domains receiving PII over HTTPS were

App Name	Domain	PII Types
com.bitstrips.imoji 10.2.32, 10.3.76	pushwoosh.com	AndroidId
com.nianticlabs.pokemongo 0.57.4	upsight-api.com	Location, AndroidId
com.psafe.msuite 3.11.6 , 3.11.8	upsight-api.com	AndroidId
com.yelp.android 9.5.1	bugsnag.com	AndroidId
com.zeptolab.ctr.ads 2.8.0	onesignal.com	AndroidId
com.namcobandaigames.pacmantournaments 6.3.0	namcowireless.com	AndroidId
com.huuuge.casino.slots 2.3.185	upsight-api.com	AndroidId
com.cmplay.dancingline 1.1.1	pushwoosh.com	AndroidId

Table 3.3: Applications with “jailbroken” field

ad-related. Although it is expected for ad domains to receive the Advertiser ID, other PII should not be collected. These findings motivated us to conduct this user study to crowdsource answers to the question of when a privacy exposure becomes a privacy leak.

Checking for Rooted Devices. We noticed a suspicious flag called “jailbroken” or “device.jailbroken” exposed by several apps (*e.g.,* com.bitstrips.imoji, com.yelp.android, com.zeptolab.ctr.ads, etc). This flag was found in the URI content or in the body of a POST method in the packets, and it was set to 1 if the device was rooted, or to 0 otherwise. In Table 3.3, we show the applications that contain this field in our dataset and the domain to which the “jailbroken” flag is being sent. We also show other types of exposures that the particular domain collects. From the table, we see that the flag is usually accompanied with a device identifier. Several apps send this flag to the same domain (upsight-api.com, an ad network), which indicates that an ad library is probably exposing this information, rather than the app itself.

Behavioral Analysis of PII Exposures. An interesting direction for analyzing the AntShield dataset is via behavioral analysis. For instance, we can ask: (i) what can the communication between mobile apps and destination domains reveal about tracking and advertising? (ii) what type of information exposes to what domains and how to define similarity of apps or domains with respect to exposures? Fig. 3.2 showcases one graph that visualizes similar destination domains with respect to exposures they received, as captured in the AntShield dataset. We define two domains to be similar if they are contacted by the same set of applications (see the box on the right inside Fig. 3.2). For example, domains A and B are similar because they are contacted by two apps

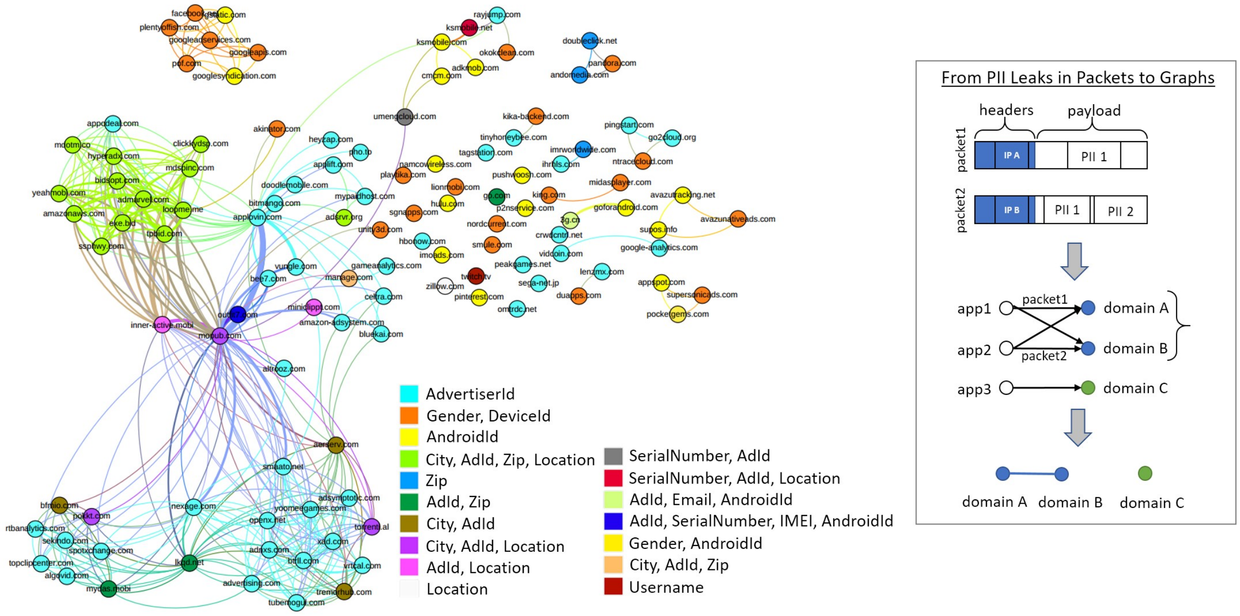


Figure 3.2: Understanding the behavior of app that expose PII through graph analysis of the AntShield dataset. The graph consists of nodes corresponding to destination domains and edges representing the similarity of two domains. Two domains are similar if there are common apps that send packets with PII exposures to both domains; the more common apps expose PII to these domains, the more similar they are, the larger the width of an edge between them. The color of a domain node indicates the types of PII it receives. One can observe from the graph structure that domains form communities that capture interesting patterns: (1) The large communities on the left and bottom consist mostly of ad networks; ad exchanges are nodes in between ad communities; (2) Facebook/Google domains are a different community on their own, on the top left; (3) small apps contact only their own domain, leading to isolate domain nodes; (4) domains in the same community receive the same type of PII (as indicated by the color of nodes).

(app1, app2). We depict the similarity of domains A and B as an edge on the graph of domains, at the bottom of the box. This data can be readily extracted from our trace, together with the type of information that was transmitted from apps to domains.

The graph depicted on the left side of Fig. 3.2 shows a projection of the underlying bipartite graph (middle step in the box) on domains (last step in the box); the graph is plotted and analyzed using Gephi [34]. Nodes in this graph represent domains; the edges indicate similar nodes as per above definition; the width of the edge indicates the number of common applications; and the domain color corresponds to the type of exposed PII. The clusters of domains in the graph are the output of a community detection algorithm, which is a heuristic that tries to optimize modularity.[2]

Interesting patterns are revealed in Fig. 3.2. First, advertising is the result of coordinated behavior. For example, it is easy to identify ad exchanges: mopub.com is in the center of all communication; and inner-active.mobi and nexage.com are also clearly shown as hubs. All three large communities on the bottom and left of the graph correspond to ad networks. Second, on the top left, there is a community of domains that belong mostly to Google and Facebook, and two domains (pof.com and plentyoffish.com) of a dating service. The latter could be because the dating app also sends statistics (*e.g.,* for advertising purposes) to Google and Facebook, in addition to its own servers, as suggested by the type of PII being sent (gender and device ID, represented by the yellow color). Third, not all domains belong to a community: some are well-behaved and are contacted only by their own app. For instance the white-colored domain zillow.com towards the bottom center of the graph is an isolate node and only receives information about the user's location, which makes sense since it provides a real-estate service. Another example is the blue-colored domain hbonow.com: it is only contacted by its own app and only receives the Advertiser ID to serve ads. Another observation from Figure 3.2 is that most domains in the same community receive the same type of PII (as indicated by the domain color). This can be explained by the common ad libraries shared among different apps that fetch the same PII.

In general, similarity of apps and domains based on their PII exposure found in their network activity can be exploited to detect and prevent abusive behavior (*e.g.,* advertising, tracking, or malware) in mobile traffic. This is one promising direction for future work.

[2] The main idea is that for specific node i, it tries to assign different communities of its neighbors like node j's community as i's community and compute the gain of modularity for whole network. The community which maximize the modularity will be the proper one. If the gain of modularity be negative or zero, i keeps its community. This process is an iterative process which is done for all nodes. This algorithm is implemented in Gephi software [34], and works with weighted graphs also.

3.3 User Study: Mobile Data Privacy in Context

In this section, we design a user study on Amazon Mechanical Turk (MTurk) in order to assess user's awareness and understanding of mobile data exposure, as well as their level of concern and potential for adopting solutions. We use the AntShield datasets as analyzed and presented in the measurement study in the previous section in order to present participants with real-world scenarios of private information that was actually exposed by mobile apps in our experiments. We provide the user with information about the types of PII exposed, as well as information about the *context* this exposure occurred, *i.e.,* whether the PII is shared with the application or a third party server; what was the app category/intended functionality; whether it is shared in clear or encrypted text, etc. We then ask the users to assess the legitimacy (*i.e.,* whether the information is needed for the app's functionality) and the risk posed by the particular PII type exposed in that particular context. We also educate the participants on how a single piece of PII can lead to even more information being discovered when combined with data fetched from a data broker. Finally, we ask users before and after the survey whether they would use privacy enhancing tools. To the best of our knowledge, this user study is the first one that collects and analyzes user input in such fine granularity (context) and on actual (not just potential or permitted) privacy exposures at large scale (as found in the packet traces of the measurement study). Section 3.3.1 presents the design rationale and questions asked in the MTurk study. Section 3.3.2 summarizes and analyzes the responses from 220 participants.

3.3.1 User Study Design

MTurk Setup.[3] We designed a Human Intelligence Task (HIT) on Amazon Mechanical Turk (MTurk) [3] and restricted it to workers who are based in the U.S., are at least 18 years old, have

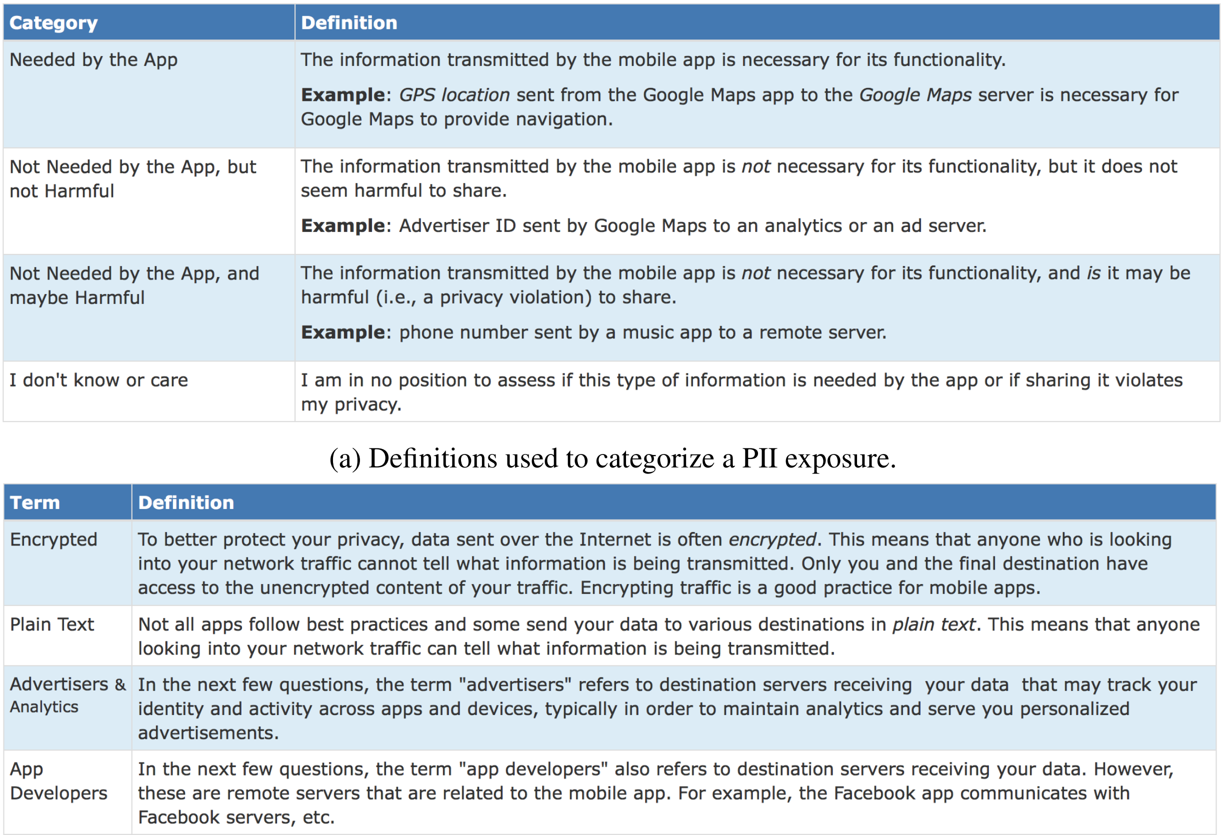
[3] We went through the IRB process in our institution and obtained exempt research registration HS# 2018-4272 "Amazon Mechanical Turk Survey on Mobile Data Privacy".

completed at least High School (in the US), and own a smartphone or a tablet device. The workers were rewarded at a rate of $0.10 per minute of their time – a standard followed in other studies [87, 74, 92]. We allotted 30 minutes for the completion of our HIT, but the majority of workers completed it within 13 minutes approximately. The participants had to pass at least one of three attention check questions in order to have their HIT approved and to receive the $3.00 payment. The HIT was open for 9 days in early May 2018. At the end, we analyzed the responses of 220 workers that passed the attention test.

Demographic questions. First, we asked a set of demographic questions, such as level of education, age, and employment sector (tech vs. non-tech). We also asked what kind of mobile operating system (OS) they use and how many different apps they use daily. In addition to these questions, we added three attention-check questions to prevent workers from gaming the system by providing answers randomly. We discard answers from participants that failed to correctly answer all three attention check questions.

Categorization questions. The main goal of our study is to learn how concerned are users about privacy exposures in different contexts, defined as: the type of PII exposed, the app category, whether the information was shared with a relevant application server (thus may be useful for the functionality of the app) or third party advertiser and analytics servers, and whether it is shared in plain text or is encrypted. To that end, we first defined these terms and categories as shown in Fig. 3.3.

First, we asked the participants how comfortable they are with sharing various **PII types** with different types of **remote servers**, as depicted in Fig. 3.4. Different PII types include: various device ids (such as phone number, IMEI, IMSI, ICCID, Android Id, etc), user ids (*e.g.,* email, Advertiser Id, username and password), location (GPS coordinates), and demographic information (*e.g.,* gender, city, zipcode, first and last name). Destination servers are roughly divided into two categories: app developers vs. ad & analytics servers. The rationale is that the application servers may need the PII to perform their functionality (*e.g., Google Maps* clearly needs location) while



Category	Definition
Needed by the App	The information transmitted by the mobile app is necessary for its functionality. **Example**: *GPS location* sent from the Google Maps app to the *Google Maps* server is necessary for Google Maps to provide navigation.
Not Needed by the App, but not Harmful	The information transmitted by the mobile app is *not* necessary for its functionality, but it does not seem harmful to share. **Example**: Advertiser ID sent by Google Maps to an analytics or an ad server.
Not Needed by the App, and maybe Harmful	The information transmitted by the mobile app is *not* necessary for its functionality, and *is* it may be harmful (i.e., a privacy violation) to share. **Example**: phone number sent by a music app to a remote server.
I don't know or care	I am in no position to assess if this type of information is needed by the app or if sharing it violates my privacy.

(a) Definitions used to categorize a PII exposure.

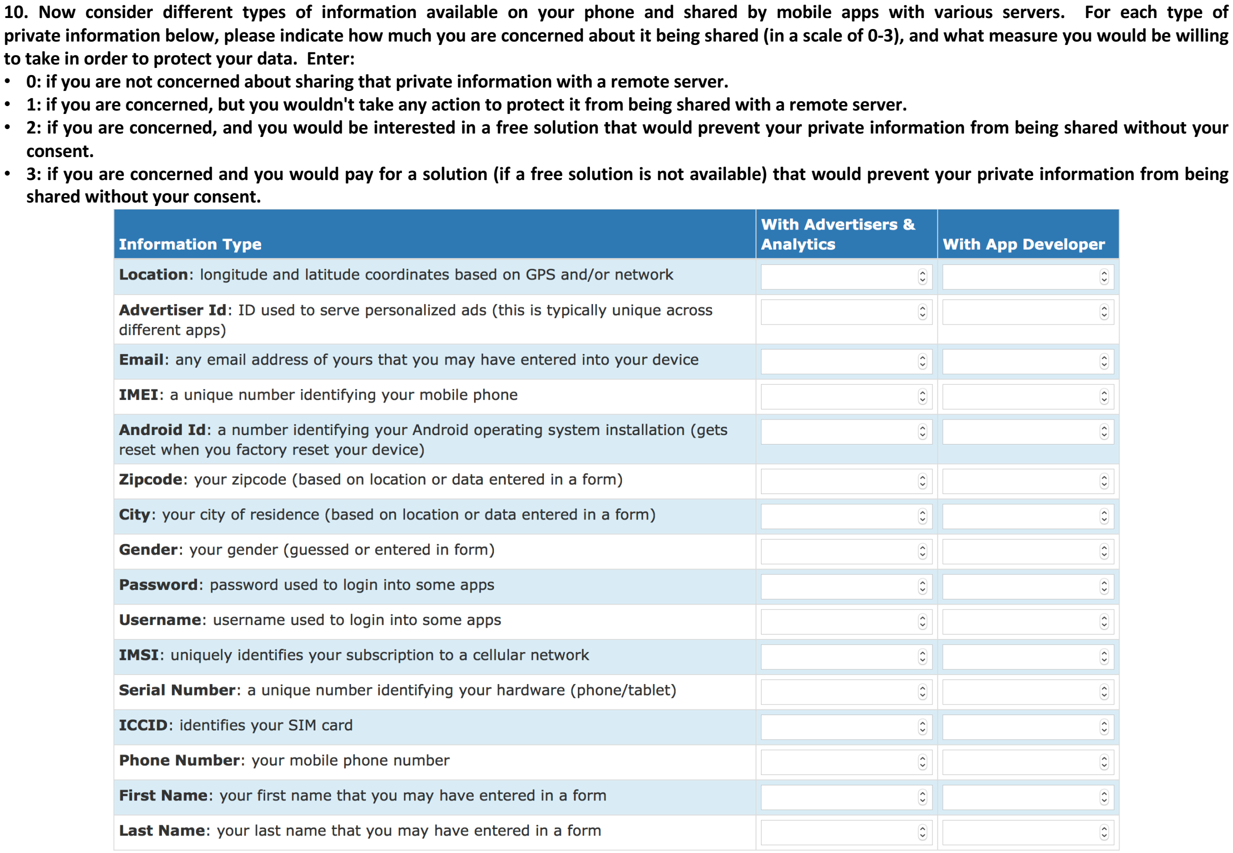
Term	Definition
Encrypted	To better protect your privacy, data sent over the Internet is often *encrypted*. This means that anyone who is looking into your network traffic cannot tell what information is being transmitted. Only you and the final destination have access to the unencrypted content of your traffic. Encrypting traffic is a good practice for mobile apps.
Plain Text	Not all apps follow best practices and some send your data to various destinations in *plain text*. This means that anyone looking into your network traffic can tell what information is being transmitted.
Advertisers & Analytics	In the next few questions, the term "advertisers" refers to destination servers receiving your data that may track your identity and activity across apps and devices, typically in order to maintain analytics and serve you personalized advertisements.
App Developers	In the next few questions, the term "app developers" also refers to destination servers receiving your data. However, these are remote servers that are related to the mobile app. For example, the Facebook app communicates with Facebook servers, etc.

(b) Definitions used to describe additional context of a PII exposure.

Figure 3.3: Terms defined before being used in the categorization tasks.

third party servers do not (thus causing more of a privacy leak rather than an exposure). For each pair of (PII type, destination type) we asked the participants to rate their comfort level of sharing that PII type with that remote server, on a scale from 0 to 3; where 0 represents the least concern and 3 represents maximum concern (and their willingness to pay for a privacy-preserving solution). See Fig. 3.4 for details.

Second, we asked participants to rate their concern in real-case scenarios of PII exposures from the AntShield dataset. For each packet that contained a PII in our dataset, we considered a **broader context** beyond just *PII type* and *destination server type* (*i.e.*, application server or ad/analytics server). Furthermore, we considered the category of the app (*e.g.*, game vs. navigation), whether the PII was *encrypted* or sent in plain text, and the *frequency* of this PII being exposed by this app category. The rationale is that the same PII type exposed may be more or less concerning to users



10. Now consider different types of information available on your phone and shared by mobile apps with various servers. For each type of private information below, please indicate how much you are concerned about it being shared (in a scale of 0-3), and what measure you would be willing to take in order to protect your data. Enter:

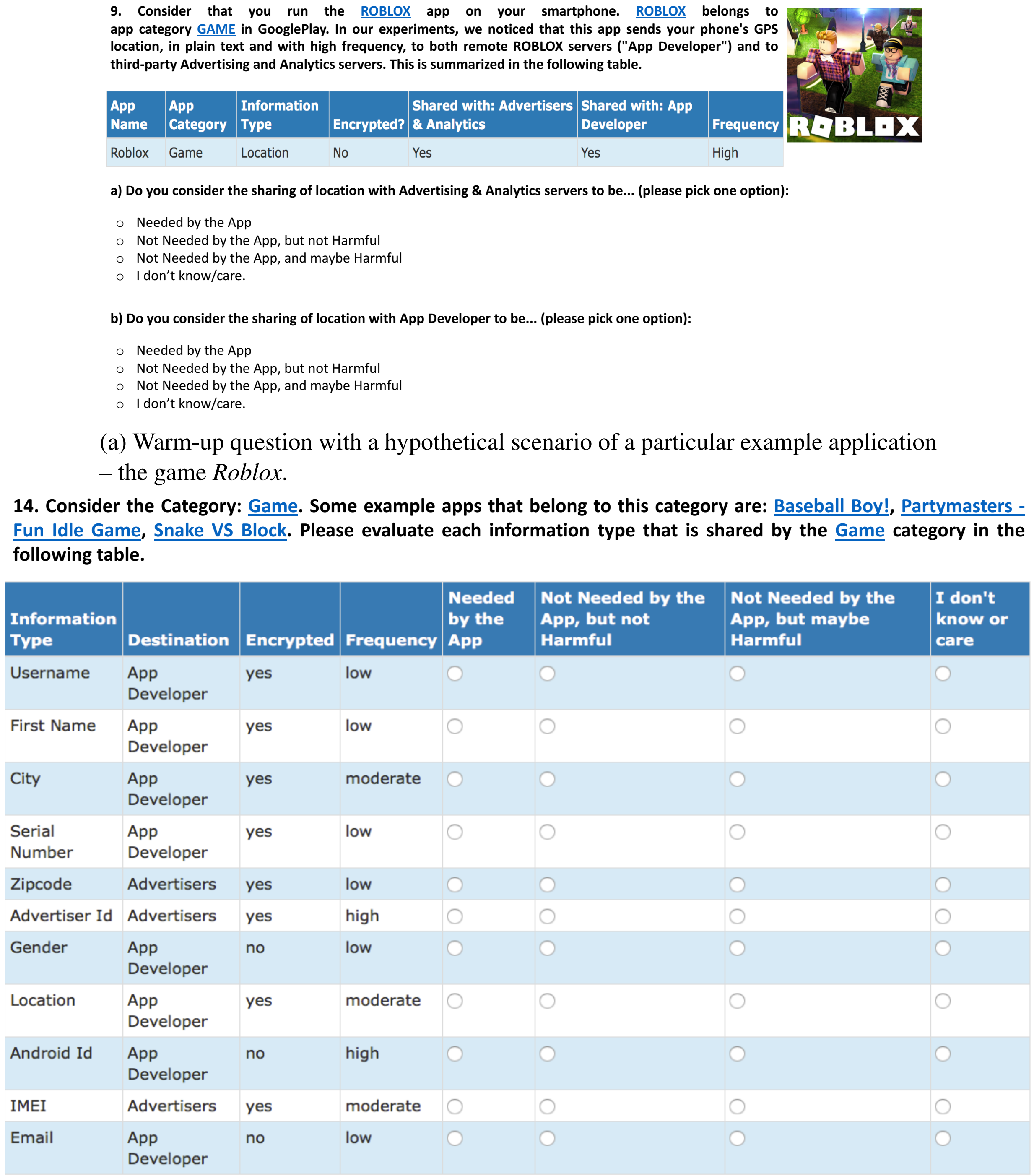
- **0: if you are not concerned about sharing that private information with a remote server.**
- **1: if you are concerned, but you wouldn't take any action to protect it from being shared with a remote server.**
- **2: if you are concerned, and you would be interested in a free solution that would prevent your private information from being shared without your consent.**
- **3: if you are concerned and you would pay for a solution (if a free solution is not available) that would prevent your private information from being shared without your consent.**

Information Type	With Advertisers & Analytics	With App Developer
Location: longitude and latitude coordinates based on GPS and/or network		
Advertiser Id: ID used to serve personalized ads (this is typically unique across different apps)		
Email: any email address of yours that you may have entered into your device		
IMEI: a unique number identifying your mobile phone		
Android Id: a number identifying your Android operating system installation (gets reset when you factory reset your device)		
Zipcode: your zipcode (based on location or data entered in a form)		
City: your city of residence (based on location or data entered in a form)		
Gender: your gender (guessed or entered in form)		
Password: password used to login into some apps		
Username: username used to login into some apps		
IMSI: uniquely identifies your subscription to a cellular network		
Serial Number: a unique number identifying your hardware (phone/tablet)		
ICCID: identifies your SIM card		
Phone Number: your mobile phone number		
First Name: your first name that you may have entered in a form		
Last Name: your last name that you may have entered in a form		

Figure 3.4: Task to assess how comfortable users are with sharing certain **PII type** with certain type of **remote server**.

depending on the context. For example, location exposed by a navigation app to that app's server in an encrypted packet is probably needed for the app to function, while sending a user id to an unrelated third party (*e.g.*, advertiser) server, frequently and/or in plain text, is indeed a privacy leak.

A side benefit of the aforementioned categorization is that it helped reduce the number of cases to be evaluated by users. Out of 8,579 exposures total (Table 2.1), 1,726 are unique when considering the application responsible, the type of PII, the destination host, and level of encryption. To further reduce the number of cases to label, we grouped the applications based on their *Google Play Store* [10] category and destination type (ad & analytics or not). To find ad & analytics domains, we used the *hpHosts* [11] list as it was found to be the most comprehensive list for the mobile ecosystem



9. Consider that you run the ROBLOX app on your smartphone. ROBLOX belongs to app category GAME in GooglePlay. In our experiments, we noticed that this app sends your phone's GPS location, in plain text and with high frequency, to both remote ROBLOX servers ("App Developer") and to third-party Advertising and Analytics servers. This is summarized in the following table.

App Name	App Category	Information Type	Encrypted?	Shared with: Advertisers & Analytics	Shared with: App Developer	Frequency
Roblox	Game	Location	No	Yes	Yes	High

a) Do you consider the sharing of location with Advertising & Analytics servers to be... (please pick one option):

- ○ Needed by the App
- ○ Not Needed by the App, but not Harmful
- ○ Not Needed by the App, and maybe Harmful
- ○ I don't know/care.

b) Do you consider the sharing of location with App Developer to be... (please pick one option):

- ○ Needed by the App
- ○ Not Needed by the App, but not Harmful
- ○ Not Needed by the App, and maybe Harmful
- ○ I don't know/care.

(a) Warm-up question with a hypothetical scenario of a particular example application – the game *Roblox*.

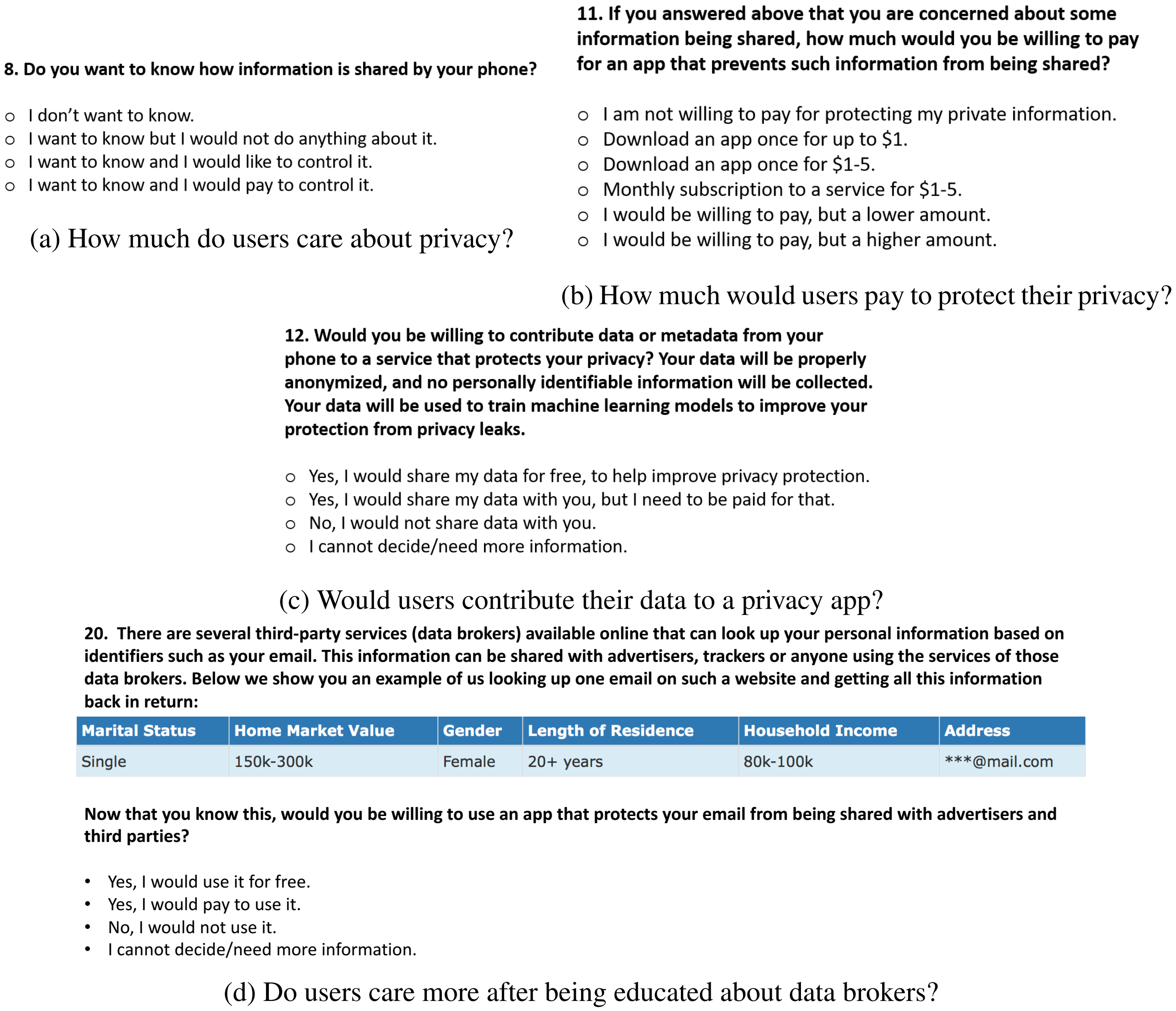
14. Consider the Category: Game. Some example apps that belong to this category are: Baseball Boy!, Partymasters - Fun Idle Game, Snake VS Block. Please evaluate each information type that is shared by the Game category in the following table.

Information Type	Destination	Encrypted	Frequency	Needed by the App	Not Needed by the App, but not Harmful	Not Needed by the App, but maybe Harmful	I don't know or care
Username	App Developer	yes	low	○	○	○	○
First Name	App Developer	yes	low	○	○	○	○
City	App Developer	yes	moderate	○	○	○	○
Serial Number	App Developer	yes	low	○	○	○	○
Zipcode	Advertisers	yes	low	○	○	○	○
Advertiser Id	Advertisers	yes	high	○	○	○	○
Gender	App Developer	no	low	○	○	○	○
Location	App Developer	yes	moderate	○	○	○	○
Android Id	App Developer	no	high	○	○	○	○
IMEI	Advertisers	yes	moderate	○	○	○	○
Email	App Developer	no	low	○	○	○	○

(b) Entire game category with real cases of PII types exposed and their broader context (remote server type but also app category, encrypted/plain text, and frequency).

Figure 3.5: Task to assess user concern about **privacy exposures in context.**

to date [110]. These grouping reduced the total number of unique combinations to 256, which contained 23 unique *Google Play Store* categories (out of 36 total). We split these combinations



8. Do you want to know how information is shared by your phone?

- ○ I don’t want to know.
- ○ I want to know but I would not do anything about it.
- ○ I want to know and I would like to control it.
- ○ I want to know and I would pay to control it.

(a) How much do users care about privacy?

11. If you answered above that you are concerned about some information being shared, how much would you be willing to pay for an app that prevents such information from being shared?

- ○ I am not willing to pay for protecting my private information.
- ○ Download an app once for up to $1.
- ○ Download an app once for $1-5.
- ○ Monthly subscription to a service for $1-5.
- ○ I would be willing to pay, but a lower amount.
- ○ I would be willing to pay, but a higher amount.

(b) How much would users pay to protect their privacy?

12. Would you be willing to contribute data or metadata from your phone to a service that protects your privacy? Your data will be properly anonymized, and no personally identifiable information will be collected. Your data will be used to train machine learning models to improve your protection from privacy leaks.

- ○ Yes, I would share my data for free, to help improve privacy protection.
- ○ Yes, I would share my data with you, but I need to be paid for that.
- ○ No, I would not share data with you.
- ○ I cannot decide/need more information.

(c) Would users contribute their data to a privacy app?

20. There are several third-party services (data brokers) available online that can look up your personal information based on identifiers such as your email. This information can be shared with advertisers, trackers or anyone using the services of those data brokers. Below we show you an example of us looking up one email on such a website and getting all this information back in return:

Marital Status	Home Market Value	Gender	Length of Residence	Household Income	Address
Single	150k-300k	Female	20+ years	80k-100k	***@mail.com

Now that you know this, would you be willing to use an app that protects your email from being shared with advertisers and third parties?

- Yes, I would use it for free.
- Yes, I would pay to use it.
- No, I would not use it.
- I cannot decide/need more information.

(d) Do users care more after being educated about data brokers?

Figure 3.6: Assessing users concern and potential actions. (a) Do users care about privacy? What they are willing to do about it: (c) share their data with a crowdsourcing system (b) use and/or pay for a privacy app). And (d) do they change attitude after being educated?

into five batches of HITs, where each HIT contains five (or three for the last batch) categories of apps to be labeled. To prepare the participants for the labeling task, we first showed a “warm up” question (Fig. 3.5a) with a hypothetical scenario of the *Roblox* app exposing certain PII and asked them to assess the risk (Fig. 3.3b). Next, we asked the participants to label the exposure scenarios for each of the five categories in their HIT – example shown in Fig. 3.5b. We also provide an example app out of each category (from our actual dataset) along with a link to the app’s *Google Play Store* page.

Assessing User Concern and Possible Actions. In order to assess participants' privacy awareness and understanding, we asked another set of questions shown in Fig. 3.6. First, we asked the participants how much they care about information being shared by their mobile device (Fig. 3.6a) and what would they do in order to better protect their information. Next, we asked if they would use an app (*e.g.,* AntMonitor) that can prevent privacy exposures and how much would they pay for such a service (Fig. 3.6b). It was our hope that the categorization questions described previously would educate users about mobile privacy and would make them more concerned towards the end of the survey. To educate them further, we showed them that a single PII in the hands of a data broker can help create or lookup a user profile and can reveal much more information (Fig. 3.6d); the question was based on a real scenario where we fetched data from a data broker based on a person's email address. We then asked them (if they would use a privacy app (and if they would pay for it) to protect their privacy. Finally, we asked them if they would contribute their mobile data to an app that crowdsources information, train machine learning models and prevents privacy leaks (*e.g.,* as in Recon [115] and in [121]). In order to assess our hypothesis that users became more privacy-aware after the categorization questions and the data broker example, the same questions appeared twice during the survey: once in the middle of the survey and once at the end.

3.3.2 User Study Results

In this section, we summarize and analyze the main results of our user study. The main observation is that users seem initially confused about the severity of PII shared in different contexts, but they are interested in and are capable of being trained. Our main findings are as follows:

- Users do not seem to understand how severe it is to share certain PII types (especially device identifiers) with either Advertisers or Developers. This may be because some of these ids, such as Android ID, IMEI, IMSI, ICCID, are difficult to understand or relate to, yet they uniquely identify the device and/or the user and hence should not be shared with remote servers.

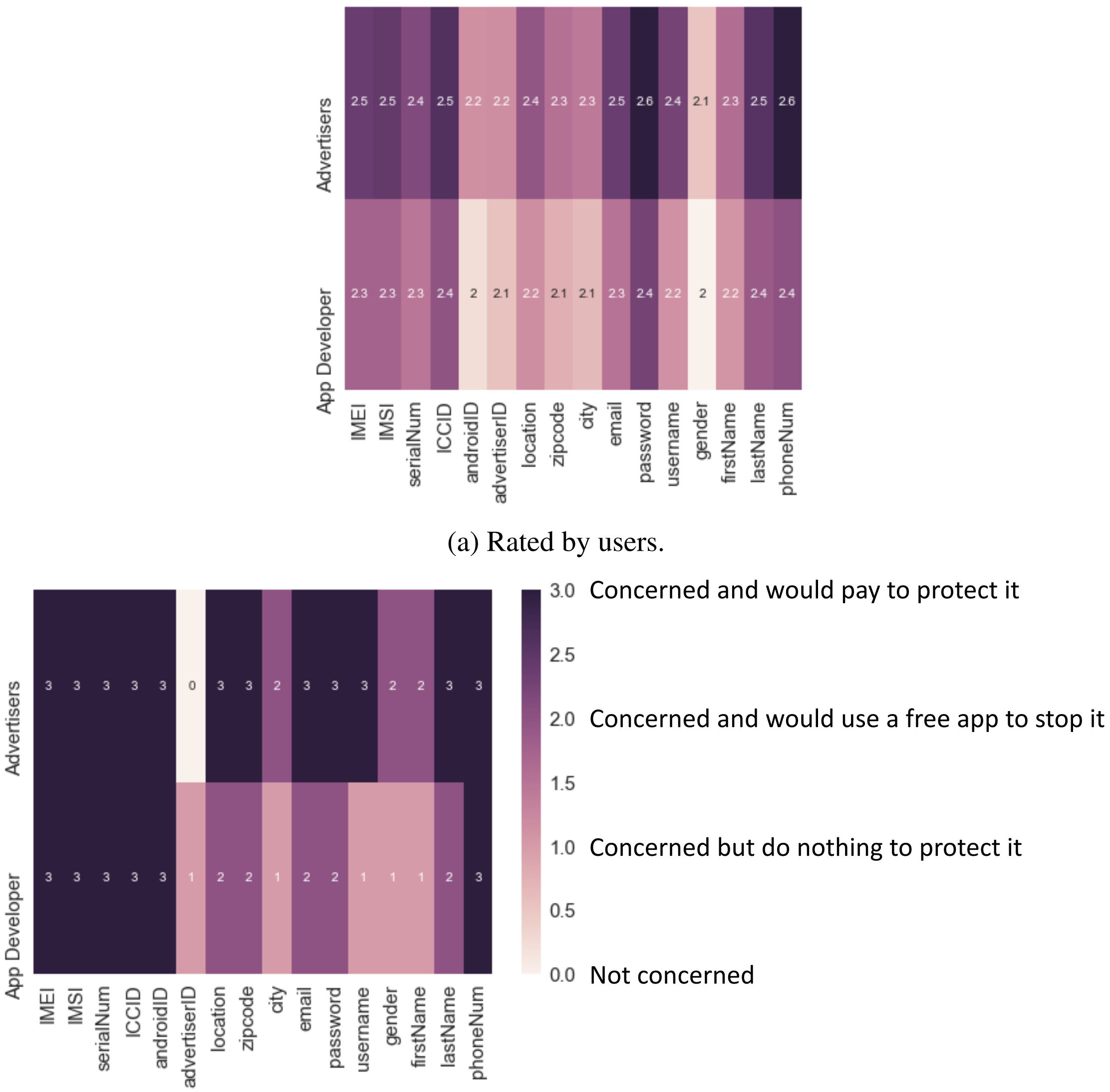
- As expected, users trust Application Developers more than Advertisers & Analytics. However, some comments stated that Developers are “a bunch of hackers behind servers”, indicating possible confusion.

- Users also seem confused about sharing PII in plain text vs. using encryption. Sharing PII in plain text is a bad practice, since it exposes private information not only to the destination servers but also to anyone that is sniffing into the network.

- Towards the end of the study, most users seem to obtain a much better understanding of mobile privacy. For example, several comments state that users are grateful for our short tutorial in mobile privacy. They are willing to educate themselves more in the future and to adopt data transparency and privacy tools. These are encouraging results for future work in developing privacy-enhancing technologies.

Summary of Demographic Info. We collected 223 responses, in total, on our MTurk survey. Since we posted multiple HITs, each with different categorization questions (see Sec. 3.3.1), some users completed more than one HIT, and there were a total of 151 unique participants. Two responses were discarded due to failing the attention-check questions, and one worker was discarded due to incomplete answers. This leaves a total of 220 valid responses, which we analyze in the rest of the section. The majority of the participants were between 25 and 34 years old, held a Bachelor’s degree and were employed in a non-tech sector. 61.8% and 37.7% of the workers were Android users, and iOS users, respectively. 118 of our workers use six to ten different apps every day, 53 use between 11 and 20 apps, 41 use fewer than five apps, and only 8 use more than 20 apps.

Categorization Results: Privacy Exposures in Context. In this subsection, we provide the results we obtained by processing user answers to the categorization questions (see Figures 3.4 and 3.5). We asked participants to rank the severity of PII exposures in different contexts since whether



(a) Rated by users.

(b) Rated by us.

Figure 3.7: How comfortable are users with sharing PII with advertisers and app developers? (a) Average rating of user responses to categorization task of Fig. 3.4 (b) Recommended rating by us ("experts").

or not an exposure is considered a privacy leak depends on the context. There are four main dimensions in each exposure: (i) its destination (app developers or advertisers), (ii) category of the app responsible, (iii) level of encryption used, (iv) PII type. These dimensions play an important role in distinguishing privacy leaks from exposures.

Figure 3.7a demonstrates the results when we asked the users how concerned they are with sharing a certain information type with a particular destination (Advertisers & Analytics or App Developer servers), as shown previously in Fig. 3.4. In this task, we observe that sharing all types of PII is concerning for our participants, regardless of its destination, and they are willing to use a free app to protect them. Furthermore, they would also pay for a tool that protects their phone number and password from leaking to advertisers. As expected, participants are more concerned over their precise longitude and latitude coordinates being shared as opposed to zipcode and city. Overall, our participants seem to trust developers more than advertisers, which is not surprising.

In Fig. 3.7b, we also provide our own "expert" rating for comparison. In particular, we would like to protect device identifiers regardless of their destination, by using a paid solution if a free version is not available. In contrast, our participants chose only to protect their device identifiers with a free solution and they are not willing to pay to protect them. Moreover, they give similar rating to their location data regardless of the destination, although they should be more careful with advertisers than developers, as there are apps that need location in order to function. On the other hand, Advertiser ID should not be protected as well as the other identifiers, since this id is known by advertisers anyway.

Understanding whether PII is needed by the app. We ask users to assess whether sharing a particular PII is legitimate and needed for the functionality of the app (or more broadly by apps in the same category), or if it is unnecessary and potentially harmful. For example, Google Maps is an app in the Navigation category and needs to share location to provide the service.

Fig. 3.8 presents a heatmap of the perceived severity of different PII types per application category. The x-axis show the different PII types and the y-axis contains the categories of apps responsible for sending that PII over the network. The values and colors represent the level of concern the participants have regarding each pair (category, PII type): "not needed by the app and maybe harmful" (value 4 - dark color), "not needed by the app and not harmful" (value 3), "needed by the app" (value 2), "don't care" (value 1 - light color). White color (or 0 value) represents missing combinations of PII and category, that were not present in our datasets. In order to produce the heatmap, we consider the mean values of all participants' answers.

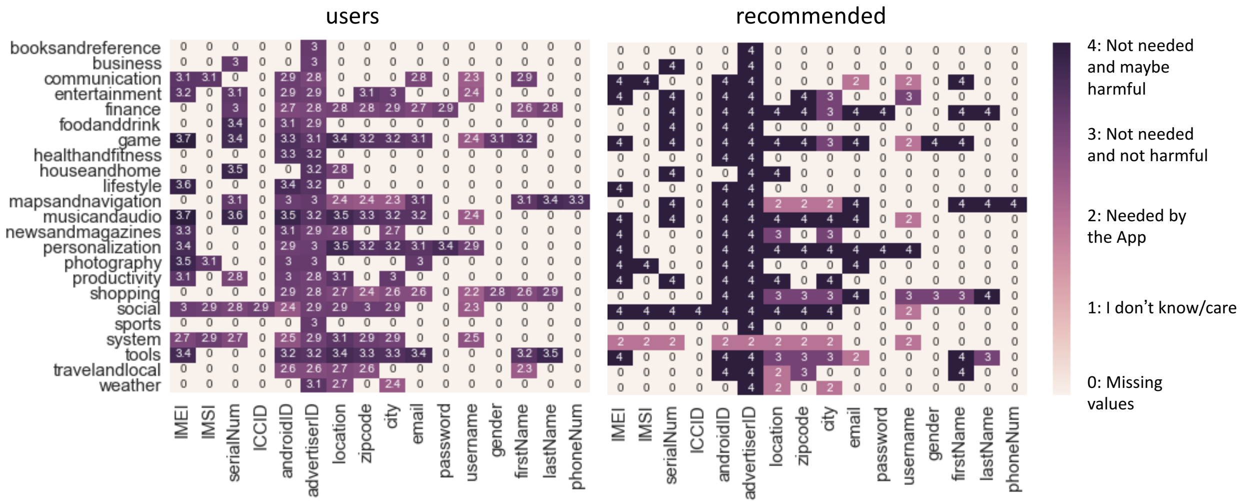
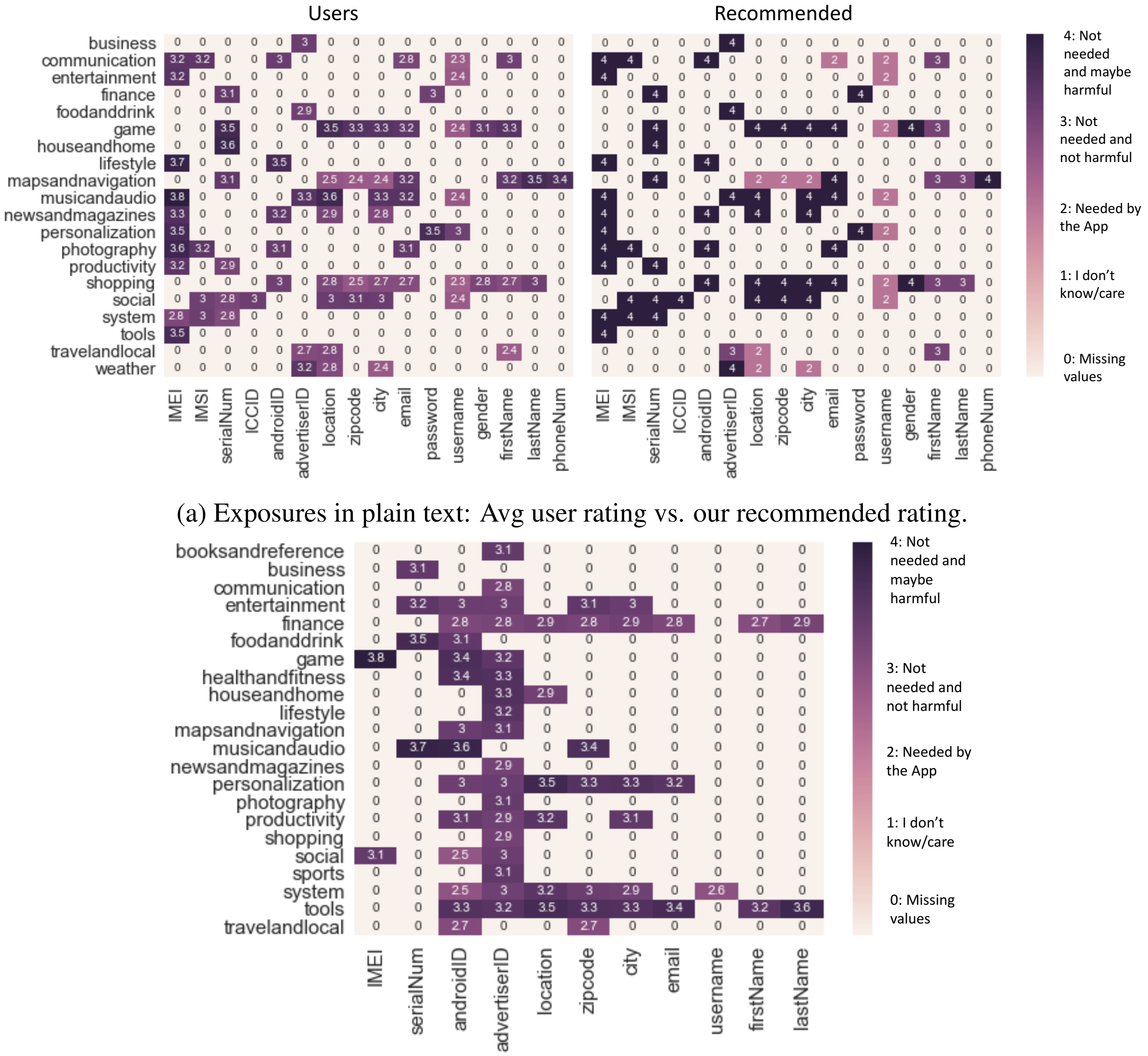


Figure 3.8: Heatmap severity of (PII type, app category). The darkness of the color indicates the perceived severity of the PII exposure: the darkest corresponds to 4.0 ("Not needed by the app and maybe harmful"), while the lightest indicates 1.0 ("I don't care"). Zeros represent missing values for combinations we did not have in our datasets. We compare the average ratings among users who answered that question (on the left column) vs. labeling recommended by us (on the right column).

For comparison, we also show a heatmap labeled by us to express our "expert" opinion. First, location information (latitude and longitude coordinates, zipcode, and city) should be available to system, maps & navigation, weather, and travel & local categories only. All the other categories do not need this kind of information for their functionality and hence when apps transmit such data, it may be considered harmful. Second, user identifiers, such as email, password, username, gender, *etc.* may be needed in some app categories (*e.g.,* social, games, communication), but not in others (*e.g.,* photography, personalization, tools). Finally, device identifiers, such as IMEI, IMSI, Serial Number, ICCID, and Android ID should not be used by any app category, except perhaps for the System. These identifiers are unique and can be used to track users across different apps and build profiles, such as the one we showed in Fig 3.6d. Furthermore, Google explicitly discourages app developers from using these identifiers and asks them to instead create and use their own unique identifiers within their app [6].

Comparing the two heatmaps in Fig. 3.8 reveals that users are not as concerned about device identifiers as they should be: 32.6% of the device identifiers per app category were categorized



(a) Exposures in plain text: Avg user rating vs. our recommended rating.

(b) Exposures in encrypted traffic: average user rating.

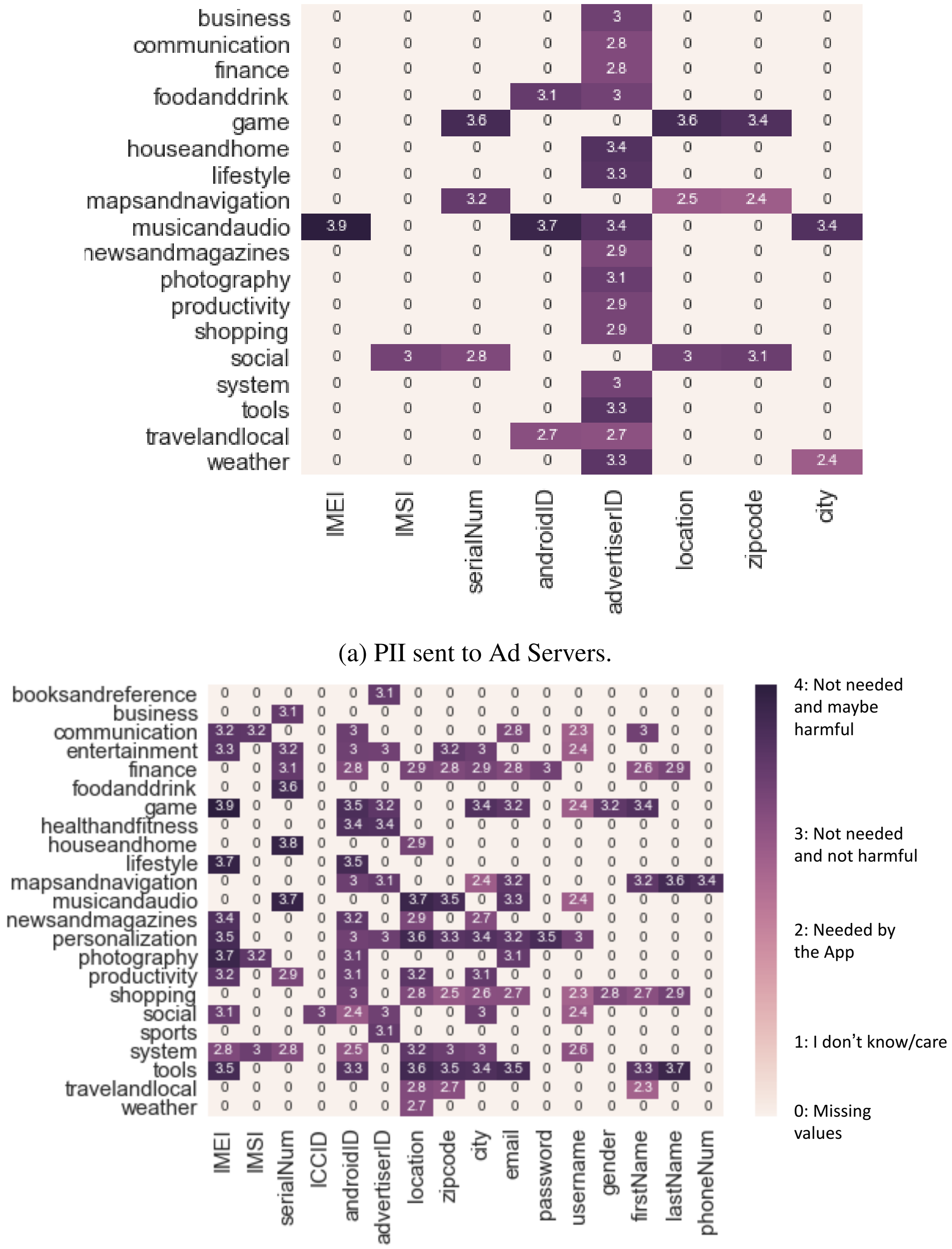
Figure 3.9: Heatmaps assessing the severity of exposures in encrypted vs. unencrypted packets, as assessed by users vs. us ("experts").

by workers as "Not needed by the app and not harmful", although they should only be accessed by the System. However, other responses make sense: app categories that should not require any information to function are trusted the least (games, lifestyle, music & audio, personalization, and tools), and categories such as weather and travel & local are trusted with location information.

Understanding Encryption. We further split the heatmap in Fig. 3.8 into Fig. 3.9a and Fig. 3.9b, depending on whether the unencrytetd packet containing the PII was unencrypted or encrypted, respectively. Our heatmaps not only reveal participants' opinions about exposures, but also showed behavioral patterns of app categories. From Fig. 3.9a, we see that the weather category does not use encryption and sends Advertiser Id, Location and City to remote servers in plain text. Similarly, maps & navigation category sends Serial Number, Location, Zipcode and City information in plain text, but encrypts the Android ID and Advertiser ID. Sending PII in plain text is more harmful since this traffic can be sniffed. Unfortunately, our MTurk participants did not seem to understand the implications of transmitting data in plain text.

Understanding Destinations. We also split our heatmaps based on whether the packet is going to App Developers (Fig. 3.10b) or Advertiser & Analytics Servers (Fig. 3.10a). As expected, developers require more types of information in contrast to advertisers. On the other hand, advertisers should ideally only require the Advertiser ID, which is indeed fetched by almost every app category. However, most of our participants indicated that none of the PII need to be sent to advertisers for the app's functionality (only 2 out of 31 values in Fig. 3.10a are below 2.5). This indicates that perhaps users don't consider ads being served a part of the app's functionality. In contrast, participants indicated more trust towards app developers (Fig. 3.10b), which is expected as certain app categories require PII to function correctly. For example, the following pairs of PII and categories are expected: username for applications with logins (communication, games, and social), email for communication, and location information for travel & local and maps & navigation. However, device identifies, such as IMEI, IMSI, Serial Number and Android Id are not needed by any app category and they should not be shared with advertisers nor developers. Our participants seem to not understand this point and they labeled most of these exposures as "Not needed and not harmful," while in fact these ids are most likely used for tracking. One interesting finding is that participants showed more concern over their IMEI vs. other device identifiers. This indicates that they may need more education about the other device identifiers.

Towards A Solution: Tools Enhancing Data Transparency and Privacy. Fig. 3.11a shows the distribution of answers to the questions of Fig. 3.6, which essentially ask how much users care



(a) PII sent to Ad Servers.

(b) PII sent to Developer Servers.

Figure 3.10: Heatmaps assessing the severity of PII sent to different destination: Third-party (Ad Servers) vs App Developers. Assessed by users (average user rating shown). The darker the color the more concerned the users).

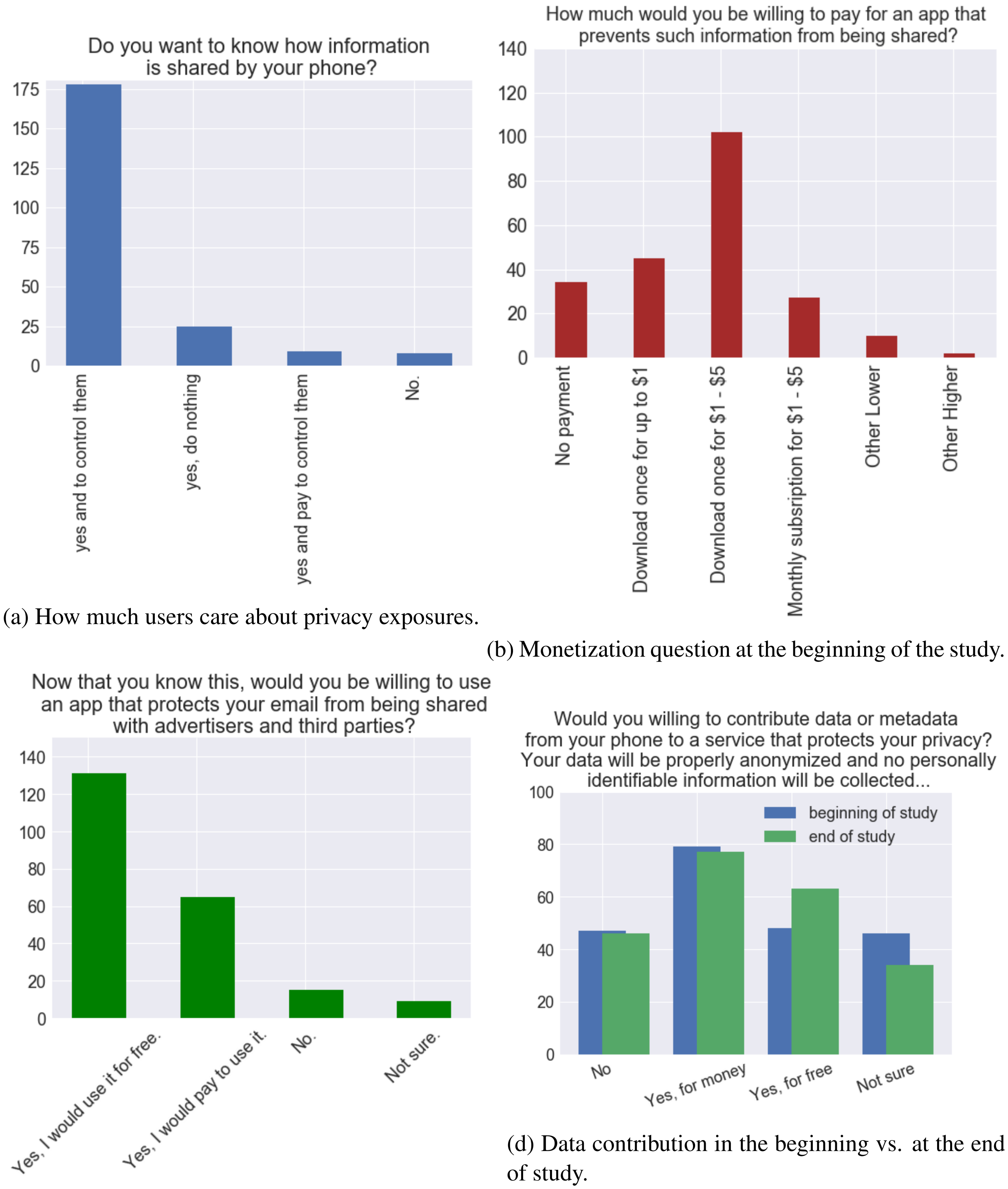
about privacy exposures and what they are willing to do about it. The overwhelming majority of users would like more control over their information being shared, if a free option is available.

Fig. 3.11b demonstrates what payment options users would prefer for a privacy app: the majority would prefer a one-time fee between $1 and $5, than a subscription model. These results show that users are not only interested in using privacy-preserving tools but are also willing to pay for them. Towards the end of the study, after being educated about the extent and severity of sharing PII, more users were willing to pay for a privacy app, *e.g.,* compare Fig. 3.11a to Fig. 3.11c. This demonstrates that although users may originally not be aware or understand the risks of privacy exposures, they become more weary after being educated about the occurrence and risks of sharing PII (especially after learning the power of data brokers). Finally, Fig. 3.11d shows the user's willingness to contribute data to a privacy-preserving tools that crowdsource information (such as Haystack, Recon, AntMonitor) before and after completing our survey. Once again, completing the study made users more open towards using and helping a privacy-preserving app.

User Comments. At the end of the study, we asked participants for their comments and suggestions; 99 out of 220 users provided their feedback. In Fig. 3.12, we summarize the comments of all workers in the form of a wordcloud. Overall, our MTurk workers seemed satisfied with the survey and stated that it was an educational experience. Most of the participants are thankful for our short tutorial on mobile privacy, as they gained more knowledge about privacy and how different apps share their information with various destinations. At the end of the survey, they seem more concerned about privacy and interested in a solution, including a privacy app free or paid; see Fig. 3.11. Several participants mentioned the recent Facebook and Cambridge Analytica scandal and wondered if our user survey was inspired by it (it was not!). Below, we provide a sample of representative user comments, out of 99 total comments, grouped by two recurrent topics.

They learned a lot:

- "I liked this. Was this all due to what is going on with Facebook and other privacy concerns?"
- "I enjoyed taking this survey. I would not share my information with anyone that would use the information in anyway that would be damaging to my life. Right now I am getting phone



(a) How much users care about privacy exposures.

(b) Monetization question at the beginning of the study.

(c) Monetization question at the end of the study.

(d) Data contribution in the beginning vs. at the end of study.

Figure 3.11: Answers to questions of Fig. 3.6: how much do users care about mobile privacy and what they are willing to do to protect it? Some questions are purposely repeated in the beginning and at the end of the study.

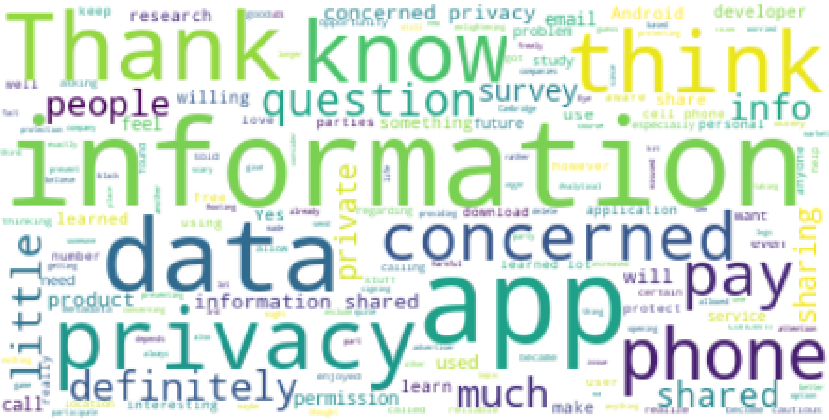


Figure 3.12: Overview of main keywords extracted from 99 MTurk workers' comments

calls I don't want. I would pay to have those stopped."

- "I definitely became more concerned about how much data is taken. You don't realize how much of your personal data is sent to advertisers and it makes you more weary of downloading and using certain apps."

- "I thought it was quite enlightening. I will certainly be paying more attention to what and how an app uses my data in the future."

- "I definitely became more concerned about my privacy. Thank you for the wake up call."

- "Good Study. Made me really think about how much of everybody's information is really out there."

- "I am surprised to learn how much information could be linked to my email address. It is like telemarketing on steroids and I would be willing to keep that info private I think we should always have the option to keep our private info private."

- "I have become a little more concerned about my privacy as some of this was new information to me. I will definitely be doing more research on this subject in the near future. Thank you for the informative study! Everything was very clear and straight forward, I appreciate the opportunity to participate."

- "I definitely learned more about how my private information is shared and will be more cautious in the future."

They are interested in using a privacy-enhancing solution:

- "I did not know all of that and if you developed a product I want it."
- "Yes - I have been concerned, but it has been difficult to know where to start with securing my data. I live in the US, so there is no GDPR to protect me, however I feel I can review application requests for information in a more informed manner. I would also love to find the service you mentioned about protecting privacy of data."
- "Thank you. I learned a lot from this study. I hope you are able to develop something that can help protect consumers and still allow developers to flourish."
- "I would love a reliable app that limited/reduced data collection and increased privacy. Problem is that it's not possible to know which privacy apps are reliable and effective, while still allowing a service or app to be used. Even a reliable, mainstream email alternative would be good!"

3.4 Summary

We provided a combination of a measurement and a user study of actual PII exposed by mobile apps. We also defined and analyzed the context (PII type, destination domain, app category/functionality, background/foreground, use of encryption vs. plain text) where these PII exposures occur, and we distinguished between PII exposure and PII leak (which is more likely to be harmful) depending on the context. In the measurement study, we analyzed a new richer dataset, AntShield, which reveals interesting PII exposures and patterns, some of which were previously unknown. Preliminary graph analysis revealed interesting patterns of apps and domains colluding to expose private information. In the user study, we compiled the large amount of information from the measurement study into a smaller number of categories (contexts) and we asked users to assess the privacy exposures in the actual context they appeared, w.r.t. their legitimacy and risk they pose. Most users were initially unaware of the severity and potential implications of PII exposures: they could not identify the most critical PII or context (*e.g.,* that it may be OK to share information with the application server instead of third parties, such as advertises and trackers). However, they seemed to appreciate the information they got through the study, which made them more willing to adopt privacy enhancing tools. Our analysis combines the scale and coverage of the network-based measurement study with the fine-granularity user input assessing privacy exposures in context [30].

Future Work. There are several directions for future work, building on the observations of this chapter. First, the behavioral analysis of PII leaks at the end of Section 3.2 of the Chapter, revealed similarities in the way apps and destination domains extract PII from mobile devices. Those observations can be further exploited to design machine learning approaches that can detect packets with potential privacy exposures [120, 31], further inspect them and eventually prevent an actual leak (*e.g.,* by using real-time tools like AntMonitor to block a packet or obfuscate a PII). Second, the user study showed that users are interested in and are capable of being educated about their data, and they want to adopt better privacy practices and new tools (such as AntMonitor, Recon or Haystack) to enhance data transparency and privacy control.

Chapter 4

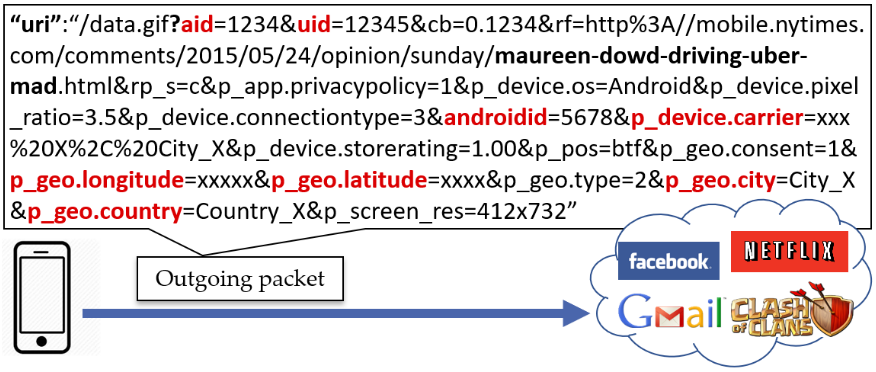
FedPacket: A Federated Learning Approach to Mobile Packet Classification

4.1 Overview

In this chapter,[1] we focus on network-based approaches that inspect packets transmitted out of mobile devices in order to detect PII, tracking, ad requests, malware or other activities; an example is depicted on Fig. 4.1. This information can then be used to take action (*e.g.,* block outgoing packets), to inform the user, or for measurement studies. Early approaches (*e.g.,* Haystack/Lumen [111] and AntMonitor [122]) performed deep packet inspection (DPI) and string matching to find PII in a network packet. Mobile browsers [2] use manually-curated filter-lists [8] to match URI and other information and block ad requests.

Machine learning approaches have been proposed to predict PII [115, 120, 121], ad requests [124] or tracking [123] in outgoing packets, based on features extracted from HTTP requests. These

[1] © 2021 IEEE. Reprinted, with permission, from E. Bakopoulou, B. Tillman, A. Markopoulou, "FedPacket: A Federated Learning Approach to Mobile Packet Classification", IEEE Trasactions on Mobile Computing, 2021. The published version can be found here: `https://ieeexplore.ieee.org/document/9352526`.

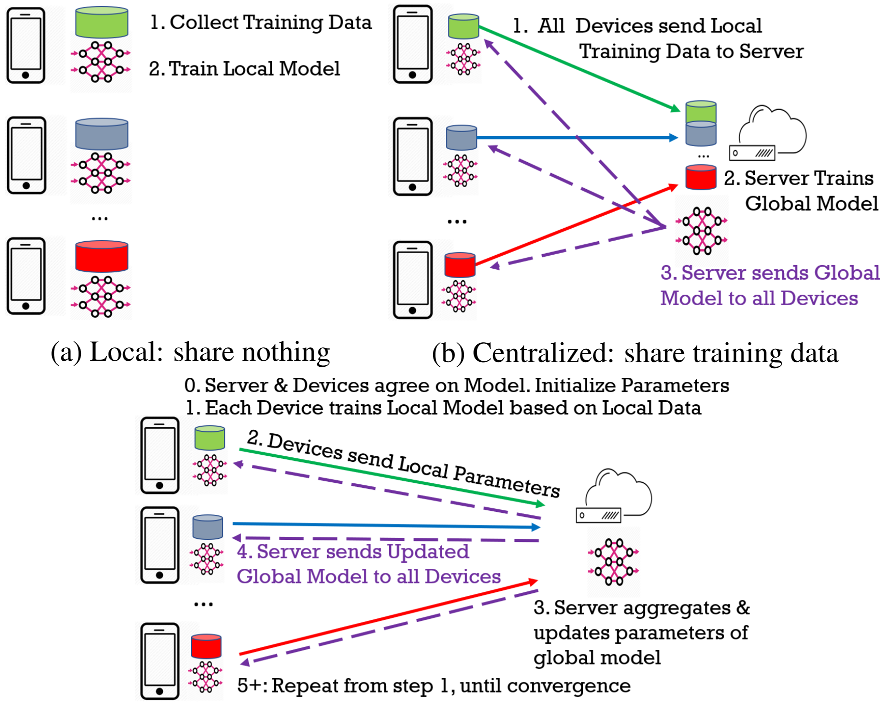


```
"uri":"/data.gif?aid=1234&uid=12345&cb=0.1234&rf=http%3A//mobile.nytimes.
com/comments/2015/05/24/opinion/sunday/maureen-dowd-driving-uber-
mad.html&rp_s=c&p_app.privacypolicy=1&p_device.os=Android&p_device.pixel
_ratio=3.5&p_device.connectiontype=3&androidid=5678&p_device.carrier=xxx
%20X%2C%20City_X&p_device.storerating=1.00&p_pos=btf&p_geo.consent=1&
p_geo.longitude=xxxxx&p_geo.latitude=xxxx&p_geo.type=2&p_geo.city=City_X
&p_geo.country=Country_X&p_screen_res=412x732"
```

Figure 4.1: Example of an outgoing HTTP packet, sent from an app on the mobile device to a remote server. The URI field alone reveals a lot of information, including various identifiers, referred domain, location, *etc.* that can be used for fingerprinting and tracking users.

classifiers are trained using labeled packet traces, obtained either through manual/automatic testing of apps, or by devices labeling packets from real user activity and uploading them to a server. Fig. 4.2 depicts the spectrum of approaches for training such models. At one extreme, we have the the Local approach (Fig. 4.2(a)): mobile users label packets on their device, train and apply their own classifier locally. In this case, users do not share any information with untrusted servers or other users, thus preserving their privacy. However, they do not benefit from the global training data that is available on a large number of devices to generalize. At the other extreme, we have the Centralized approach (Fig. 4.2(b)): mobile users upload their packet logs to a central server, which then trains a global classifier and shares it with all users to apply on their devices. However, network packet traces, collected on users' devices, contain sensitive information (in the label, features, or any part of the HTTP packet) that users may not want to share with a server or other users, because it directly exposes sensitive identifiers (GPS location, device ids) and can be used to infer further sensitive information, such as browsing history.

In this chapter, we propose FedPacket for federated mobile packet classification, which combines the best of both worlds: it allows devices to collectively train a global model, without sharing their raw training data. The Federated Learning (FL) framework was originally proposed to collaboratively train machine learning models, for text and images, from mobile devices without sharing raw training data [93]. An overview is depicted on Fig. 4.2(c). The main idea is that mobile devices


1. Collect Training Data
2. Train Local Model
...
1. All Devices send Local Training Data to Server
...
2. Server Trains Global Model
3. Server sends Global Model to all Devices
0. Server & Devices agree on Model. Initialize Parameters
1. Each Device trains Local Model based on Local Data
2. Devices send Local Parameters
4. Server sends Updated Global Model to all Devices
...
3. Server aggregates & updates parameters of global model
5+: Repeat from step 1, until convergence

(a) Local: share nothing

(b) Centralized: share training data

(c) Federated: share model parameters, not data

Figure 4.2: Overview of general approaches to train machine learning models for packets from mobile devices.

train a local model, and send only model parameters to the server, instead of their raw training data; the server aggregates the information from all users, and sends the updated parameters of the global model to all devices, and the process repeats until convergence. In this chapter, we apply FL to classify outgoing HTTP packets w.r.t. two specific tasks, namely predicting whether an outgoing packet contains: (1) exposure of Personally Identifiable Information, which we refer to as PII; or (2) an Advertising (Ad) request, which typically results in an ad being served in the HTTP response. Methodological challenges we had to address include model and feature selection and tuning the FL parameters, specifically for packet classification.

With respect to *model selection*, we had to bridge the gap between the theory of federated learning (originally developed for image classification using Stochastic Gradient Descent (SGD)-based models, such as Deep Neural Networks (DNNs)) and the practice in network packet classification (previously relying on Decision Trees (DTs) [115, 121, 124, 123], whose rules have intuitive interpretation and can be implemented as regular expressions to easily filter traffic on mobile devices). Unfortunately, DTs do not naturally lend themselves to federation, because they are not SGD-based. In this chapter, we choose *Federated SVM* as the core of our FedPacket framework, as discussed in detail in Section 4.2.3.

With respect to feature selection, we propose a *feature space based on HTTP keys* that performs well for both classification tasks (since PII exposure and Ad requests use the same fields to profile users), while protecting sensitive information and reducing the feature space. First, we observe that not only training data, but also features can expose sensitive information; *e.g.,* that would be the case if some of the PII shown in Fig. 4.1 were selected as features.Therefore, we use only HTTP Keys as features from an HTTP packet, (i) keys from URI and Cookie fields (ii) custom HTTP headers and (iii) the presence of a file request. We purposely do *not* use neither destination domains or hostnames, nor any information from the URI path (which could be sensitive itself if a user visits a sensitive website with *i.e.,* political, medical or religious content) but only the keys mentioned above. Prior work [115, 124] used all the words from the HTTP packets after discarding the most frequent ones and the rarest ones. Our choice of features not only minimizes the sharing of sensitive information, but also reduces the number of parameters that need to be updated. Second, we observe that the size of the feature space depends on the mobile apps and third-party libraries. For example, Webview apps can access any domain, which leads to an explosion of feature space size and wide variation across users; in contrast, non-Webview apps have limited APIs and result in a small feature space, which is the same across different users.

We evaluate our methodology and show that it achieves high F1 score for both classification tasks (PII exposure and Ad Request), with minimal computation and communication. To that end, we use *three real-world datasets:* the publicly available NoMoAds for Ad requests [124, 13] and AntShield for PII exposures [120, 4]; and our in-house datasets collected from real users. For the first two datasets, we create synthetic users by splitting the data evenly or unevenly, and we evaluate how it affects FL. We compare Federated to Centralized and Local approaches w.r.t. the classification performance, communication rounds and computation. We show that the Federated models are superior to Local models and comparable to their corresponding Centralized models, in terms of classification performance (they achieve an F1 score above 0.90 for PII and above 0.84 for Ad request prediction), without significant communication and computation overhead per device. We also demonstrate the benefit of crowdsourcing: a relatively small number of users is sufficient to train a Federated model that generalizes well. Finally, we evaluate different user selection strategies w.r.t. convergence and show that random client selection performs well.

Finally, we address *privacy considerations.* FedPacket clearly improves the privacy of mobile packet classification, by enabling devices to collaboratively train a classifier, without uploading their raw training data to a server. Although it significantly raises the bar in the arm-race, federated packet classification is not immune to inference attacks, which are inherent to any distributed learning approach [96, 98, 147, 64, 138]. In Section 4.5, we demonstrate an attack by an honest-but-curious-server that uses the non-zero gradient updates to infer the features of a target user, and we evaluate its sensitivity to various FL parameters. Furthermore, we show what additional sensitive information can be inferred, such as predicting the websites that a user has visited. To the best of our knowledge, this is the first time that such inference attacks have been demonstrated on HTTP data, and are specific to our problem setup. To mitigate these attacks, we show that Secure Aggregation on top of FL [38], can provide effective protection against feature leakage, essentially by anonymizing the gradient updates.

The rest of this chapter is organized as follows. Section 4.2 presents our methodology for FedPacket. Section 4.3 provides more information about the datasets used for evaluation. Section 4.4 presents various scenarios and provides insights along the way. Section 4.5 presents privacy attacks and mitigation approaches specific to FedPacket. Section 5.6 concludes the chapter.

4.2 Methodology

4.2.1 Problem Setup

Our goal is to train classifiers that use features extracted from HTTP requests coming out of mobile devices, to predict whether those packets contain information of interest, *i.e.,* a PII exposure or an Ad request.[2] In order to train such classifiers, we need training data, *i.e.,* packet traces and labels indicating whether a packet contains the information of interest. We assume that training data are crowdsourced, *i.e.,* obtained and labeled on mobile devices and sent to a server that aggregates them and trains classifiers. We also assume that the devices do not trust the server or other devices but they do want to contribute to the training and use the resulting global classifier. Our goal is to provide a methodology that enables devices to collaborate in training global classifiers, while avoiding to upload training data or even sensitive features to the server.

We apply Federated Learning (FL), for the first time, to the problem of mobile packet classification, which has some unique characteristics and challenges. First, FL has been developed in the context of either text classification (with dictionary words) or image classification. However, the features extracted from packets are non-dictionary words (URI keys are random combination of letters as defined by API developers) and we cannot use well-known pre-trained embeddings on NLP corpus. Second, state-of-the-art on PII/Ad Request classification [115, 121, 124] trained DTs in fully centralized way, and features *e.g.,* words extracted from packets) were fixed a priori. In our case, mobile devices must share their feature space before they start FL, which poses both privacy and scalability challenges. The privacy concerns go beyond the usual ones in FL (*i.e.,* sharing training data) since a value used as feature may reveal sensitive information (thus we propose to use only HTTP keys, not values, as features). W.r.t. scalability, the union of features from all users may explode as more and more users participate and share their feature space. This is especially

[2] Being able to classify packets enables further action *i.e.,* blocking the packet or obfuscating sensitive information, which is out of scope here.

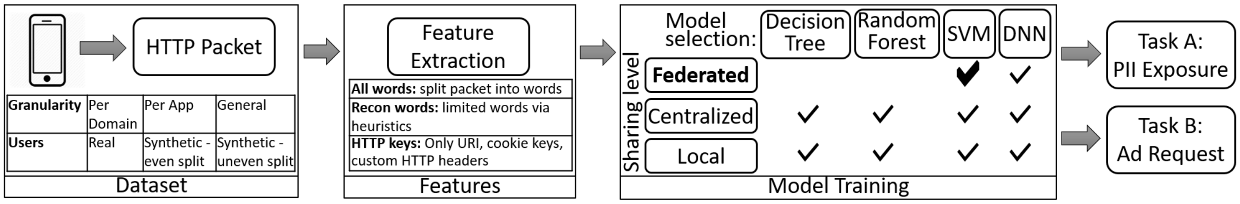


Figure 4.3: Our pipeline for FedPacket. During *training*, the input is a packet trace with HTTP packets sent from mobile apps to remote destinations, labeled for the Tasks (PII Exposure, Ad Request); the output is an ML model, which is trained in a local, centralized or federated way. During *testing*, the input is an HTTP packet, and the output is a binary label (indicating the presence of PII or Ad request).

true for Webview apps, where users have completely different browsing behaviors, thus URI keys. This combined with the lack of well-known embedding for URL words, makes scalability a unique challenge for federated packet classification setup, and one we demonstrate for the first time here.

Federated Learning Approach. To achieve this goal, we apply the FL framework (depicted on Fig. 4.2c) for the first time to the problem of mobile packet classification. This requires addressing several challenges and making design choices and optimizations, specific to our context, such as the following.

Q1. What features can best predict the target label (*i.e.,* PII exposure or Ad request) in a packet? Section 4.2.2 discusses how we select HTTP Keys features from HTTP packets, to achieve high classification performance while also meeting privacy and other constraints.

Q2. What model should we train with FL? Section 4.2.3 compares different models and proposes Federated SVM.

Q3. What datasets should we use to train and test those classifiers? Our training dataset consists of HTTP packets sent by mobile devices, labeled appropriately for each prediction task, *i.e.,* with binary labels to indicate PII exposure or Ad requests in each packet. Collection can be done using an existing VPN-based tool for intercepting traffic on the mobile device [122], and labeling can be done using DPI [115, 121, 120], blacklists [8] or other tools [124]. Section 4.3 provides more information about the three real-world datasets used in this chapter.

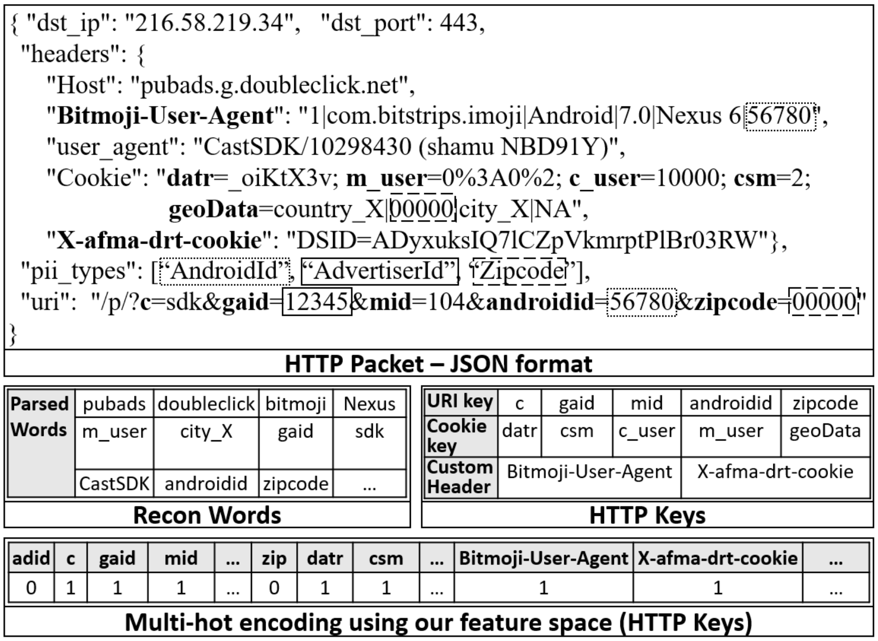
We focus on adapting and evaluating the FL framework specifically for mobile packet classification. An overview of our pipeline is provided on Fig. 4.3 and is described in the rest of this section.

4.2.2 HTTP Features

Feature extraction. We build on the Recon [115] approach which was used by follow-up classifiers [121, 120, 124]: every HTTP packet is split into words based on delimiters; the resulting words include keys and values from all HTTP packet headers. Fig. 4.4 shows an example HTTP packet from Bitmoji (mobile app that creates personal avatars), which sends several identifiers (Android Id, Advertiser Id and zip code) to an ad server. The question is which words to select as features to predict the presence of PII or Ad request in packets, while facilitating FL, preserving privacy and meeting other constrains.

There are several challenges when defining this feature space. First, we need to consider the trade-off between privacy and classification performance. This implies that we may not use some words that can help accurately classify packets, if these features themselves expose sensitive information (*e.g.,* part of URLs and domains can contain sensitive information about user's political views, medical conditions or sexual orientation); to that end we do not use any values as features, only keys. Second, the feature space must have a small size (for high training speed, low memory and computation overhead for updates) and be fixed and known to all participating devices in the FL. Taking these constraints into account, we consider three different feature spaces, two baselines and our proposed one.

Baselines: All Words vs. Recon Words [115, 121, 120, 124]. Instead of considering the union of all words as the feature space (All Words), Recon applied heuristics to remove the words that appear rarely and the most frequent words (stopwords, which correspond to common HTTP headers, common values such as values parsed from the user_agent header). This results in removing



```
{ "dst_ip": "216.58.219.34",   "dst_port": 443,
 "headers": {
    "Host": "pubads.g.doubleclick.net",
    "Bitmoji-User-Agent": "1|com.bitstrips.imoji|Android|7.0|Nexus 6|56780",
    "user_agent": "CastSDK/10298430 (shamu NBD91Y)",
    "Cookie": "datr=_oiKtX3v; m_user=0%3A0%2; c_user=10000; csm=2;
                 geoData=country_X|00000|city_X|NA",
    "X-afma-drt-cookie": "DSID=ADyxuksIQ7lCZpVkmrptPlBr03RW"},
 "pii_types": [“AndroidId”, “AdvertiserId”, “Zipcode”],
 "uri":  "/p/?c=sdk&gaid=12345&mid=104&androidid=56780&zipcode=00000"
}
```

HTTP Packet – JSON format

Parsed Words	pubads	doubleclick	bitmoji	Nexus
	m_user	city_X	gaid	sdk
	CastSDK	androidid	zipcode	...

Recon Words

URI key	c	gaid	mid	androidid	zipcode
Cookie key	datr	csm	c_user	m_user	geoData
Custom Header	Bitmoji-User-Agent			X-afma-drt-cookie	

HTTP Keys

adid	c	gaid	mid	...	zip	datr	csm	...	Bitmoji-User-Agent	X-afma-drt-cookie	...
0	1	1	1	...	0	1	1	...	1	1	...

Multi-hot encoding using our feature space (HTTP Keys)

Figure 4.4: An HTTP packet in JSON, where Android Id, Advertiser Id and zipcode are sent by Bitmoji app to an ad server (doubleclick.net) and thus, it would be labeled positive both for PII exposure and Ad request. Our **HTTP Keys features** are highlighted in bold: these keys are defined by the HTTP protocol and extracted from (1) the URI query keys, (2) the Cookie keys and (3) custom HTTP headers (*i.e.,* “Bitmoji-User-Agent”). Compared to baselines (All Words, Recon Words), HTTP Keys do not use sensitive information *i.e.,* “city_X”.

some but not all values from consideration. In particular, Recon discards the values after the “=” delimiter, however certain values that do not follow this syntax will not be removed from the feature space and those might contain sensitive information. We refer to the remaining features as Recon Words. The URI path also contains potentially sensitive information and words from URI path are also included in Recon Words. Fig. 4.4 shows an example HTTP packet and a subset of the vocabulary from Recon Words, which includes sensitive values such as “city_X”.

Our feature space: HTTP Keys. HTTP keys describe the API calls made by an app or library to destination domains, and they are common across all users of the app/library, as opposed to values that may be specific to the users. For example, many apps use advertising/tracking libraries, and will contain a key-value pair “adid=1234” in the URI header of their outgoing HTTP packets. All users will have the key “adid” in their URI and the presence of such key indicates an ad request. In contrast, the value of each user’s adid is specific to that particular user; if used as a feature (1) it would lead to overfitting and (2) could be used as a sensitive PII. In FL, first, all mobile devices and the server need to agree on the model and features and then they exchange model parameters updates. Both the features themselves and the parameter updates can potentially contain sensitive information. To avoid that privacy risk, we purposely limit our feature space to use only non-sensitive keys from HTTP packets. In particular, we consider the structure of HTTP packets and extract features from: (1) the URI query keys, (2) the Cookie keys, and (3) custom HTTP headers; and (4) whether or not there is a file request in the packet. We refer to this set of features as HTTP Keys.

First, consider the URI: it typically contains a relative path on a given domain and queries, usually built using key-value pairs separated by "&". Sensitive information in URI typically appears in relative paths and query values, while query keys represent API calls to the destination domain. We only use query keys as features. We do not extract any features from the domain or the URI path, as it may contain sensitive information about the user. Second, we include keys from the Cookie field. Query keys from these two fields are sufficient to extract features for most packets in our datasets. Third, to differentiate more packets, we extract custom HTTP headers, which are defined by apps and can embed sensitive information about users. In recent years, apps have started using custom headers to provide app specific functionality. We remove the standard HTTP headers [20] from all HTTP headers to retrieve the custom ones.

Finally, if a packet does not have any keys in the URI, Cookie fields or custom HTTP headers, we include file request– a Boolean feature that indicates the presence of a file request. This case will be mainly a benign activity *i.e.,* requesting static HTML content. Packets that do not contain any of the four features, which we refer to as keyless, are excluded from our pipeline.

Feature Space Size. Selecting HTTP Keys as features already reduces the feature space. However, the feature space size varies widely across apps and users. Various apps use various APIs (which leads to different query keys and thus HTTP Keys) and they may contact different domains. We differentiate between two broad categories of apps based on the number of contacted domains:

apps with or without Webview. Webview apps can contact any domain and present web content in the Webview (*i.e.,* browsers, social media apps like Facebook). Non-Webview apps are more likely to only contact a small fixed set of domains, *e.g.,* back-end servers, analytic and advertisement services. Apps with Webview present new challenges, as the feature space could explode with hundreds of features from every new user, who visits previously unseen domains. We discuss more about Webview apps and their impact on the features in Sec. 4.3.

Vocabularies. Vocabularies are used in ML models with non-numerical features; in our case the vocabulary is the unique words in the dataset. Throughout this chapter, we refer to vocabulary and feature space interchangeably. In this work, we use Multi-hot encoding to represent the extracted words per packet, which is a sparse binary vector with the length of the vocabulary such that it has 1s at the locations of words in the vector; 0 otherwise. An example is shown at the bottom of Fig. 4.4. We use the same feature space for both classification tasks (Ad request and PII exposure), because there is a relation between the two tasks: apps use PII information for serving ads. In FL, the vocabulary must be fixed and shared apriori between all mobile devices and the server. Recon Words potentially expose sensitive information during the construction of such shared feature space. Fixing a vocabulary across various users is successful when the feature space is fixed *i.e.,* non-Webview apps. The intuition is that a single user might not explore the entire API of a service, but across multiple users this is more likely to happen.

4.2.3 Model Selection: Federated SVM

Once the feature space is fixed, our goal is to train a model using FL. The first step is to select the classification model, *e.g.,* Decision Tree (DT), Random Forest (RF), Deep Neural Networks (DNNs), Support Vector Machines (SVM), etc. Next, we train that model in a Federated way (Fig. 4.2(c)) and compare it to its Centralized (Fig. 4.2(b)) and Local versions (Fig. 4.2(a)). The choices we evaluate across these two dimensions (classification model and degree of collaboration among users) are summarized under “Model Training” on Fig. 4.3.

Selecting SVM as the Classification Model. State-of-the-art classifiers for mobile packets have trained DTs [115, 120, 124] to predict PII exposure or Ad requests based on features extracted from outgoing HTTP packets. DTs were chosen primarily due to: (i) their interpretability (nodes in the trees are intuitive rules) for small tree sizes, and (ii) their good classification performance and efficiency for on-device prediction [121, 120, 124] – they can be implemented as regular expressions to filter traffic on the mobile device. Unfortunately, DTs do not naturally lend themselves to federation which has been developed for models based on Stochastic Gradient Descent (SGD), and there is no framework for “aggregating” multiple DTs collected from multiple devices at the server in a federated way. In this work, we choose *Federated SVM* as the core of the FedPacket framework. We show that (i) SVM performs similarly to DTs for our problem, (ii) *Federated SVM* achieves similar F1 score to Centralized SVM, within few communication rounds and with low computation cost per user, and (iii) SVM can be as interpretable as DTs and we also discuss knowledge transfer between the two.

Federated averaging uses models based on SGD, primarily DNNs [93]. In SGD-based models, the mobiles and the server exchange gradient updates, and the server simply averages the local gradients to update the global model. Unfortunately, DNNs require a large number of samples to train, which is costly (in device resources and user experience) to obtain and train on mobile devices. While FL is mostly used to train DNNs, it applies to any SGD-based model. In this work, we select SVMs. Compared to DTs: SVMs are SGD-based (amenable to federation), achieve similar F1 score (due to the simple binary vector representation with multi-hot encoding) and interpretability (via weight coefficients). Compared to DNNs: (1) SVMs use fewer parameters which means less computation, communication and faster training; (2) Linear Kernel SVMs have convex loss functions where more principled guarantees can be provided for convergence; (3) SVMs usually perform better than DNNs on datasets with limited size; (4) SVMs are easier to interpret.

Algorithm 2: Federated SVM

```
Given $K$ clients (indexed by $k$); $B$ local minibatch size; $E$ number of local epochs; $R$ number of
    global rounds; $C$ fraction of clients; $n_k$ is the training data size of client $k$; $n$ is the total data size
    from all users and $\eta$ is learning rate.
Server executes:
  Initialize $w_0$
  for each round t = 1,2, ... $R$ do
      $m \leftarrow max(C \cdot K, 1)$
      $S_t \leftarrow$ (random set of $m$ clients)
      for each client $k \in S_t$ in parallel do
          $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
      $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate($k, w$):
  $B_k \leftarrow$ (split of local data into batches of size $B$)
  for each local epoch i from 1 to $E$ do
      for batch $b \in B_k$ do
          $w \leftarrow w - \frac{\eta}{B} \sum_{i \in B_k} y_i \cdot x_i$, when $y_i(w_i x_i) < 1$
  return $w$ to server
```

Federated SVM. In this work, we use *Federated SVM* with linear kernels. The linear kernel SVM minimizes the following objective function, f, over weight vector w:

$$f(w, X, Y) = \sum_i l(w, x_i, y_i) + \alpha ||w||^2, \tag{4.1}$$

where x_i is the feature vector (*i.e.,* the Multi-hot encoding for a packet), y_i is the binary label, α is the regularization term and the Hinge loss function: $l(w, x, y) = max(0, 1 - y \cdot (w \cdot x))$. Pegasos [117] applied the SGD algorithm for SVM, which we call "SVM SGD". The Hinge loss function is convex and has the necessary sub-gradients, *i.e.,* if $y \cdot w \cdot x < 1$, then $\nabla l(w, x, y) = -y \cdot x$, otherwise 0. This step is easily added to the SGD algorithm, but more importantly to Federated Averaging [93].

Algorithm 1 shows the Federated SVM algorithm: we apply the SVM-based gradient updates to the Federated Averaging [93]. Federated SVM trains an SVM model distributively over K clients (corresponding to mobile devices), where C fraction of the clients update their model in each round and all clients update the global model by averaging their model parameters. A client update consists of multiple local epochs E, and minibatch split of local data into B batches similarly to standard SGD algorithm. Clients compute the SGD updates based on the above Hinge loss.

Client Selection. To select the k clients in a round based on the C fraction of users, we follow the original FL paper [93], where users are chosen at random and uniformly. However, there is ongoing research on how client selection affects convergence [76]. The authors in [85] proposed selecting clients based on probabilities proportionate to their train data size. In this work, we show in Sec. 4.4.4 that the random client selection performs reasonably well in comparison to two other client selection strategies based on train data size. However, in practice the server might not be able to select clients with these ideal conditions, but rather it depends on how many users are connected to Wi-Fi, charging their device and time of the day (*e.g.,* night) [76, 99].

The Federated SGD algorithm is a special case of Federated Averaging for $C = 1$, $E = 1$, $B = \infty$ [93] (*i.e.,* use every client in a round with a single pass on all their local data once). Usually, we look to push more computation to the clients by setting $E > 1$ and B to a small number, and use a small fraction of clients C in each global round. [93] explores the trade-off between these hyper-parameters and shows how to decrease the global rounds R required to reach a target accuracy on the test sets for image classification and next word prediction. The FL framework trains a shared model, hence the feature space has to be fixed and shared across multiple users. Moreover, the feature space size affects parameter updates, and thus communication costs during training.

Federated vs. Centralized and Local models. Once we have fixed the feature space and the underlying model (SVM with SGD and linear kernel), we compare the Federated vs. Centralized and Local models, as shown in the overview depicted in Fig. 4.2.

- Local models are trained on data available on each device, similarly to prior works [115, 121, 120, 124]. Devices share nothing, thus preserving privacy but not prediction power.
- Centralized models: devices upload their training data to a server, where a global model

is trained, similarly to previous works [115, 121, 120, 124]. This approach trains better classifiers but shares potentially sensitive training data.

- Federated models: devices do not share training data with the server, but send model parameter updates to the server, which then aggregates, updates the global model and pushes it to all devices; the process repeats until convergence.

4.3 Datasets Description

				Prior Work	HTTP Keys					
Dataset	**#Apps/Users**	**#Packets**	**#Ads/PII**	**#Features All/Recon Words**	**#URI keys**	**#Cookie keys**	**#Custom Headers**	**#File Requests**	**#Keyless POST Packets**	**#Dest. Domains**
NoMoAds	50/(synth)	15,351	4,866/4,427	12,511/6,743	2,580	216	204	3,342	2,334/2,123	366
AntShield	297/(synth)	41,757	-/8,170	39,304/19,778	3,855	302	609	4,644	8,836/777	674
In-house Chrome	1/8 real	84,716	-/3,424	40,936/22,714	7,573	3,591	47	12,903	13,786/153	1,607
In-house Facebook	1/10 real	33,580	-/1,347	11,921/6,718	4,370	1,160	19	172	12/0	861

Table 4.1: Summary of datasets: total features (our HTTP Keys vs. prior work), total packets, users, visited domains and classification labels.

We use three real-world datasets, summarized on Table 4.1, to evaluate the performance of our federated approach, w.r.t. two packet classification tasks: PII exposures and Ad requests. For the in-house datasets, to run our ML algorithms, we have preprocessed the raw packet traces into JSON, by keeping only HTTP packet-level information. We redacted all user sensitive information with a prefix and the type of PII it contained (*e.g.,* ANT_email) and labeled the packets with exposures if they contained one of these scrubbed PII exposures.

Packet Classification Tasks. In all three datasets, a packet is considered to have a *PII exposure*, if it contains some personally identifiable information (PII), including the following, as defined in prior work: (i) device identifiers, *i.e.,* International Mobile Equipment Identity (IMEI), Device ID, phone number, serial number, Integrated Circuit Card ID (ICCID), MAC Address; (ii) user identifiers *i.e.,* first/last name, Advertiser ID, email, phone number; (iii) Location: latitude and longitude, city, zipcode. Our framework can be used to detect more PII types if the corresponding labeled ground truth is provided. If a packet contains at least one of these PII types, we assign label 1 to the packet, otherwise 0. For the ad prediction task, if a packet contains an *Ad Request* it is labeled as 1, otherwise 0.

Summary of the Datasets. Table 4.1 summarizes the feature space, as relevant to our federated learning framework, including: number of unique features (URI keys, Cookie keys and custom HTTP headers), number of packets, keyless packets and how many of them were POST requests, packets that contain a file request only but no other features, and unique destination domains.We do not include HTTP POST packets in our training or testing, or keyless packets, *i.e.,* packets without any features (query keys in the URI or Cookie field, custom HTTP headers, or file request). There is no standardized syntax for the POST body in order to obtain only the keys without parsing the values too. Thus, for privacy reasons we decided to not parse them at all and to discard such packets from our experiments.

The AntShield dataset contains the most apps and packets with a PII exposure (8,170), while in--house Chrome contains the most packets (84,716) and the highest number of unique domains (1,607). In the NoMoAds dataset, the feature space has 12,511 features with All Words from the HTTP packet (including values) and 6,743 using Recon Words. On the other hand, HTTP Keys uses only 3,001 features (Table 4.1: sum of URI, Cookie keys, custom headers + 1 for file request), which is less than half of the Recon Words. Similarly, in the AntShield dataset the feature space increases from 4,767 with HTTP Keys, to 19,778 Recon Words and to 39,304 with All Words. This explosion of feature space affects the training speed, the size of the trained models and might expose sensitive information of user data (*i.e.,* values to sensitive keys). The benefit of our HTTP-Keys approach is the following: (1) our significantly reduced feature space can describe both prediction tasks (Ads and PII), (2) users share limited sensitive information, without sacrificing classification accuracy and (3) the reduced number of features leads to smaller models and faster training, which is important in mobile environments.

Webview vs. non-Webview apps. Webview is an Android component that can be embedded into

Chrome	**Intersection features**	**Union features**	**#Packets**	**#Domains**
In-house	370	11,212	84,716	1,607
AntShield	-	75	206	15
NoMoAds	-	-	-	-
Facebook	**Intersection features**	**Union features**	**#Packets**	**#Domains**
In-house	14	5,550	33,580	861
AntShield	-	63	110	4
NoMoAds	-	820	392	82

Table 4.2: Two Webview apps and comparison of their feature space in our datasets. We present the intersection/union of features, number of packets and domains across all datasets.

an app to view the web (albeit more light-weight than a browser). We call "Webview-apps" the apps that allow the user to browse the web, which in turns leads to more unpredictable URLs and key-value pairs (HTTP Keys). An example is Chrome which can visit unlimited domains; each domain will have its own set of features (HTTP Keys). As more domains are seen, the feature space explodes. See Table 4.2, where users had only 370 common keys for Chrome but if we consider the union of visited domains from all users, the feature space explodes to 11,212. We observe a similar explosion of the feature space in our in-house Facebook data, which results in only 14 common features out of 5,550 features across 10 users. In contrast, a non-Webview app will, generally, have a fixed feature space that includes keys corresponding to API calls of that app. Overall, the feature space of Webview apps depends on the usage of each app, *e.g.,* the duration (in terms of packets), user behavior (in terms of domains visited). Fig. 4.5 shows the distribution of features and domains for the top 12 apps with most features in our in-house dataset. There is a positive correlation between the number of features and visited domains for each app. This is not surprising since the number of visited domains will increase the total features. Webview apps, *i.e.,* Facebook and Chrome, have the most features, as expected.

In contrast, non-Webview apps have fewer features due to their limited number of contacted domains. In this chapter, we assume that the datasets contain all possible visited domains and the feature set is fixed. To do so, we extract the union of HTTP keys from all users and we assume this

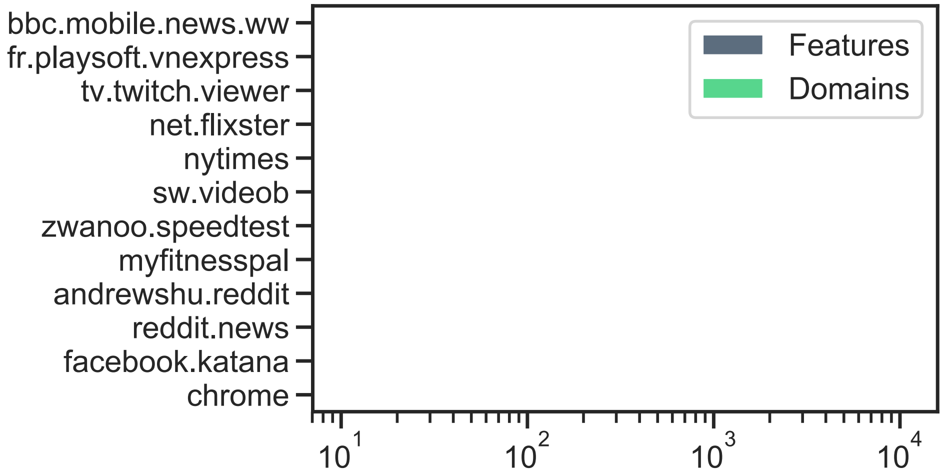


Figure 4.5: Number of features and domains for the top 12 apps with most features from our in-house dataset. The number of features correlates with number of visited domains.

global feature space is known to the server and all users in advance when they are initializing their corresponding models.

Dealing with Heterogeneity. Dealing with system and data heterogeneity across different users, and the related notion of personalization, are open problems in FL [76]. FedProx [84] adds a proximal term to restrict the local model's weights to be closer to the global model. The Continual Learning (CL) literature can tackle convergence in non-IID settings primarily for streaming data, using methods such as: 1) Generative Replay/Memory Rehearsal [102, 26, 27], not applicable to FL due to the need of data sharing; and 2) regularization-based methods [118, 142, 143] adapted to address differences between CL and FL (such as: sequential vs. parallel tasks, one-pass of data vs. multiple passes, etc.). These works aim to achieve better generalization and tackle catastrophic forgetting [77] in non-convex settings mainly by identifying parameters that are most informative for each task and penalizing the changes to these when a new task is being learned. In this work, we chose linear kernel SVM, which has a convex loss function, and we show empirically that it reaches convergence within tens of communication rounds in various settings (IID and non-IID data); Sec. 4.4.4. The CL methods could further speed up convergence, but they would also add computation and communication cost, *i.e.,* tripling the size of the model parameters exchanged [118], or the need of a proxy dataset on the server [142].

Pipeline	**Options**		
Dataset:	NoMoAds	AntShield	In-house Chrome Facebook
Users:	Real users	Even split with k users	Uneven split with k users
Classifier Granularity:	General	Per App	
Models:	Federated SVM	Local SVM	Centralized SVM /baselines
Tasks:	PII exposure	Ad request	Domain

Table 4.3: Parameters of the Evaluation Setup.

4.4 FedPacket Evaluation

General Setup. For each scenario evaluated in this section, we describe the evaluation setup, rationale and results in terms of classification accuracy, communication and computation cost. Table 4.3 lists the possible options for evaluation based on our pipeline defined in Section **??**. We compare the Federated model to Local and Centralized models where the test data comes either from a user or is the union of test data from all users.

We train only general and per-app models, but no per-domain model (it would be impractical to train a model for each domain since there are too many). We split the available data into 80% train and 20% test data and compute F1 score on the positive class (*i.e.,* Ad request or PII exposure is detected). Before training, we balance our dataset via down-sampling the majority class (non-PII, non-Ad) so that it contains an even amount of positive and negative examples to avoid training with a bias towards the most frequent class. We chose down-sampling over oversampling the minority class, as we did not want many users having the same (over-sampled) datapoints from the minority classes which can boost the classification performance. For each of the following experiments we train and test five times each model (unless mentioned otherwise) to obtain an average F1 score.

Creating synthetic users. The NoMoAds and AntShield datasets do not come from real users, since they were produced manually or automatically via Monkey [18] scripts. We create k synthetic users by partitioning the available data via two different approaches: a random split into equal amounts of data (even split) and a random split of data with random sizes of sampled data so that each user contains a different amount of packets (uneven split). The synthetic users have Independent and Identically Distributed (IID) data since they are sampled/created from the same overall distribution. The type of split (even or uneven) controls how homogeneous/balanced the users are in terms of total amount of data. This is different from the real users who are non-IID and unbalanced, as their local data are not sampled from the same distribution and they also differ in terms of data size. We test both methods and show their results, since the advantage of FL is that it can handle various distributions of data across participating users. For both synthetic and real users, we apply the train and test split per user to train Local, Centralized or Federated classifiers. Moreover, we show in Sec. 4.4.4, that training on a subset of users can provide good classifiers for all users.

4.4.1 Scenario 1: Centralized Models

Setup 1. In this experiment, we use the following setup from Table 4.3: *Dataset:* NoMoAds. *Users:* None. *Classifier Granularity:* General. *Models:* Centralized SVM (linear and non-linear kernel, SGD) and baselines (DT, RF). *Tasks:* PII exposure and Ad request. The goal is to validate our choice of Federated model (SVM with SGD) and feature space (HTTP Keys and file request) in the rest of the chapter.

Results 1a: HTTP Keys vs. Recon Words features. In Table 4.4, we compare various Centralized models on 4 different feature spaces: HTTP Keys (3,000 features), HTTP Keys with file request, Recon Words (6,580 features), All Words (12,195 features). HTTP Keys with file request uses a smaller feature space (3,001) but achieves an F1 score above 0.94 and 0.85 for PII and Ads, respectively. Adding the file request feature includes more packets which results in a classification loss of approximately 8% (Ads) and 3% (PII). The drop in performance is slightly larger for Ad

Feature Space (# Features)	HTTP Keys (3000)		HTTP Keys + file request (3001)		Recon Words (6,580)		All Words (12,195)	
Task	Ads	PII	Ads	PII	Ads	PII	Ads	PII
Centralized Classifier	F1 score	F1 score	F1 score	F1 score	F1 score	F1 score	F1 score	F1 score
Decision Tree (DT)	0.936	0.98	0.854	0.95	0.98	0.984	0.979	0.983
Random Forest (RF)	0.938	0.981	0.861	0.949	0.982	0.986	0.979	0.987
SVM with SGD	0.929	0.975	0.838	0.944	0.971	0.981	0.975	0.979
SVM linear kernel	0.933	0.979	0.857	0.952	0.984	0.984	0.981	0.984
SVM rbf kernel	0.706	0.762	0.625	0.744	0.785	0.756	0.761	0.719

Table 4.4: **Results 1a and 1b.** The performance of various ML models on the NoMoAds dataset for the two tasks: Ads and PII prediction. The reported F1 score is averaged, after training and testing each model 5 times. We show that SVM with SGD performs as well as DT and RF. We increase the feature space (packet information used) from left to right. HTTP Keys results in significant reduction in the number of features, while achieving high F1 score for PII (0.94) and for Ads prediction (0.85).

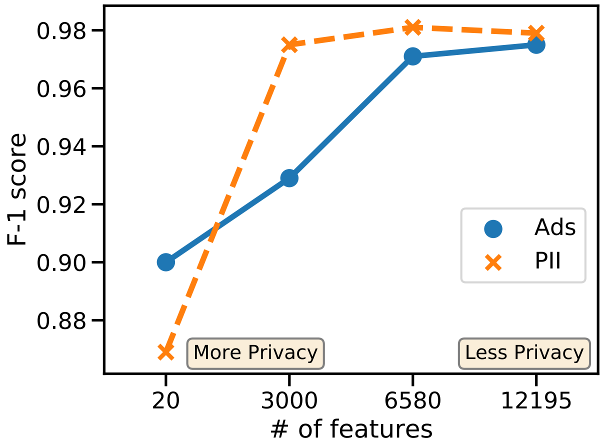


Figure 4.6: Feature explosion and privacy vs. utility for top 20 important HTTP Keys (from Fig. 4.8), HTTP Keys (3,000), Recon Words (6,580), All Words (12,195) depicting Table 4.4.

prediction, since our feature space does not include information from domains that is important for this task as shown in [124]. Prior work [115, 121, 120, 124] uses domain information in addition to other potentially sensitive features, and achieves higher F1 score. There is always a trade-off between privacy and utility, however, our defined feature space and the distributed framework are good steps towards private packet classification, without sacrificing classification performance.

Fig. 4.6 depicts Table 4.4: we reduce features from All words (12,195) to Recon words (6,580) and HTTP keys (3,000), and finally to the 20 most important HTTP features (from Fig. 4.8) and evaluate the F1 score. F1 score decreases but remains above 0.9 for HTTP keys, considering

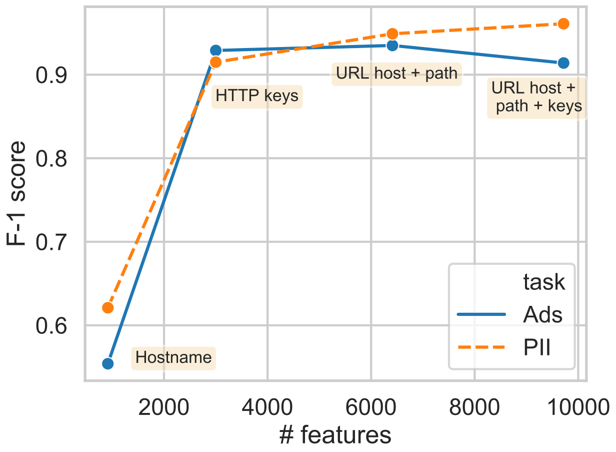


Figure 4.7: Utility vs. privacy and feature explosion when using features from different parts of the URI compared to HTTP Keys feature space.

the 50% reduction from Recon words to HTTP keys, which excludes sensitive information from packets, *i.e.,* values to sensitive identifiers or URI domain and path. This balance of privacy vs. utility trade-off applies to both tasks.

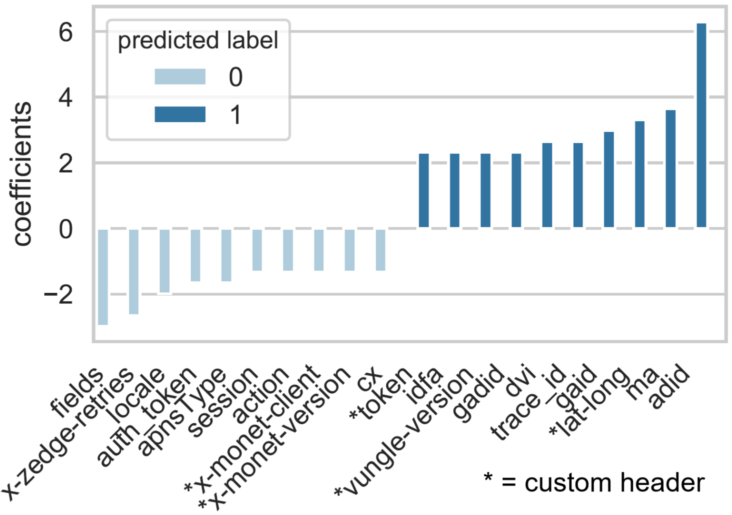


Figure 4.8: Top 10 negative and positive coefficients and the corresponding features obtained from Centralized SVM with HTTP Keys for PII.

Privacy-utility tradeoff. We show in Fig. 4.7 how the F1 score is affected when we use different parts of the URI as features; 1) URI domain only (or hostname), 2) URL host and URI path and 3) full URL including keys but not values. For comparison, we add a datapoint that corresponds to the performance with HTTP Keys. The results show that features extracted only from hostnames

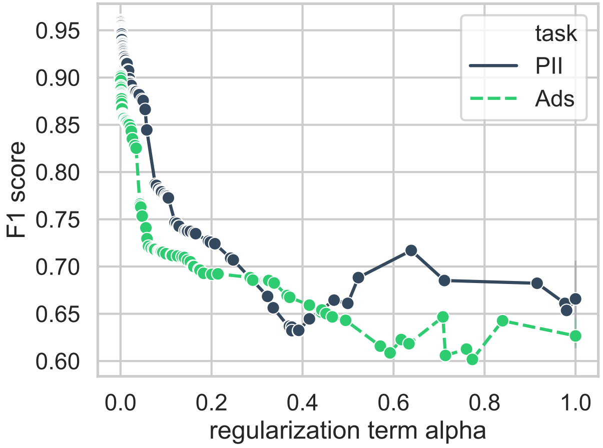


Figure 4.9: Regularization term α and its effect on F1 score for Centralized SVM with SGD for both tasks.

are not sufficient to achieve a high F1 score. Adding more information from the URI results in better F1 score, but is also more vulnerable to privacy attacks that can infer sensitive information about user's train data, such as values to identifiers[3] or user's browsing history; see Section 4.5. In contrast, the HTTP Keys feature space achieves high F1 score while using limited sensitive information from users. In addition to privacy risks, using more features results in models with a large number of parameters, which in turn increases the training time and the model size.

Results 1b: SVM with SGD performs similarly to Decision Trees. Table 4.4 compares SVM with SGD to state-of-the-art baseline models, such as Decision Tree (DT) and Random Forest (RF) (used in prior work [121, 115, 124]) on the NoMoAds dataset. For all feature spaces (i) the linear SVM and SVM with SGD perform similarly to DT and RF; and (ii) SVM with a non-linear kernel (rbf) seems to not generalize well and it is likely to overfit. Thus, we select SVM with SGD as the basis of our FL framework.

Results 1c: SVM Parameter Tuning. After the comparison of different kernels for SVM in Table 4.4, we tuned further the SVM with SGD via Grid Search and Randomized Parameter Optimization. Grid Search exhaustively picks each parameter value within a provided range $[10^{-5}, 1]$

[3] An example of sensitive identifiers is shown in Fig. 4.8 *i.e.,* trace_id, adid (advertiser id), and lat-long. The latter corresponds to precise location and it is embedded in custom HTTP headers indicating even higher privacy risk.

and evaluates its F1 score with 5-fold cross validation on hold-out data. Randomized Parameter Optimization is similar but it samples 30 values randomly with a Log Uniform distribution from $[10^{-4}, 1]$. We used both methods to tune the regularization term α as shown in Fig. 4.9. For the rest of the experiments, we chose $\alpha = 0.0001$ which maximizes the F1 score for both prediction tasks. Regarding the learning rate η, we use the "optimal" η as defined by L. Bottou [41, 40] throughout this chapter unless mentioned otherwise. We discuss how η affects convergence in the federated setting in Sec. 4.4.4 and Fig. 4.15 and show that our chosen η speeds up convergence. We would like to note that after choosing the best model parameters, we apply the same set of parameters to all user models following the original FL work [93], as the question of personalization in FL [84] is still an open problem and out-of-scope in this work.

4.4.2 Scenario 2: NoMoAds for PII, Ad Request

Setup 2a. We use the following setup from Table 4.3. *Dataset:* NoMoAds.*Users:* Even and Uneven split across 5 synthetic users; *Classifier Granularity:* General. *Models:* Federated SVM vs. Centralized SVM. *Tasks:* PII exposure and Ad request. We set local epochs to $E = 5$, batch size to $B = 10$ and we use all users by setting the fraction $C = 1.0$, as we use only 5 synthetic users due to the limited size of the dataset.

Results 2a: Federated vs. Centralized vs. Local. Table 4.5 shows the F1 score, where the Federated model performs as well as the Centralized model and significantly outperforms the Local models. In particular, Federated training performs similarly to its centralized version trained on the union of all users' data. Moreover, the F1 score of the Federated model on each user's test data is slightly higher than their Local models, especially for the uneven split. For uneven split, the average number of rounds (R) required to reach the target F1 score = 0.96 for the Ads prediction is 8.8, while for PII prediction 1 round was sufficient. For even split, to reach the same F1 score, 2.6 rounds were required for Ad prediction and 2.2 rounds for PII prediction.

		Uneven split		**Even split**	
		F1 score		**F1 score**	
Trained on	**Tested on**	**Ads**	**PII**	**Ads**	**PII**
Federated	user 0 test	0.83	0.96	0.84	0.95
Federated	user 1 test	0.92	0.96	0.81	0.95
Federated	user 2 test	0.86	0.95	0.84	0.95
Federated	user 3 test	0.63	0.97	0.88	0.92
Federated	user 4 test	0.85	0.96	0.86	0.96
Federated	all test data	0.84	0.96	0.85	0.95
Local user 0	user 0 test	0.82	0.95	0.78	0.9
Local user 1	user 1 test	0.89	0.94	0.8	0.92
Local user 2	user 2 test	0.8	0.9	0.79	0.93
Local user 3	user 3 test	0.64	0.82	0.83	0.9
Local user 4	user 4 test	0.77	0.87	0.81	0.91
Centralized	all test data	0.85	0.96	0.84	0.94

Table 4.5: **Results 2a.** Federated performs as well as Centralized and outperforms Local models. We show the F1 score for each user, when testing on their hold-out test set and on the union of all users test data.

Setup 2b. This is similar to Setup 2a with 20 synthetic users instead of 5. For B = ∞, we use all available local data as a single minibatch, similarly to [93]. We require all models to reach a target F1 score on test set (0.85 for Ads, 0.95 for PII predictions), chosen to match their Centralized F1 score as shown in Table 4.4.

Results 2b: Impact of Federated parameters. Table 4.6 shows how the average number of rounds (R), until the models reach a target F1 score, depends on the fraction of participating clients (C), a different batch size (B) and local epochs (E). A general trend is that increasing C, decreases R significantly and the gap between min and max decreases. Moreover, increasing B decreases R, as small B leads to faster convergence. These observations apply to both uneven and even splits and to both prediction tasks. In contrast, increasing E and pushing computation to users increases R, except for the case when $C = 1.0$. The reason for this is that our model is simple and more local epochs lead to overfitting. In the context of packet classification, R is much lower than observed in prior work [93] which used more complex models.

Results 2c: Training Time. Fig. 4.10 depicts the average training time (from 5 runs) of the

	—— Uneven split ——		—— Even split ——	
C	**B = ∞**	**B = 10**	**B = ∞**	**B = 10**
Task: Ads with target F1 score = 0.85, E = 1				
0.05	36.6 [24, 58]	22.4 [11, 29]	25 [19, 30]	33.4 [25, 43]
0.1	15 [10, 20]	15.2 [9, 24]	14 [11, 23]	23 [13, 34]
0.2	10 [8, 13]	6.8 [5, 10]	8.6 [7, 14]	11 [6, 17]
0.5	2.6 [2, 4]	3 [2, 4]	4.4 [3, 6]	8 [6, 11]
1.0	1.6 [1, 2]	2.4 [1, 4]	2.6 [2, 4]	4.8 [4, 6]
Task: Ads with target F1 score = 0.85, E = 5				
0.05	43.6 [40, 48]	49 [27, 75]	34.8 [27, 53]	48.8 [43, 63]
0.1	21.2 [13, 28]	20.8 [17, 26]	22.6 [19, 27]	22.4 [18, 27]
0.2	12 [8, 15]	10 [7, 11]	9.2 [8, 12]	10.6 [10, 12]
0.5	3.4 [2, 6]	4.2 [3, 5]	3.8 [3, 5]	5.6 [3, 11]
1.0	1 [1, 1]	1.2 [1, 2]	2.8 [2, 6]	3.4 [2, 7]
Task: PII with target F1 score = 0.95, E = 1				
0.05	30 [19, 37]	28.8 [21, 40]	27.8 [21, 33]	27.8 [23, 31]
0.1	14.2 [9, 18]	15.6 [12, 18]	16.4 [13, 19]	16.8 [16, 18]
0.2	7.4 [4, 9]	7.4 [5, 12]	7.4 [6, 9]	7.2 [6, 10]
0.5	3.6 [3, 5]	3.6 [3, 5]	3.6 [3, 4]	3.4 [3, 4]
1.0	1.8 [1, 2]	2 [2, 2]	2.8 [2, 3]	2.6 [2, 3]
Task: PII with target F1 score = 0.95, E = 5				
0.05	39.6 [32, 44]	48.4 [35, 58]	34.6 [31, 40]	39.2 [31, 44]
0.1	21.6 [16, 37]	20.2 [14, 27]	16.2 [14, 17]	20.2 [18, 22]
0.2	9.4 [7, 14]	10.4 [8, 16]	7.8 [7, 8]	9.2 [7, 11]
0.5	3.2 [2, 5]	3.6 [3, 5]	3 [3, 3]	3.2 [3, 4]]
1.0	1 [1, 1]	1.2 [1, 2]	1.2 [1, 2]	1 [1, 1]

Table 4.6: **Results 2b.** Impact of Federated parameters for NoMoAds data with 20 synthetic users. All models are trained until they reach a target F1 score (selected to match Centralized per task). We vary the parameters C, B, E and we report the rounds R) until the target F1 score is reached: average and [min, max] are reported over 5 runs.

baseline centralized models with NoMoAds data for both prediction tasks in comparison to Federated SVM with 20 even synthetic users.[4] To measure the time for the Federated models, we set the same target F1 score as before (0.95 for PII and 0.85 for Ads) and we report the total training time from all rounds required until the convergence with E=1, B=10, C=1.0. We define the Federated training time as: $t_{fed} = max(t_{local}) + t_{aggr}$ and thus, it depends on the worst case local train

[4] Time was measured on a machine with Intel(R) Xeon(R) CPU E5-2623 v3 @ 3.00GHz and 62GB RAM. The reported train times are on models with default parameters as selected in scikit-learn [103], except for Random Forest (RF) which we limit to 25 estimators instead of the default 100 estimators.

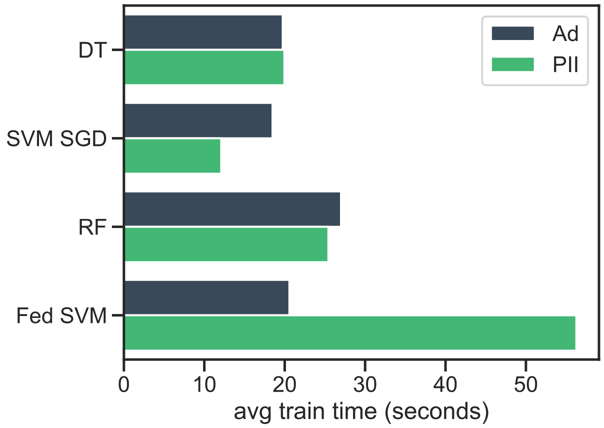


Figure 4.10: **Results 2c.** Comparison of average (from 5 runs) training time for Centralized models with NoMoAds data using HTTP Keys and Federated SVM with 20 synthetic users for both prediction tasks.

time. The aggregation time on the server side, t_{aggr}, is negligible as it takes only 40 microseconds. Overall, the SVM with SGD is comparable to DT and RF and the Federated SVM is comparable to the centralized models; PII task requires 3 rounds (each round taking approximately 145 seconds), while for the Ads task one round was sufficient resulting in faster training.

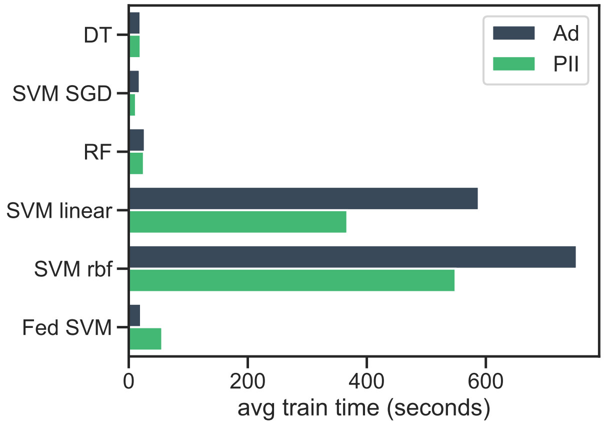


Figure 4.11: Comparison of average training time for all Centralized models from Table 4.4 with NoMoAds data using HTTP Keys and Federated SVM with 20 synthetic users for both prediction tasks.

Trained on	**Tested on**	**Uneven split F1 score**	**Even split F1 score**
Federated	user 0 test	0.91	0.93
Federated	user 1 test	0.94	0.95
Federated	user 2 test	0.95	0.94
Federated	user 3 test	0.95	0.91
Federated	user 4 test	0.93	0.93
Federated	all test data	0.93	0.93
Local user 0	user 0 test	0.92	0.89
Local user 1	user 1 test	0.91	0.91
Local user 2	user 2 test	0.93	0.92
Local user 3	user 3 test	0.87	0.87
Local user 4	user 4 test	0.85	0.89
Centralized	all test data	0.94	0.94

Table 4.7: **Results 3a.** AntShield dataset for predicting PII exposures, for 5 synthetic users created with uneven and even split of data. The F1 score is averaged from 5 runs for $C = 1.0, B = 10, E = 5$.

4.4.3 Scenario 3: AntShield for PII Prediction

Setup 3a. We use the following setup from Table 4.3. *Dataset:* AntShield. *Users:* Even vs. Uneven split with 5 synthetic users. *Classifier Granularity:* General. *Models:* Federated SVM vs. Centralized SVM, *Tasks:* PII exposure. We set $B = 10, E = 5, C = 1.0$, similarly to Setup 2.

Results 3a. Table 4.7 shows the results. For even split of data, the Federated model has an F1-score of 0.94 when it is tested on the union of user test sets, while the corresponding Centralized model has an F1 score = 0.96, achieved within 5.8 rounds on average. For the uneven split of data among users, the Federated model achieves the same F1 score = 0.94, but slightly slower (in 6.6 rounds). We observe that some users achieve lower F1 score on their corresponding Local models, which is expected as these users have much less data and especially positive examples, because of the skewness of data in the uneven split. In summary, we show that even with a different dataset, our FedPacket approach still performs well when compared to its Centralized model for both types of splits, with a small difference in communication rounds to achieve the same F1 score.

Setup 3b. We use the same setup as Setup 3a but with 100 users instead of 10. The goal is to evaluate convergence with more users who have few train data points (most of the users have 20-30 datapoints and only 3 users have more than 500 datapoints).

Results 3b. Fig. 4.12 shows the convergence in terms of F1 score across the rounds for AntShield with 100 users with uneven split. We observe that even if most users have few datapoints, the Federated model reaches the Centralized F1 score within less than 10 rounds. We also evaluated the average F1 score (from 5 runs) when we test the Federated model on each user's test data and the lowest F1 score for a user was 0.70 and only 20% of them had 0.90 or lower F1 score. To conclude, we showed that even if most users have few local data points, the Federated model will not overfit due to lack of training data. Moreover, we show the regularization effect of the Federated Averaging algorithm when varying the C parameter of participating users within each round. When $C = 0.5$ there is a regularization effect similar to dropout in DNNs, since only the parameters from half the clients are being averaged in each round which results in slightly more stable (less variance and higher) F1 score after round 10.

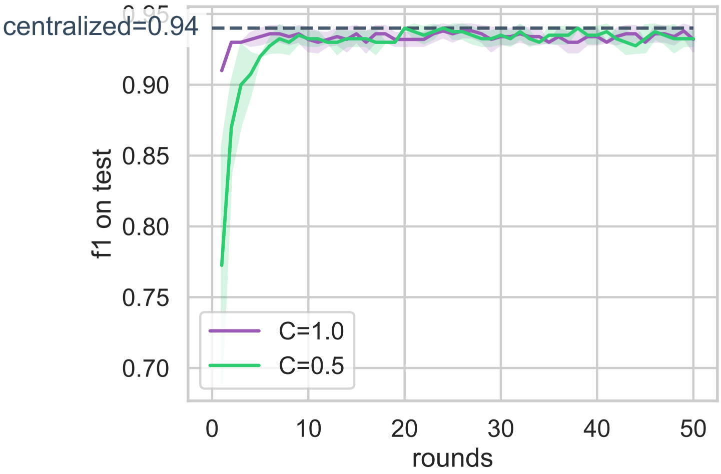


Figure 4.12: **Results 3b.** Convergence of AntShield with 100 synthetic users with uneven spit when $B = 10$, $E = 1$, and varied C.

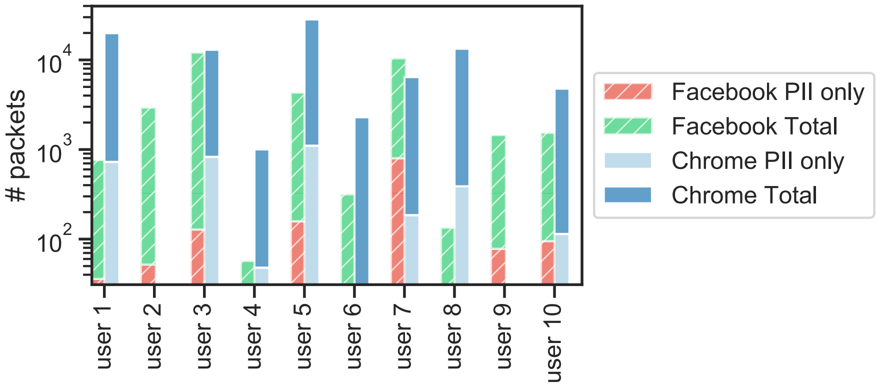


Figure 4.13: Distribution of packets for Facebook and Chrome.

4.4.4 Scenario 4: In-house Datasets for PII Prediction

Setup 4. We use the following setup from Table 4.3: *Dataset:* In-house Chrome, Facebook. *Users:* 10 real users. *Classifier Granularity:* Per App. *Models:* Federated SVM vs. Centralized SVM. *Tasks:* PII exposure. The goal is to evaluate our FedPacket framework (1) on real user activity (instead of systematic tests of apps) and (2) over a longer time period (7 months instead of five/ten minutes). Fig. 4.13 shows the distribution of Chrome and Facebook packets (including labels) present across the 10 real users in our in-house dataset.

Results 4a. We evaluate the classification performance of Centralized and Federated models for Chrome and Facebook, with $C = 0.5$, $B = 10$ and $E = 5$. Chrome's Federated model achieves 0.84 F1 score compared to its Centralized version with F1 score = 0.92. Facebook's Federated model maintains similar F1 score (0.94) compared to its Centralized version (0.95).

Results 4b. In Table 4.8, we evaluate the impact of batch size (B) and local epochs (E) on the average rounds (R) required to reach a target F1 score for Chrome and Facebook. We observe that increasing B increases slightly R to achieve the target F1 score, while increasing the E parameter increases R significantly. The reason is that we use a simple model and most likely the model overfits with large E. The FL paper [93] showed the opposite effect: increasing E decreases R; however, they train more complex models (DNNs) that do not overfit for those E values. Moreover,

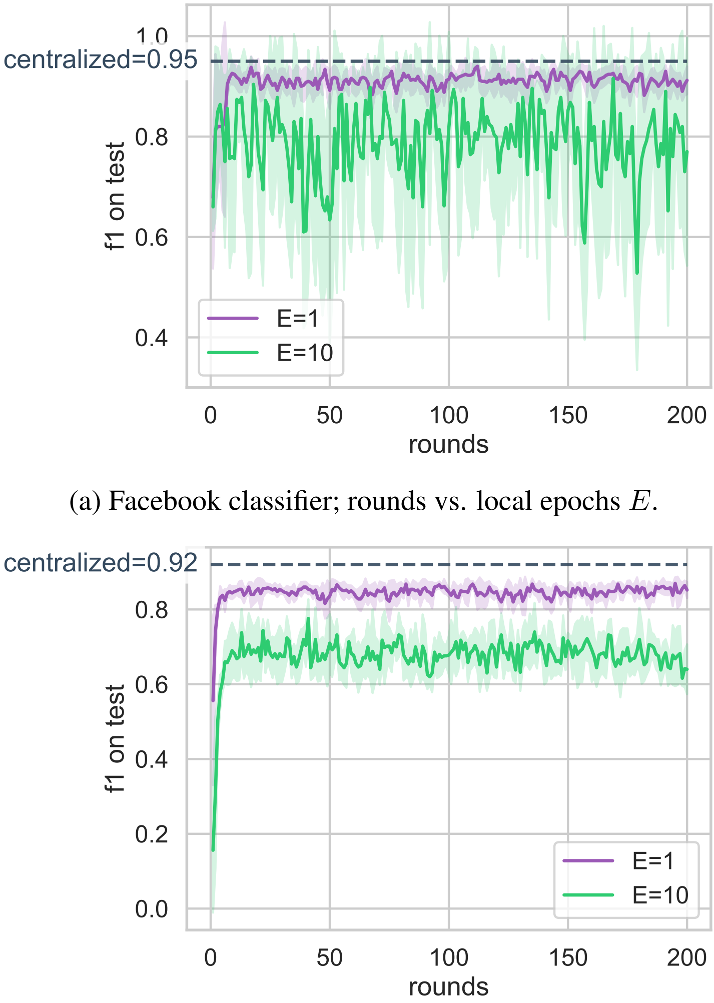
E	B	R: avg [min, max]	
		Facebook	Chrome
1	10	16 [6, 31]	7 [4, 10]
1	20	20.2 [12, 41]	6.4 [5, 11]
1	40	15.6 [8, 27]	5 [4, 7]]
1	∞	9 [6, 14]	6.2 [4, 11]
5	10	33.2 [5, 113]	82 [14, 200]
5	20	37.6 [20, 46]	27 [3, 61]
5	40	39.6 [8, 97]	25.6 [8, 55]
5	∞	26 [3, 47]	23.2 [6, 56]
10	10	53.8 [4, 190]	756.2 [531, 800]
10	20	71.6 [12, 200]	-
10	∞	72.8 [21, 146]	283.2 [126, 800]

Table 4.8: **Results 4b.** We report the average [min, max] R communication rounds required to reach a target F1 score (0.94 for Facebook, 0.84 for Chrome). We vary the batch size (B) and local epochs (E) to evaluate their impact on R, with $C = 0.5$. If the target F1 score is not reached within 800 rounds over 5 runs, we assume that it does not converge.

we observe that B does not significantly affect the number of rounds. In contrast, E plays an important role in the model's convergence, which is explored next.

Results 4c: Convergence of Federated models. Fig. 4.14 shows the performance of Federated SVM for Facebook and Chrome when we vary the local epochs, E, with $C = 0.5$ and $B = 10$. We train each model with an E value five times and report the average and standard deviation (in shadowed color). The main difference between the two apps is that the F1 score of the Federated model is closer to the Centralized one for Facebook, however, its standard deviation is much larger than Chrome's. In addition, $E = 1$ for Chrome can reach a better F1 score (0.89) than in the previous experiments, because of the lower E value. We observe that the Federated model is more sensitive to the E parameter, which leads to overfitting for SVM.

Results 4d: Learning Rate vs. Convergence. Throughout this chapter we used the "optimal" learning rate from scikit-learn [103] which is defined as $\eta^{(t)} = \frac{1}{a(t_0+t)}$ where t is the time step and t_0 is determined on a heuristic proposed by L. Bottou [41, 40] as $t_0 = \frac{1}{\eta_0 * a}$ where a is the regularization term. Fig. 4.15 shows the impact of the η on the Federated model's convergence



(a) Facebook classifier; rounds vs. local epochs E.

(b) Chrome classifier; rounds vs. local epochs E.

Figure 4.14: **Results 4c.** Convergence of F1 score over R rounds for Chrome, with $C = 0.5, B = 10$ and varied E. Models are trained 5 times, and shaded regions represent standard deviation from average F1 score. The Centralized model (dashed line) reaches F1 score 0.92.

for Facebook data with $C = 0.5$, $B = 10$, $E = 1$ similarly to Fig. 4.14a. We observe that only the smaller learning rate (η=0.01) requires more rounds to converge. Our chosen η reaches the target F1 score within 20 rounds and is comparable to $\eta >= 0.1$ and the "adaptive" η which is defined as: $\eta = \eta_0$ if the training loss keeps decreasing, otherwise $\eta = \frac{\eta}{5}$.

Discussion of Convergence. Understanding the effect of heterogeneity in terms of resources and data and how it affects convergence, especially in non-convex settings like DNNs, is currently an active research area within the FL literature [118, 133, 76, 85, 84]. Our model has a linear

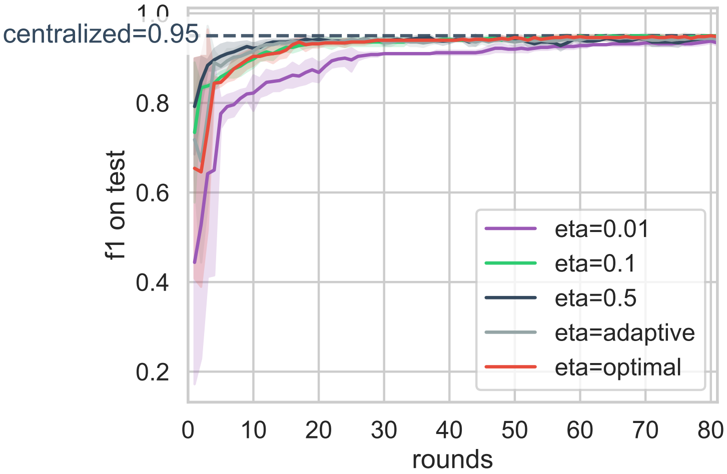


Figure 4.15: **Results 4d.** Convergence of F1 score over R rounds for Facebook with $C = 0.5$, $B = 10$, $E = 1$ and learning rate η varied.

kernel and the loss function is convex which guarantees convergence for both IID and non-IID data following the original FL paper [93]'s empirical conclusions which stated that for convex problems with E$\rightarrow \infty$ the global minimum will be always reached regardless of the model's initialization. However, the original FL paper [93] did not provide a mathematical analysis of convergence. The authors in [85] assumed strongly convex and smooth ℓ_2-norm regularized linear regression model and proved a convergence rate of O($\frac{1}{T}$) where T is the number of SGD updates during local training and data is non-IID. They also proved that a decaying η is necessary to guarantee convergence in case of non-IID data. Otherwise, they showed that a fixed η will converge to a solution at least $\Omega(\eta)$ away from the optimal. The reason is that constant learning rates combined with $E > 1$ generate biased local updates, and a decaying η can gradually tackle this bias. In our work, the Federated SVM is linear, ℓ_2-norm regularized and strongly convex but it is not smooth and we chose decaying η as discussed above. Overall, our experiments showed convergence consistent with the findings in [93] and [85].[5]

Results 4e: Benefit of Crowdsourcing. We ask the question: how many users need to collaborate

[5] We show throughout this work that the Federated model reaches convergence within tens of rounds for both IID and non-IID settings. Additional regularization-based methods [118, 142, 143, 84] can be considered in order to speed up further the convergence and are deferred to future work.

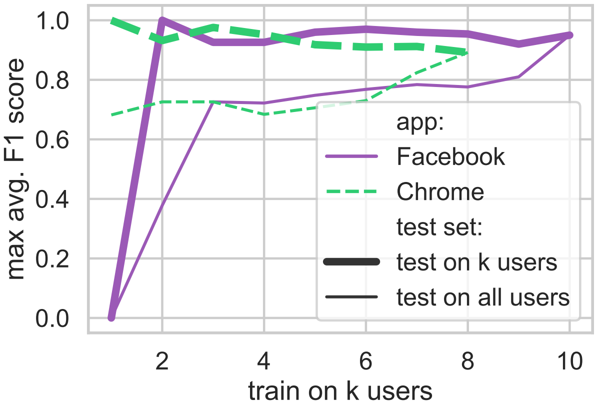


Figure 4.16: **Results 4e.** Benefit of crowdsourcing with k users for Chrome and Facebook. The average F1 score is shown for all users' test data (thin line) and for test data of k users (bold line).

to train a global model in order to get most of the predictive power? Fig. 4.16 shows that a few users participating in the training phase during FL can be beneficial for all users. We show the maximum average F1 score obtained from 5 runs, as a function of the number of users (k) participating in training. The F1 score is evaluated both on all user's test data and on the test data of k users who participated in the training. We sort the users by increasing amount of training data: for $k = 1$, one user with the fewest data points participates in training, for $k = 2$, in addition to the previous user, another user with more data is used during training. Adding more users in the training phase is beneficial to increase the F1 score for both apps. However, some users do not help and slightly worsen the F1 score, as their data might confuse the classifier. For Facebook, at least 3 users are needed to obtain a decent F1 score, while Chrome reaches the same F1 score with only 2 users. The F1 score on the test data of k users is much higher than on the union of all users' test data, as the models only fit to the data of the participating users. The lack of generalization is one of the reasons that Webview apps are a challenging special case in the packet classification problem. However, both Chrome and Facebook train Federated models generalize well with F1-score > 0.80, if enough users with useful (diverse) data participate in training, as the data of each user is generated by different usage of the apps. This shows the necessity of a crowdsourced model which is beneficial to all users, instead of training locally (F1 score with k=1 starts at 0).

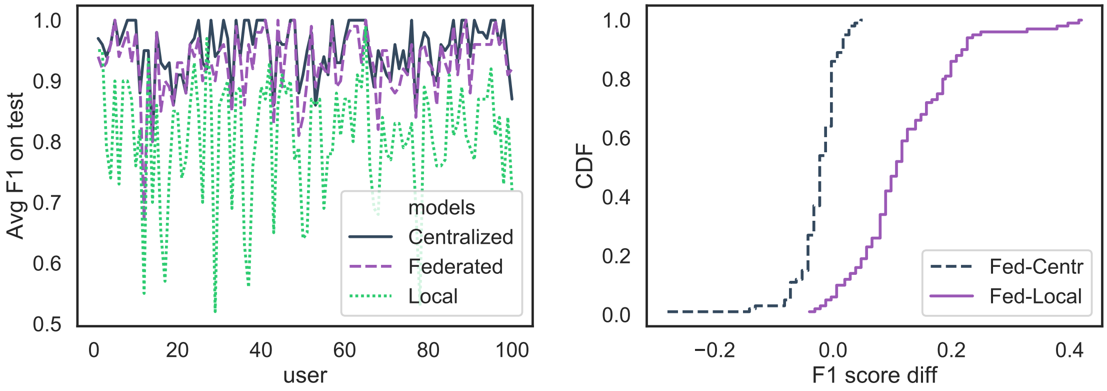
model:	Centralized	Federated	Local	Local
tested on:	**each user's data**	**each user's data**	**each user's data**	**all test data**
user 1	0.92	0.92	0.92	0.3
user 2	1.0	0.95	1.0	0.67
user 3	0.89	0.84	0.88	0.62
user 5	0.72	0.55	0.77	0.33
user 7	0.98	0.99	0.99	0.86
user 9	1.0	0.9	1.0	0.3
user 10	1.0	0.99	0.97	0.26

Table 4.9: For Results 4e, we compare the F1 score of Centralized vs. Federated vs. Local models, tested on each user's test data vs. test data from all users (merged). The Local model is better for some users than the Centralized and Federated models. However, when these Local models are tested on the test data from all users, the F1 score drops significantly. This is because these models do not generalize well for other users. (*Note:* We only show users who have some positive labels and omit the rest (users 4, 6, 8) whose F1 score is always 0.)

Table 4.9 reports more details for Results 4e. The Centralized and Federated models achieved similar F1 score when tested on each user's test data. The F1 score of Local models might be higher for some users (*e.g.,* user 5) than the Federated or Centralized models when tested on that user's data. However, when the Local models are tested on the data from all users, we observe a significant drop in performance: *e.g.,* the F1 score of user 1's Local model reaches 0.92 on their own test data, but it decreases to 0.3 on all test data. This is due to Local models overfitting on each user's data and not generalizing well across all users. This is even more pronounced for IID data, as described below.

Using Setup 3b with 100 synthetic users and uneven IID split of data from Sec. 4.4.3, we expand "Results 4e", "Benefit of Crowdsourcing", and consider 100 synthetic users instead of 10 real non-IID users to compares the performance of Federated to Centralized and Local models. The goal is to further demonstrate the benefit of moving away from locally trained models, to crowdsourced models, especially Federated (which essentially performs as well as a Centralized model).

Fig. 4.17 shows the results for this scenario. We compare the F1 score on each user's test data for crowdsourced models: Centralized (their training data was shared with a server) and Federated (only model parameters were shared), and their corresponding Local models (they shared nothing).



(a) Raw values of F1 score of Federated, Centralized and Local models when tested on each user's test data.

(b) Empirical cumulative distribution function (CDF) of the differences of F1 scores for (Federated − Local) vs. (Federated − Centralized).

Figure 4.17: Comparison of Local vs. Federated vs. Centralized models when tested on each of the 100 uneven synthetic users with AntShield data. Federated F1 score is comparable to Centralized and both perform better (positive difference in F1) than the corresponding Local models. All users benefit from the crowdsourced models due to IID nature of the data, but at a different degree: the increase in F1 can be up to 0.4, with 80% of the users up to 0.2.

Fig. 4.17(a) shows the F1 score achieved from Federated, Centralized and locally trained models for each of the 100 users when tested on each user's test data. Fig. 4.17(b) shows the empirical CDF of the difference between the F1 score of Federated and Centralized models and between the Federated and Local models. We make two observations by looking at Fig. 4.17(b). First, the Federated achieves similar F1 score to the Centralized model for 90% of the users, except for a few users, that Centralized performs slightly better. Second, the Federated model performs better than the corresponding Local models: for 80% of users the Federated F1 score reaches an increase up to 0.2, compared to the Local model, and for some users it is almost 0.4, which is significant. In summary, all users benefit from the use of crowdsourcing, *i.e.,* there are positive differences in F1 score for (Federated-Local), but at different degrees.

4.4.5 Scenario 5: Client Selection and Convergence

In all previous experiments, the clients were selected randomly in each FL round as in the original FL paper [93]. Here, we explore how different client selection strategies affect convergence when the local data of each client is non-IID (Setup 5a) vs. IID (Setup 5b). We explore a second client selection strategy, which we refer to as "data size": clients are selected with probability based on their training data size as in [85], such as $P = \frac{data_k}{total}$, where total is the amount of all training data summed from all users and $data_k$ is the training data of user k. We would like to note that in the aggregation step in Algorithm 1, the updates are weighted based on the training data of each client, giving more importance to the updates of clients with most data. So, with the "data size" client selection strategy those users are "boosted" even further, which can lead to overfitting. For this reason, we added a third client selection strategy, which we refer to as "inverse data size", and chooses clients to participate in each round with probability inversely proportional to the size of their training data, thus assigning higher probability to clients with few data.

Setup 5a. We use the following setup from Table 4.3: *Dataset:* In-house Facebook. *Users:* 10 real users. *Classifier Granularity:* Per App. *Models:* Federated SVM. *Tasks:* PII exposure. The goal is to evaluate the convergence of the Federated model when different client selection strategies are in place for non-IID users.

Results 5a: Non-IID clients. We extracted the probabilities of each user being selected in a round based on their training data size. User 7 has a 50% chance of being selected in a round since their data had the most PII positive packets according to Fig. 4.13 and thus, with data balancing they have the most training data compared to the rest of the users. The rest of the users had probability of being selected less than 0.1. Fig. 4.18 shows the convergence for three different client selection strategies in case of non-IID data. With random sampling the F1 score is more stable across rounds however it starts with lower value. In contrast, the "data size" strategy starts with F1 score of 0.90 since the user with the most data participates in almost all rounds and only the rest of the users vary.

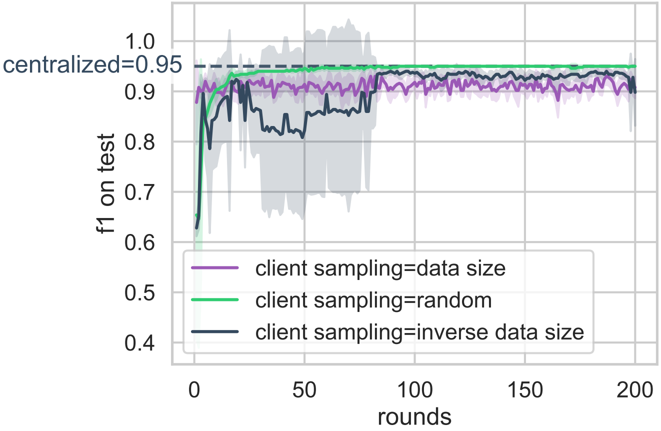


Figure 4.18: **Results 5a.** Convergence of F1 score over R rounds vs. various client selection strategies for Facebook (non-IID) data with $C = 0.5$, $B = 10$, $E = 1$.

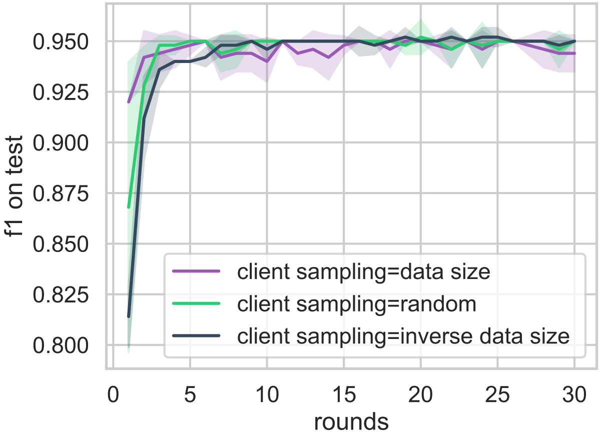


Figure 4.19: **Results 5b.** Convergence of F1 score over R rounds vs. various client selection strategies with NoMoAds 20 synthetic (non-IID) users when predicting PII with $C = 0.5$, $B = 10$, $E = 1$.

Moreover, "data size" reaches a slightly lower F1 score as the Federated model seems to slightly overfit to user 7's data. Overall, the difference between the two strategies is not significant; the model's F1 score is still above 0.90. However, "inverse data size" is significantly lower with high variance in the first 50 rounds. After the 50th round, it exceeds the F1 score of the other two strategies and shows low variance. Thus, even if clients with few data points are selected for their updates, after a certain number of rounds the model still converges due to the regularization effect similar to dropout in DNNs, as $C = 0.5$ requires half of the clients to send their model updates.

Setup 5b. We use a similar setup to Setup 5a, but with NoMoAds 20 uneven synthetic users. *Classifier Granularity:* General. *Models:* Federated SVM. *Tasks:* PII exposure, Ad request. The goal is to evaluate the convergence of the Federated model when different client selection strategies are in place for IID clients.

Results 5b: IID clients. Fig. 4.19 shows the convergence of the Federated model for various client selection strategies when the clients are IID. We observe similar effects to the non-IID case, except that the "inverse data size" strategy does not have as significant impact. Thus, in the case of IID synthetic clients, the convergence is not affected by the client selection strategy and random client selection seems to perform well. We omit the comparison between the two prediction tasks and even, uneven splits due to space limit since the observations were identical.

4.4.6 Scenario 6: Interpreting SVM vs. DT

Setup Scenario 6. We use the following setup from Table 4.3: *Dataset:* NoMoAds. *Users:* None. *Classifier Granularity:* General. *Models:* Centralized SVM vs. DT. *Tasks:* PII exposure. Prior work chose DT over other models partially because of their interpretability. In our context, these models learn similar separation of our datasets, which we demonstrate by (1) observing the most important coefficients in SVM, (2) by knowledge transfer from SVM to DT. The goal here is to compare SVM to DT in terms of their interpretability.

Results Scenario 6. Fig. 4.20 shows the ten most important negative and positive coefficients and their corresponding features for our Centralized SVM. In order to distinguish important features, we use the model's coefficients, where the positive ones correspond to the features whose presence leads to positive labels and the negative coefficients correspond to features responsible for predicting label 0 (*e.g.,* No PII detected). This is not a one-to-one mapping of important features between SVM and DT due to their internal representation of features. However, we observe certain keys

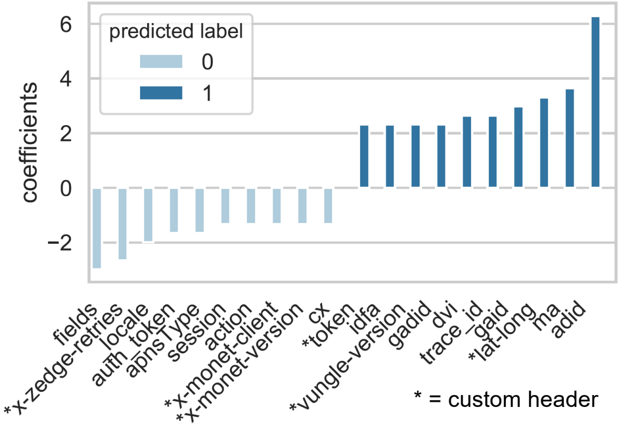
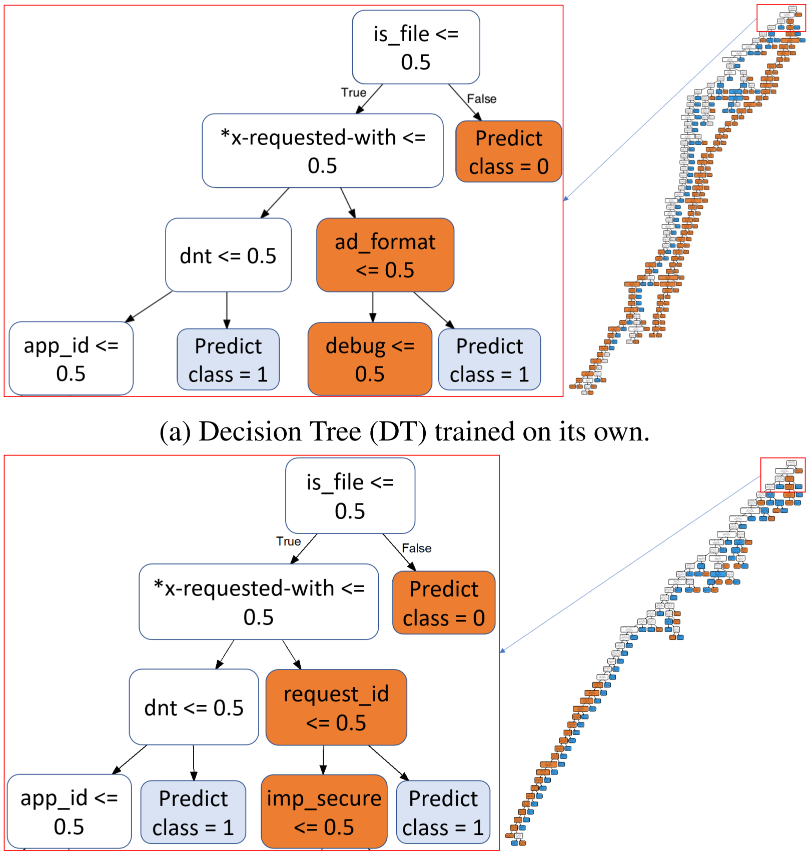


Figure 4.20: Top 10 negative and positive coefficients and the corresponding features obtained from Centralized SVM.

that are responsible for PII exposures such as "gaid", that also appear in the corresponding DT.



(a) Decision Tree (DT) trained on its own.

(b) First train SVM, then transfer knowledge to DT.

Figure 4.21: Interpretability of DT vs. SVM in Setup 5.

Fig. 4.21(b) shows the DT after knowledge transfer from SVM. To perform knowledge transfer from SVM to DT, we first split the data into 40% for training SVM, another 40% for training a DT, which is labeled with predictions from the aforementioned SVM. The remaining 20% of the data is used for testing. This is one way to leverage the interpretability of DTs via knowledge transfer from SVM. In Fig. 4.21(a), we show a DT which was trained with NoMoAds for PII prediction, while in Fig. 4.21(b), we show the DT after knowledge transfer from SVM. We observe that both DTs, at least at the top levels, have similar important features and thus, capture similar patterns. The original DT and SVM reached F1 score = 0.95 and the after knowledge transfer DT reached F1 score = 0.94 on the same test data. This is only a minor F1 score loss during knowledge transfer.

The most notable difference between the trees in Fig. 4.21 is the lack of a large branch that only predicts label 0, which is the result of how the original tree unsuccessfully attempts to separate data. However, the DT after the knowledge transfer is oblivious to this error, since the SVM most likely suffers from the same issue as the original DT. Such errors propagating from the SVM make the DT after the knowledge transfer smaller (269 vs. 141 nodes) than the original DT. Please refer to the appendix of the extended version of this work [31] for full pictures of the above DTs.

4.5 FedPacket: Privacy Considerations

The FL framework clearly raises the privacy bar in mobile packet classification, by allowing devices to collaboratively train a classifier, without uploading their raw packet traces or training data to a server.[6] However, federated learning, and more generally distributed learning, has its own inherent vulnerabilities to inference attacks based on observed updates [96, 98, 147, 64, 138]. In this section, we consider two inference attacks specifically designed against packet classification. These attacks are application-specific in the context of HTTP data, as opposed to generic *e.g.,* adversarial attacks against federated image classification.

[6] Crowdsourcing training data from users to the server is the current practice in mobile data analytics, including mobile packet classification. However, it can directly expose personal and device identifiers, location and other sensitive information stored on the device, as well as enable inference of other sensitive information about user behavior.

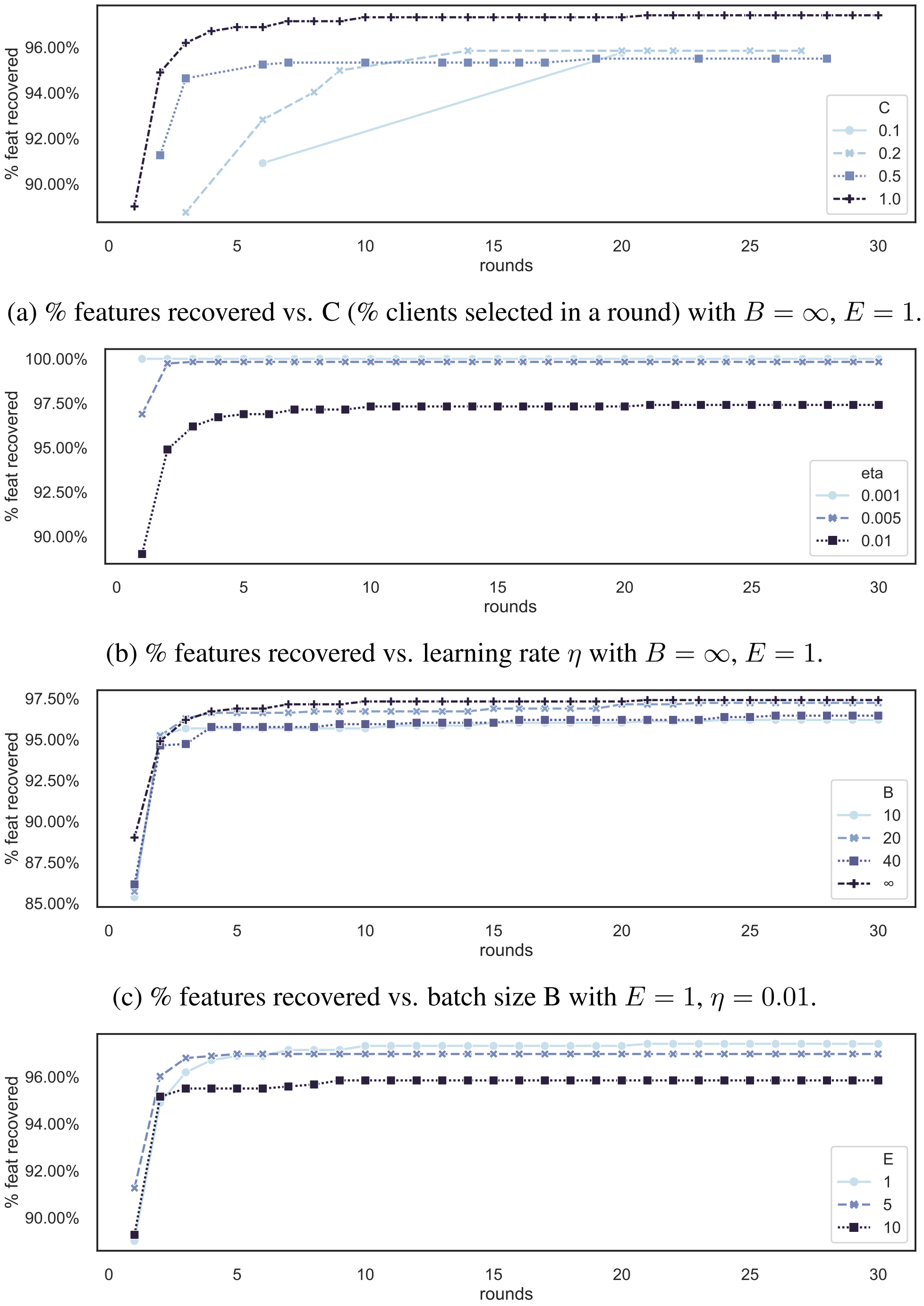
4.5.1 Inference Attacks

Threat model. We assume an honest-but-curious server. It is honest because it receives updates from all users, it computes and sends back to them the updated model parameters, correctly and without any modification. It is curious because it wants to infer sensitive information about a target user. In each FL round, the server observes the gradient updates, analyzes and stores them for future use. Since no additional privacy mechanism, such as Differential Privacy [53] or Secure Aggregation [38] is assumed, at this point, the server knows the exact updates sent by each user and can target users individually to infer sensitive information from their updates. In Threat Setup 6a, the server aims to recover the features of a target user via observing and storing their non-zero gradients in each FL round. In Threat Setup 6b, we assume that the server has already reconstructed successfully all local training data of the target user and further attempts to infer additional information related to the target's browsing history.

Threat Setup 6a: Feature Recovery. We use the following setup from Table 4.3: *Dataset:* In--house Facebook. *Users:* User 7. *Classifier Granularity:* Per App. *Models:* Federated SVM. *Tasks:* PII exposure. The goal is to evaluate how the FL parameters affect the success rate of the attack in terms of % of features recovered.

Results 6a: Feature Recovery. The server observes the gradient updates from users participating in each FL round. In every round, the adversary stores the current gradients in order to subtract them in the next round. Recall that the server receives the updated model weights from each user, which are obtained locally via $w \leftarrow w - \frac{\eta}{B} \sum_{i \in B_k} y_i \cdot x_i$ as already mentioned in Algorithm 1. Fig. 4.22 shows the percentage of recovered features in each round during FL when we vary the FL parameters: batch size B, local epochs E, percentage of clients C in each round and learning rate η. Fig. 4.23 shows how the model's convergence is affected with selected parameters from the privacy attack.

One of the parameters that affects the success of the attack the most is the fraction C of participat-



(a) % features recovered vs. C (% clients selected in a round) with $B = \infty$, $E = 1$.

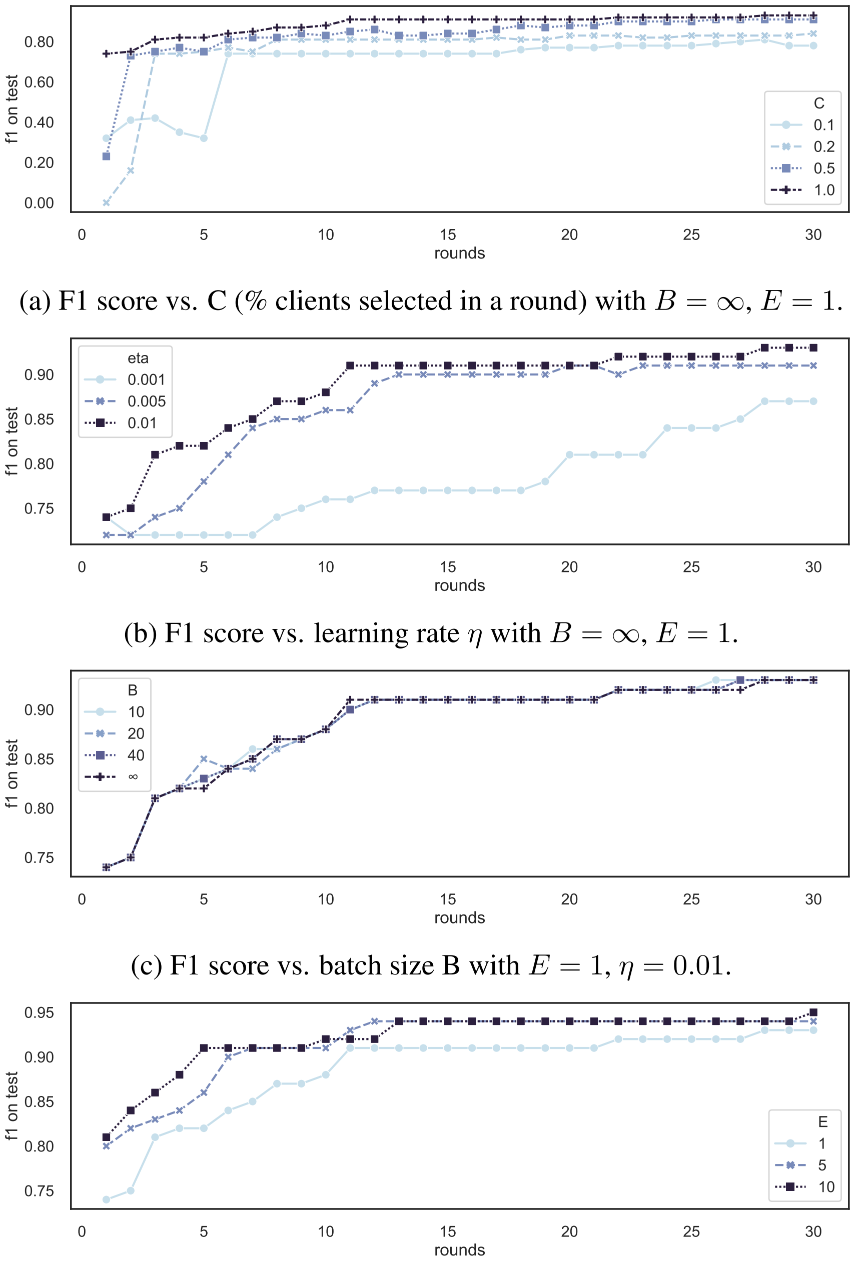
(b) % features recovered vs. learning rate η with $B = \infty$, $E = 1$.

(c) % features recovered vs. batch size B with $E = 1$, $\eta = 0.01$.

(d) % features recovered vs. local epochs E with $B = \infty$, $\eta = 0.01$.

Figure 4.22: Evaluating the success of our privacy attack in terms of features recovered (%) when we vary the FL parameters.

ing clients, as shown in Fig. 4.22a, since the target user might not participate in every round due to random selection of clients in each round. Fig. 4.23a shows that selecting $C = 1.0$ speeds up



(a) F1 score vs. C (% clients selected in a round) with $B = \infty$, $E = 1$.

(b) F1 score vs. learning rate η with $B = \infty$, $E = 1$.

(c) F1 score vs. batch size B with $E = 1$, $\eta = 0.01$.

(d) F1 score vs. local epochs E with $B = \infty$, $\eta = 0.01$.

Figure 4.23: Convergence in terms of F1 score for selected FL parameters corresponding to privacy attack.

the convergence in addition to speeding up the privacy attack, since the target user is participating in every round. However, if the server can control the percentage of clients or can select the target user regardless of the C parameter, then the attack can be successful in fewer rounds.

Another parameter that affects significantly the feature recovery rate is the learning rate η. Fig. 4.22b indicates that a smaller learning rate can recover 100% of user features within few rounds. However, η affects the performance of the global model and convergence; smaller η slows down convergence, as shown in Fig. 4.15. We demonstrate this privacy-utility trade-off in Figures 4.22b and 4.23b: smaller learning rate will speed up the privacy attack but will impact negatively the model's convergence. For instance, setting η to 0.001 speeds up feature recovery, but the F1 score of the federated model drops from 0.93 (with $\eta = 0.01$) to 0.87. From the attacker's point of view, the server must choose a learning rate to balance the trade-off between the feature recovery rate and the global model's F1 score. From privacy-preserving point of view, larger η accelerates the convergence and slows down the feature recovery. Due to this trade-off, we chose $\eta = 0.01$ and evaluate the B and E parameters. Fig. 4.22c shows how the batch size affects our privacy attack. Smaller batch size seems to result in faster feature recovery, although the difference is not very significant. Similarly, the convergence of the model is not affected significantly by the B parameter, as shown in Fig. 4.23c. Finally, Fig. 4.22d indicates that smaller local epochs E results in better feature recovery, as increasing local epochs introduces a notion of aggregation before the server receives the model updates from the client. However, Fig. 4.23d shows that with $E = 1$ the model converges slower and the best trade-off is achieved when $E = 5$ in terms of attack success and convergence.

Overall, we showed that more than 90% of features can be recovered within tens of rounds if the server controls the client selection and asks the target user for their updates, when the learning rate is reasonably small regardless of the B and E parameters. The next question we ask in Setup 6b is what additional sensitive information can be inferred from these recovered features.

Threat Setup 6b: Predicting visited domains. We use the following setup from Table 4.3: *Dataset:* In-house Facebook. *Users:* User 7. *Classifier Granularity:* Per App. *Models:* Centralized SVM. *Tasks:* Domain.

The goal is to show that additional sensitive information can be inferred from a successful attack on features. Specifically, we assume that the privacy attack from Setup 6a (attack to FL updates) was successful and the server was able to recover *all* features of a target user, which is an upper bound in practice. The question is then: can the server also infer the user's browsing history, *i.e.,* the domains the user visited, based on all the HTTP keys? How well can the attacker predict the visited domains based on the Recon features, that contain more sensitive information than HTTP keys? Browsing history is only one, but important, example of sensitive information that can be inferred from HTTP data.

Results 6b: Predicting visited domains. In this experiment, we assume the attacker has already recovered all features of the target user. If sensitive features are included in the feature space, as in All Words or Recon Words, *i.e.,* values to sensitive keys, then the attacker will able to recover those within few rounds as we showed in the previous experiment. Although HTTP Keys do not have *explicit* information from about the URL path or domain, it is possible that such sensitive information can be *inferred* from HTTP Keys. Here, we ask the following question: how well can we predict visited domains based on HTTP Keys? As a baseline for comparison, we also predict domains from Recon Words features, which actually contain parts of the URL path. For a fair comparison, we are testing the performance on the intersection of common domains on the test data for both Recon Words and HTTP Keys experiments: there were 105 such common domains. We train a Centralized SVM model with HTTP Keys (2,411) vs. Recon Words (5,120) feature space.

To provide a summary of all 105 domains,[7] we report the macro average F1 score, *i.e.,* the non-weighted average of all per-domain F1 scores. With HTTP Keys, the macro average F1 score was 0.55 and approximately 48.57% of domains reached F1 score $>$ 0.80. In contrast, Recon Words achieved 0.60 macro average F1 score and 62.86% of the test domains reached F1 score $>$ 0.80. Recon Words increase the performance of domain predictions and thus the attacker can infer better such sensitive information from user data. This was expected since Recon Words contain more

[7] The results for all 105 common domains, are presented in Fig. 4.24 and a zoomed-in version with top 30 (in alphabetical order) domains in Fig. 4.25.

	HTTP Keys	Recon Words
# ad/tracking domains (ATS)	39 (76.5%)	38 (57.6%)
# non-tracking domains (non-ATS)	12 (23.5%)	28 (42.4%)
total domains	51 (100%)	66 (100%)

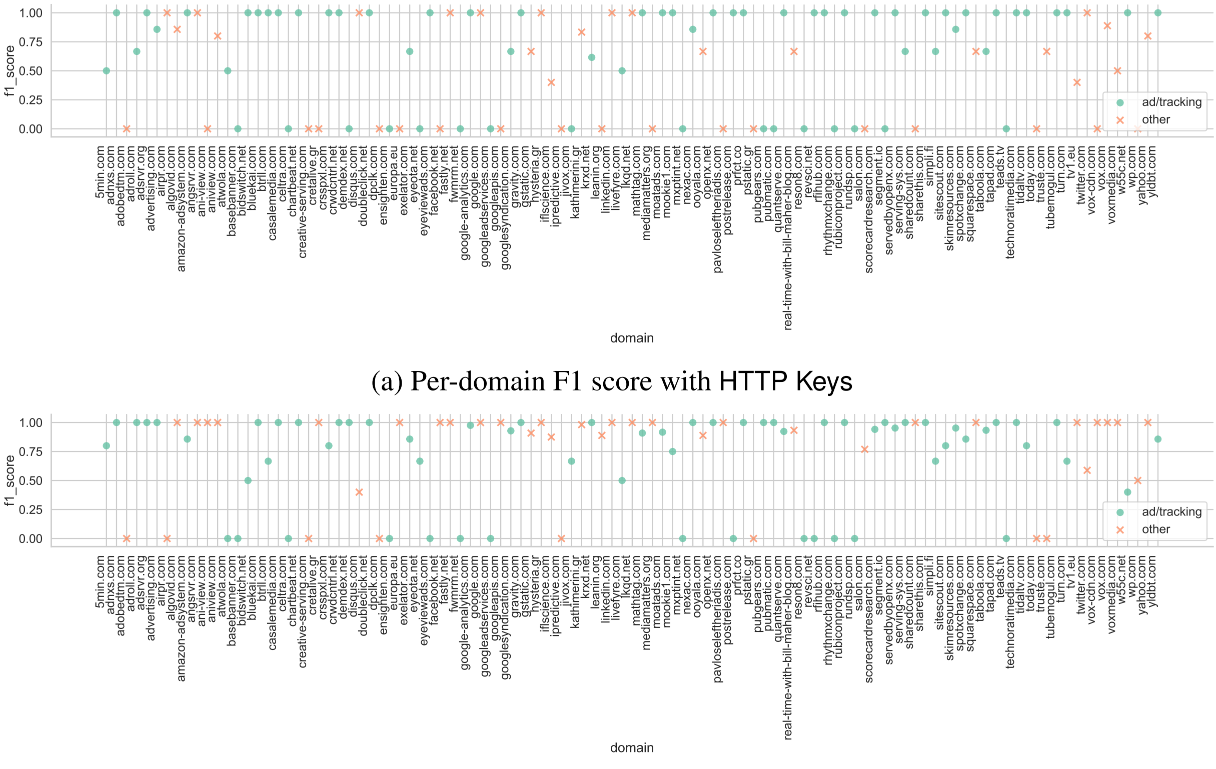
Table 4.10: Summary of domains that reached F1 score above 0.80 when using two different feature spaces: HTTP Keys and Recon Words. Recon Words resulted in more domains that were predicted successfully and many of those domains were non-advertising/tracking resulting in a higher privacy risk.

information from the packet and especially the URL field. However, it was previously unknown how well an attacker can infer domains and this is the first work that quantifies such leakage.

Next, we further distinguish domains that provide advertising and tracking services (a.k.a. ATS) from the rest of the predicted domains. We consider the non-ATS domains to be more sensitive as they represent the user's browsing history in Webview apps like Facebook. We used the Mother-Of-all AdBlockers (MOaD) [12] filter list and the module AdblockRules from AdblockPlus [2] to label ATS domains. Figures 4.24 and 4.25 show the results for all domains and for the top 30 (sorted alphabetically) domains respectively. Some examples of non-ATS domains are the following: europa.eu, facebook.net, vox.com, while some ATS domains: doubleclick.net, google-analytics.com, openx.net. Table 4.10 shows how many non-ATS domains were predicted well, *i.e.,* achieving F1 score above 0.80, based on Recon Words compared to HTTP Keys. In particular, Recon Words predict more domains (66) successfully than HTTP Keys (51). Recon Words result in prediction of 42.4% non-ATS domains compared to 23.5% with HTTP Keys, thus increasing the risk of sensitive information revealed to the attacker (server).

4.5.2 Mitigation via Aggregation

There are two families of defense mechanisms that are usually applied on top of federated learning: differential privacy [53] (and other types of noise, including federated GANs [29]) and Secure



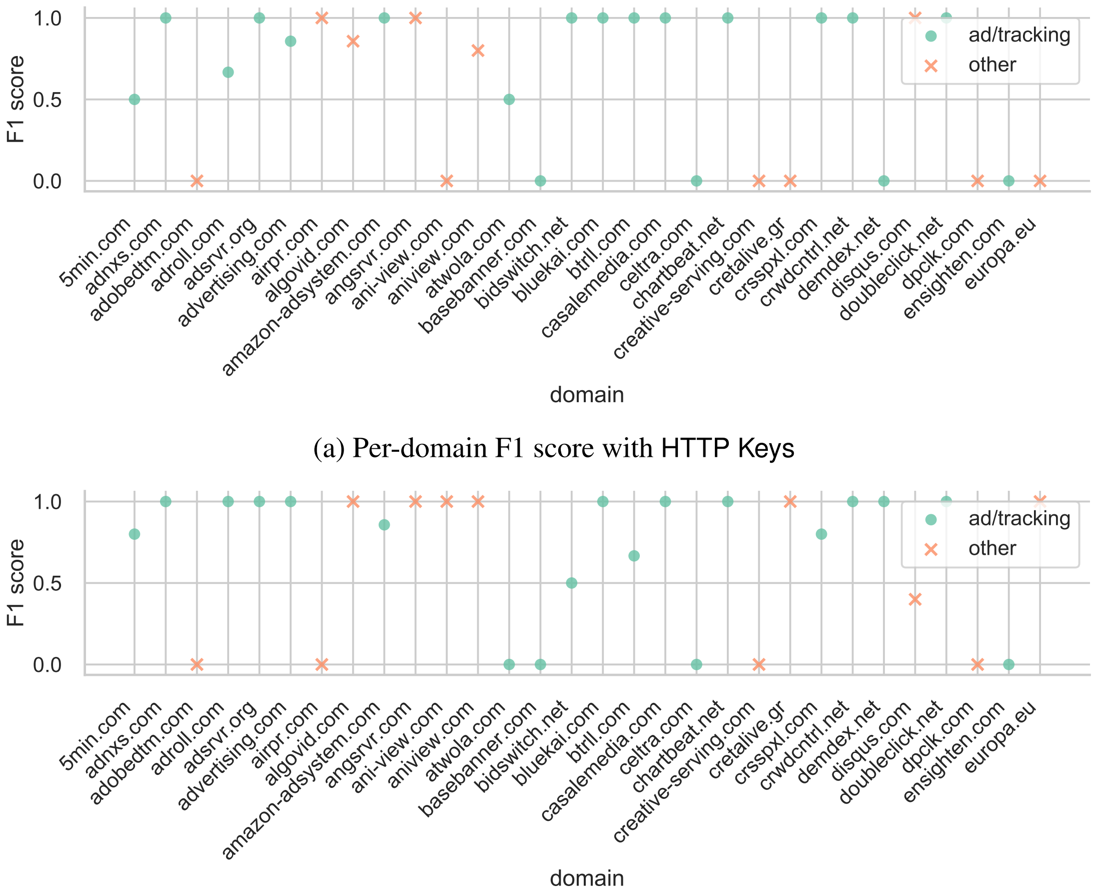
(a) Per-domain F1 score with HTTP Keys

(b) Per-domain F1 score with Recon Words

Figure 4.24: Comparison of per-domain prediction with SVM and two feature spaces: Recon-Words and HTTP Keys for all 105 domains. "Advertising and Tracking" (ATS domains), marked with "o", are usually contacted by third party libraries used by mobile apps, and are thus less sensitive. "Other" (non-ATS) domains, marked with "x", reflect the domains the user actually intended to visit and are more sensitive.

Aggregation (SA) [38]. Recall that in both attack scenarios described above, the honest-but-curious server had access to the updates from the target user, which it used to infer features (in 6a) and visited domains (in 6b) for that target user. To defend against the particular attacks for our problem, SA seems naturally suited to hide which updates come from which user.

SA [38] was proposed early as a defense mechanism added on top of FL. It is a multi-party computation (MPC) mechanism that enables clients to submit their updates to the server, but the server sees only the aggregate of the updates needed for learning. A user's gradient is aggregated with a set of $k-1$ other gradients, from ($k = C\dot{K}$) users sending updates within the FL round, and cannot be traced back to the individual user. Intuitively, the more clients participate in a round (larger k), the better the protection in the k-anonymity sense. However, the value of k also affects the MPC,



(a) Per-domain F1 score with HTTP Keys

(b) Per-domain F1 score with Recon Words

Figure 4.25: Zoomed-in version of per-domain prediction with SVM and two feature spaces: Recon Words and HTTP Keys.

the communication and computation cost of FL and the convergence. We show that even a small k (*e.g.,* $k \geq 3$) provides good enough protection (*i.e.,* reduces the number of inferred features to 65%) even for the strongest hypothetical adversary.

Threat Setup 6c, with Secure Aggregation (SA). We assume that SA is used and the serve can only observe the aggregate of the gradients of k users participating in each FL round. For $k = 1$, this is the Attack Scenario 6a discussed before, where the server could see the updates of the target user. For $k > 1$, the server observes the aggregate non-zero gradients from a set of k users, including but not limited to the target. It can keep track of different sets and features seen in the previous and current rounds, in order to infer the features of the target. To that end, there are many possible inference algorithms the server could implement.

We implemented a heuristic that carefully picks the sets of k users to pull updates from, in every round. It maintains counts of users that participated in sets that had non-zero gradient for a certain feature (in a matrix $I(user, feature)$). We outline the main idea of the algorithm 3.

I Consider all subsets of k users, including the target user. For each k-subset, pull the secure aggregate of the gradients, identify the features with non-zero gradient. Update the matrix I based on the following rules: (i) if a feature appears in a round but did not appear in previous rounds, conclude that users that participated in earlier rounds do not have it and update the count for users in k-subset, (ii) if a feature appears in previous and the current round, update the count for users in the k-subset, (iii) if a feature appeared in previous rounds and not in the current round, the current users do not have it.

II Exclude the target user, consider $(k-1)$-subsets of other users and repeat Phase I for those subsets. If a feature does not appear in Phase II but appeared in Phase I, we are sure that the target user has it. If a feature appears in Phase II but not Phase I, we are sure that the target user does not have it.

The algorithm eventually infers which features are present or absent in the target user, some deterministically (0 or 1), others with varying degrees of confidence (a number between 0 and 1), reflecting the fraction of rounds that a user participated when the feature appeared. Finally, we focus on the target user, and apply a confidence threshold: features with confidence above or below that threshold are declared as present or absent, respectively.

Results 6c. Fig. 4.26 evaluates the success of the inference algorithm in Threat Setup 6c, in a way that is consistent with the evaluation of Threat Setup 6a ($k = 1$): we report the percentage of features recovered for the target user (user 7) at various confidence thresholds. As expected, the attack is less successful when more users participate in a round (larger k), thus anonymizing

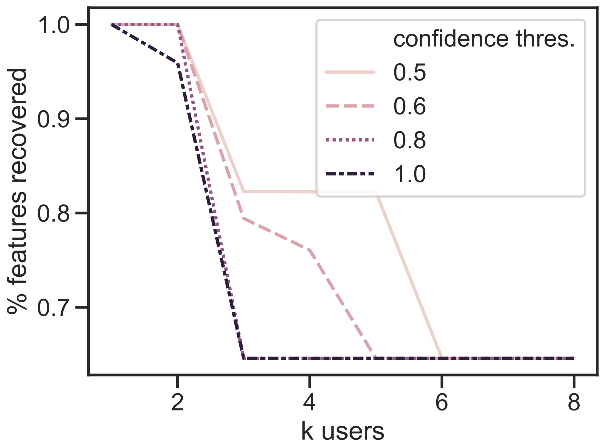


Figure 4.26: Evaluating Attack Algorithm 3, with secure aggregation on. We report the percent of the target user's recovered features, for varying k (participating users in a round) and confidence thresholds.

the inferred features within a group of k users. With confidence threshold 1, we observe a sharp decrease for $k \geq 3$ leading to 65% recovered features. The decrease is less sharp and offers less protection for $k \leq 5$ when confidence threshold < 0.8, which shows a trade-off between % of features recovered vs. confidence of attributing those features to the target user. Overall, the server recovers correctly with high confidence only 65% of the features if at least 3 users participate in the FL rounds.

Fig. 4.27 shows the accuracy of the privacy attack with Secure Aggregation on. Moreover, we show more statistics about the participating users from in-house Facebook dataset regarding their feature (HTTP Keys) size (Fig. 4.28) and their pairwise similarity based on common features (HTTP Keys) (Fig. 4.29). Finally, we show in Fig. 4.31 the results when other users are targeted.

Summary of privacy protections. First, we showed that using HTTP Keys instead of Recon-Words as feature space is more privacy-preserving: we use only keys and not values or packet fields that contain sensitive identifiers and other information. Second, the FL framework prevents uploading raw training data from devices to the server. Third, the inference of features and other sensitive data from observed updates is an inherent vulnerability to all distributed learning. Although HTTP Keys reduce the success of a domain classifier or leakage of features themselves,

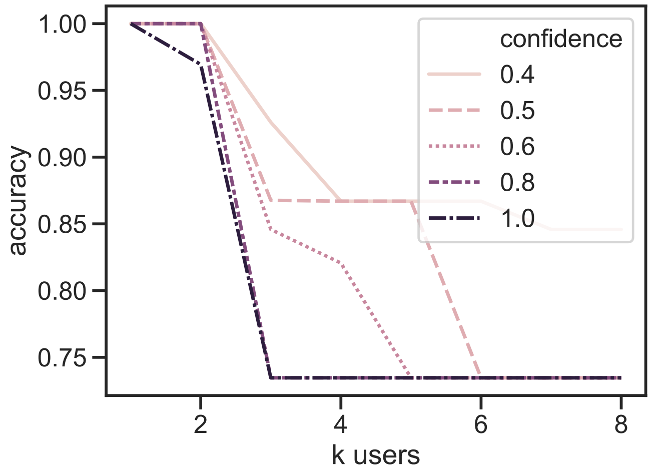


Figure 4.27: Evaluating Attack Algorithm 6c, with secure aggregation on. We report the accuracy ($\frac{TP+TN}{T+P}$), for varying k (participating users in a round) and confidence threshold.

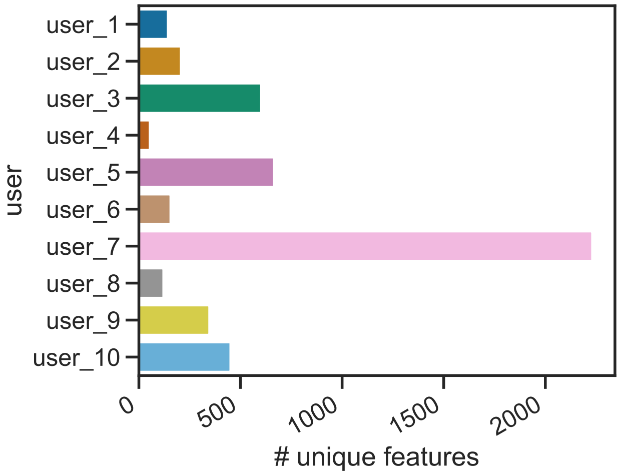


Figure 4.28: Per user unique HTTP Keys features for in-house Facebook dataset.

there are still privacy risks involved with unprotected gradients in FL. We evaluated and quantified the leakage of features and browsing history specifically for HTTP packets of a target user. We also showed that Secure Aggregation (a well-known form of MPC) can significantly help remedy this problem: even a small number of participating users per round ($k \geq 3$) can reduce the number of features deterministically inferred to 65%.

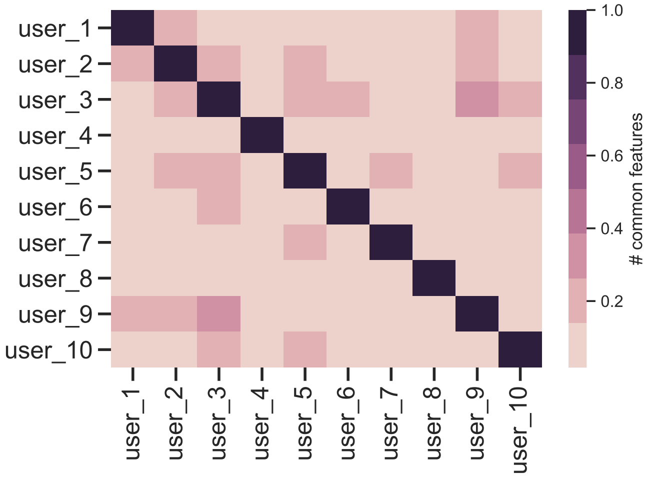
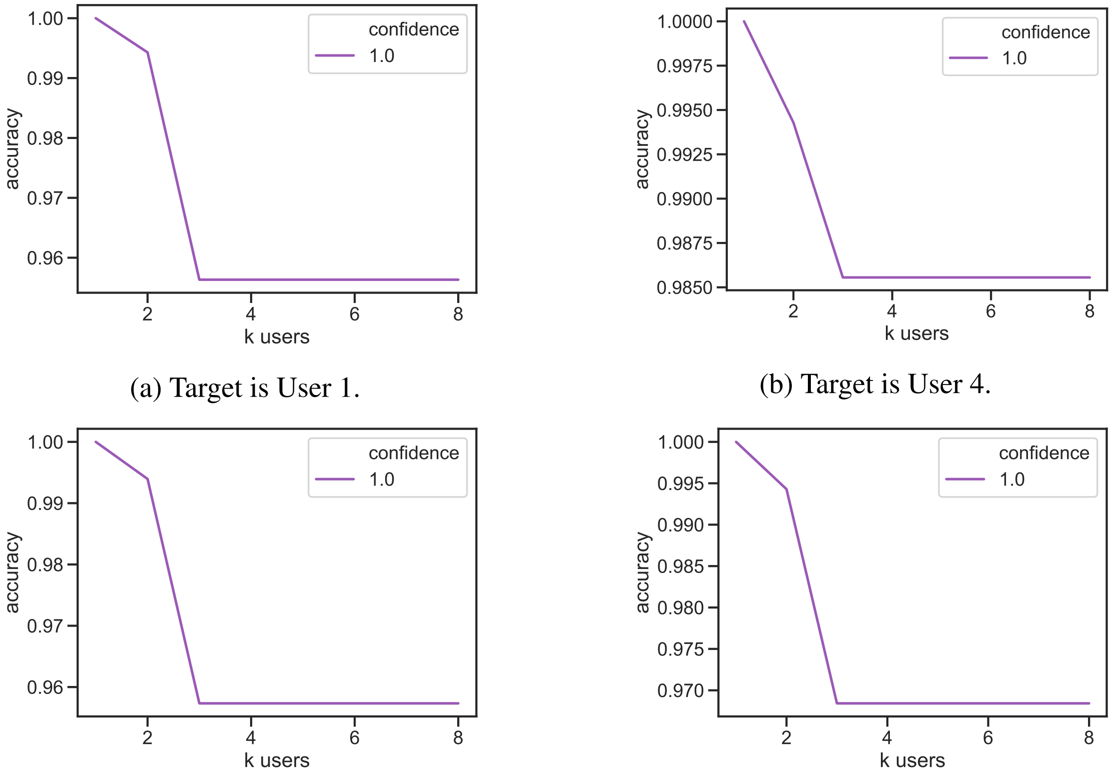


Figure 4.29: Pairwise user similarity based on Jaccard similarity of common features for in-house Facebook dataset.



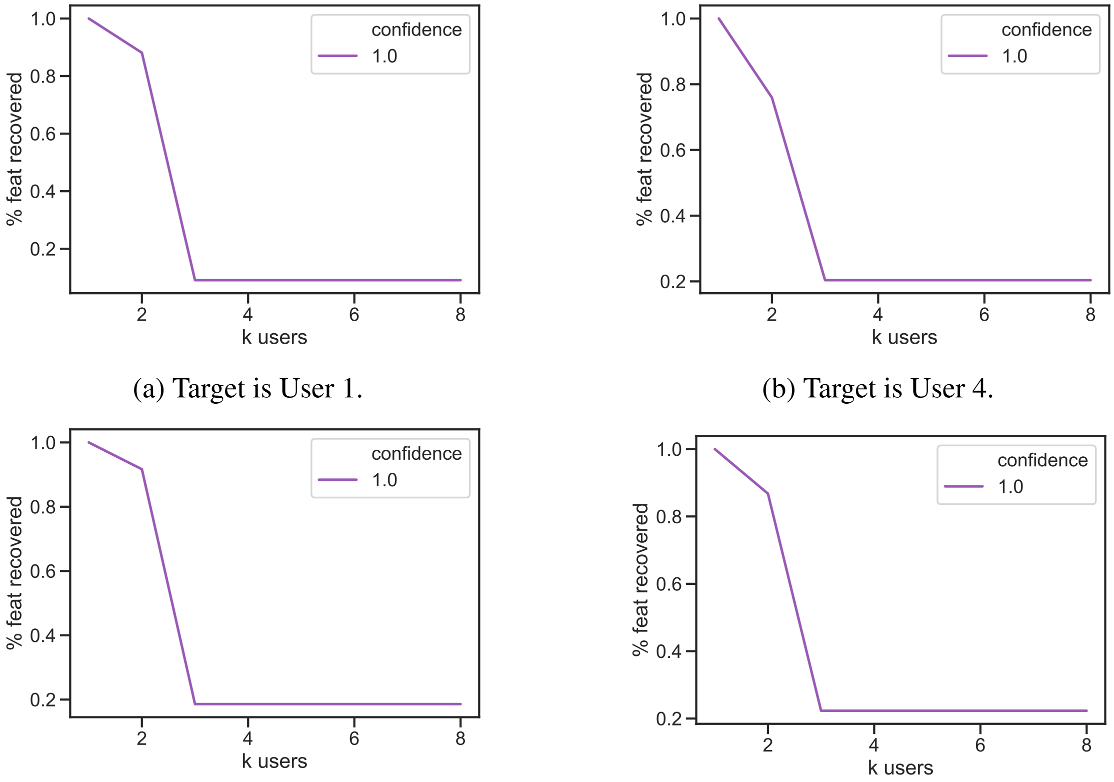
(a) Target is User 1.

(b) Target is User 4.

(c) Target is User 6.

(d) Target is User 8.

Figure 4.30: Accuracy of recovered features for different target users from Facebook dataset with maximum confidence. The fewer the unique features a user has (user 4 has the fewest), the better the worst case accuracy is for features recovered with maximum confidence.



(a) Target is User 1.

(b) Target is User 4.

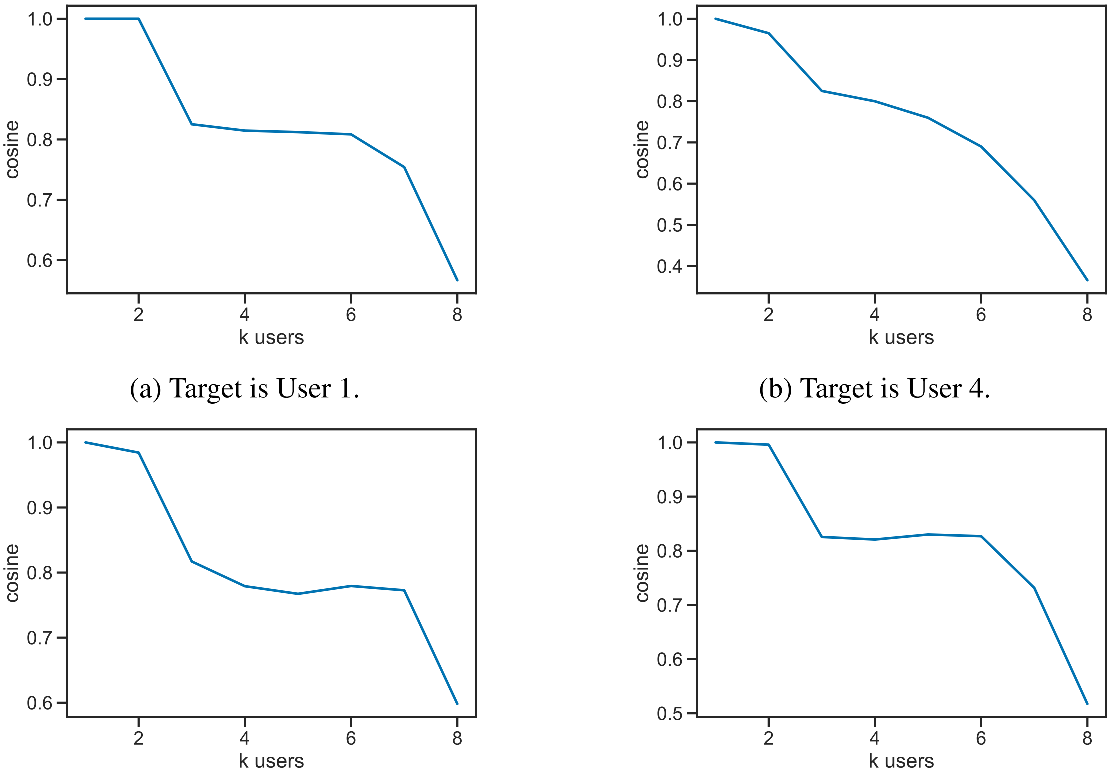
(c) Target is User 6.

(d) Target is User 8.

Figure 4.31: Percent of recovered features for different target users from Facebook dataset with maximum confidence.

4.6 Summary

This chapter proposes FedPacket, a framework for federated mobile packet classification, and evaluates its effectiveness and efficiency, using three real-world datasets and two different tasks (namely PII exposure and Ad request). First, we propose a reduced feature space (HTTP Keys), which limits the sensitive information shared by users. Then, we show that SVM with SGD performs similarly to decision trees used by state-of-the-art [115, 121, 120, 124], in terms of F1 score as well as interpretability. We also show that Federated achieves a significantly higher F1 score than Local and is comparable to Centralized models, and it does so within a few communication rounds and with minimal computation per user, which is important in the mobile environment. Finally, we demonstrate an attack by an honest-but-curious server that can infer features and brows-



(a) Target is User 1.

(b) Target is User 4.

(c) Target is User 6.

(d) Target is User 8.

Figure 4.32: Cosine similarity of true and recovered features for different target users from Facebook dataset. The larger the k the worse the cosine similarity is due to decreasing confidence levels which increases the distance between true and recovered features.

ing history, and demonstrate that simple existing add-on mechanisms can provide significant levels of protection.

Future Work. One can consider additional privacy protections on top of our FedPacket framework, beyond secure aggregation, *e.g.,* differential privacy (DP) [94, 65, 36], selecting a subset of gradient updates, compression of gradients or federated GANS [29]. Another promising direction for addressing both feature space explosion and privacy attacks is to train packet or URL embeddings specifically for this problem. Finally, our framework can be applied to tasks beyond PII/Ad prediction (*e.g.,* to detect tracking [123] or fingerprinting [148]), and beyond mobile devices (*e.g.,* for network traffic generated by different IoT devices).

Algorithm 3: Attack with Secure Aggregation in place

```
Given a target user $t$, $K$ total number of clients, $k$ number of clients who participate in a round, $n$
  feature size, encoded_value_certain value that represents a feature is present with certainty:
 Initialize: feature_array[K][n]=-1, $S_{feat} \leftarrow$ (set of features seen so far)
Phase 1:
 $S_{tuples} \leftarrow$ (set of all user k-tuples with $t$ being the first user in each tuple)
 for each k-tuple $p \in S_{tuples}$ do
     $S_{cur} \leftarrow$(union of features of users in $p$)
     if $S_{feat}$ is empty then
         for each $u$ in $p$ do
             for each feature $f \in S_{cur}$ do
                 feature_array[u][f] +=1
     else
      $S_{new} \leftarrow Scur - S_{feat}$
      for each feature $f \in S_{new}$ do
          feature_array[p[0]][$f$] = 0
          if len(p)==2 then
              feature_array[p[1]][$f$] = encoded_value_certain
          else
           for each $u$ in p[1:] do
               feature_array[u][f] +=1
      $S_{common} \leftarrow Scur \cap S_{feat}$
      for each feature $f \in S_{common}$ do
          for each $u$ in $p$ do
              feature_array[u][f] +=1
      $S_{prev} \leftarrow S_{feat} - Scur$
      for each feature $f \in S_{prev}$ do
          for each $u$ in $p$ do
              feature_array[u][f] = 0
          for each $u$ in all_users do
              if feature_array[u][f] $\geq 1$ && feature_array[u][f]!=encoded_value_certain then
                  feature_array[u][f] +=1
     $S_{feat}$.update($S_{cur}$)
Phase 2:
 $S_{feat2} \leftarrow$ (set of features seen in Phase 2)
 for $u$ in set($K$-$t$) do
     $S_{tuples2} \leftarrow$ (set of user k-tuples (without the target $t$) with $u$ being the first user in each
     tuple)
     for each k-tuple $p \in S_{tuples2}$ do
         repeat phase 1 with u as target
         $S_{feat2}$.update($S_{cur}$)
After Phase 2:
 for $f$ in $S_{feat}$ - $S_{feat2}$ do
     feature_array[t][$f$] = encoded_value_certain
 for $f$ in $S_{feat2}$ - $S_{feat}$ do
    feature_array[t][$f$] = 0
 for $u$ in range(K) do
     for $f$ in feature_array[u] do
         if feature_array[u][$f$] != 0 feature_array[u][$f$] != encoded_value_certain then
             feature_array[u][$f$] = feature_array[u][$f$]/rounds_user_u_seen
```

Chapter 5

Federated Signal Maps and Location Leakage

5.1 Overview

Mobile crowdsourcing is widely used to collect data from a large number of devices, which are useful on their own and/or used to train models for properties of interest. This data is often used to train models for a number of properties of interest, such as cellular/WiFi coverage, sentiment, occupancy, temperature, COVID analytics etc. Within this broader class of spatio-temporal models trained by mobile crowdsourced data [43], we focus on the representative and important case of cellular signal maps, described next.

Cellular operators rely on key performance indicators (a.k.a. KPIs) to understand the performance and coverage of their network, in their effort to provide the best user experience. These KPIs include wireless signal strength measurements (*e.g.,* LTE reference signal received power, a.k.a. RSRP, which is going to be the focus of this chapter), other performance metrics (*e.g.,* coverage, throughput, delay) as well as information associated with the measurement (*e.g.,* location, time, frequency band, device type etc.).

Cellular signal strength maps consist of KPIs in several locations. Traditionally, cellular operators collected such measurements by hiring dedicated vans (a.k.a. wardriving [141]) with special equipment, to drive through, measure and map the performance in a particular area of interest. However, in recent years they increasingly outsource the collection of signal maps to third parties [24]. Mobile analytics companies (*e.g.,* OpenSignal [100], Tutela [129]) crowdsource measurements directly from end-user devices, via standalone mobile apps, or measurement SDKs integrated into popular partnering apps, typically games, utilities or streaming apps. The upcoming dense deployment of small cells for 5G and smart city/IoT at metropolitan scales will only increase the need for accurate and comprehensive signal maps [62, 72]. Because signal strength measurements are expensive to obtain, they may not be available for all locations, times and other parameters of interest, thus the need for signal map prediction based on limited available spatio-temporal measurements.

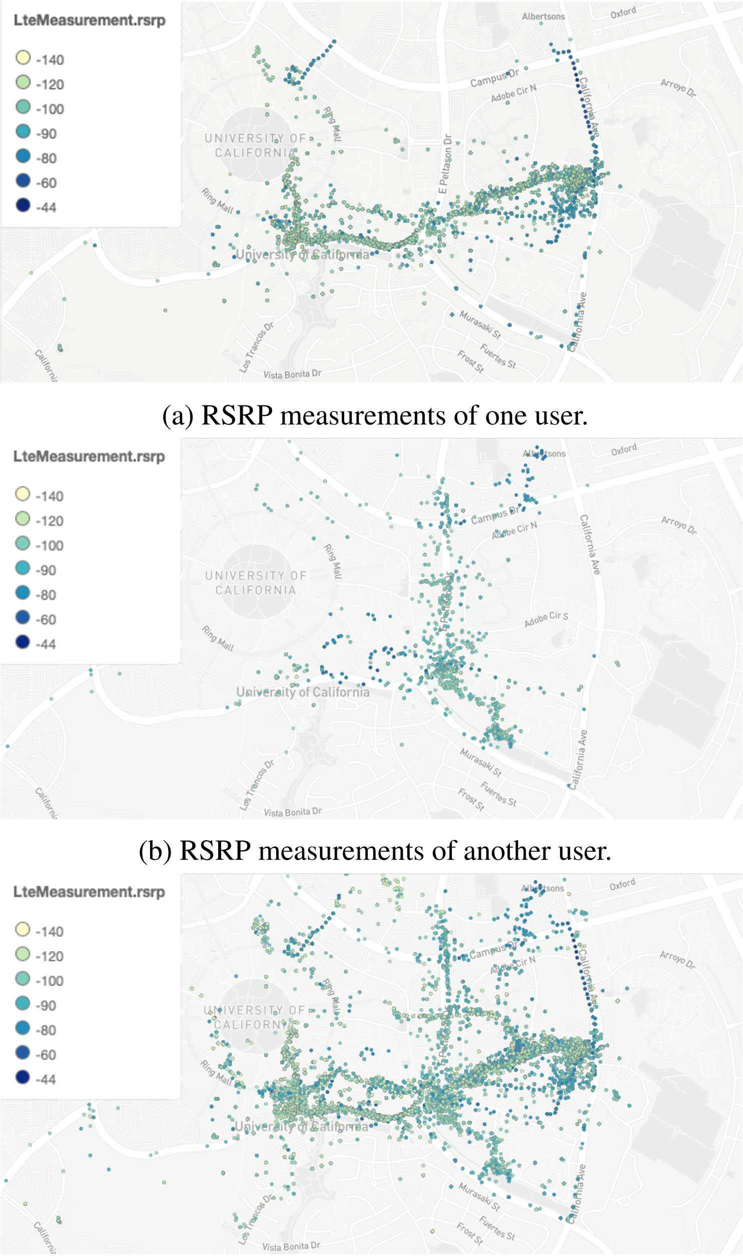
Signal map prediction is an active research area and includes: propagation models [144, 44], data-driven approaches [59, 46, 70], combinations thereof [104], and increasingly sophisticated machine learning models for RSRP [108, 55, 24, 79] and throughput [134, 145]. However, all of these prediction techniques are consider a *centralized setting*: mobile devices upload their measurements to a server, which then trains a global model to predict cellular performance (typically RSRP) based on location (and potentially time and other features). Clearly, this introduces privacy risks for the participating users: sharing location data alone can reveal the user's home and work and other frequently visited locations, as well as occasional visits to places that reveal their medical conditions, political beliefs and and other sensitive information [19].

Fig. 5.1 depicts an example of UCI Campus LTE Dataset which was introduced in Chapter 2. We see the locations where measurements of signal strength (RSRP) were collected by two (out of seven) different volunteers as they move around the campus. The measurements are then uploaded to a server, which can then create a signal map for the campus. In particular, the server can then merge the datasets from different users and store them; and/or may train a global model for predicting signal strength based on location and other features. However, this utility comes at the expense of privacy: as evident in Fig. 5.1, frequently visited locations may reveal user's home, work, other important locations, as well as their mobility pattern. In this particular example, the trajectories of the two users are also sufficiently different from each other, and can be used to distinguish between them, even if pseudo-ids are used by the users.

In this chapter, we make the following contributions: (1) we formulate the *signal strength prediction* problem within an *online federated learning* framework and (2) we consider, for the first time, *location privacy attacks* launched by an honest-but-curious server.

(1) W.r.t. the prediction problem itself: we consider the simplest and cleanest version that lies at the core of this problem: we train a DNN model to predict signal strength (RSRP) based on GPS location (latitude, longitude), while local training data arrive in an online fashion. The problem lends itself naturally to federated learning: training data are collected by the mobile devices, which want to collaborate without uploading sensitive location information. Federated learning enables mobiles to do exactly that by exchanging model parameters with the server but keeping their training data from the server [93]. The problem further lends itself to online federated learning because the training data are collected over time as the users move around, thus in an online fashion [42, 82].

(2) W.r.t. the location privacy attacks: Since gradient updates (explicitly sent in FedSGD or can be computed from model updates in FedAvg) are sent from users to the server in every round, federated learning lends itself naturally to inference attacks from gradients. We adapt the Deep-Leakage-from-Gradients (DLG) attack [147, 64], originally developed (and exclusively used so far) to reconstruct training images [147, 64] and text used for DNN classifiers [147]. A key observation, that we confirm both empirically and analytically, is that an honest-but-curious server who observes gradient updates from individual target users, can reconstruct the *average* location of points in a single batch. This is different from state-of-the-art gradients-based attacks where the attacker aimed to reconstruct N images from a



(a) RSRP measurements of one user.

(b) RSRP measurements of another user.

(c) RSRP measurements from all users.

Figure 5.1: Locations where signal strength (RSRP) measurements are collected by different mobile users on the UCI Campus LTE Dataset. Users have different trajectories; see Fig. a) and b). Fig. c) shows the measurements from all users merged, which motivates crowdsourcing-based training for signal maps.

gradient obtained on N images. Over multiple rounds of federated learning, this allows the reconstruction of the target(s)' mobility pattern at a coarse granularity of a batch. We show that averaging of gradients inherent in FedAvg [93] provides a moderate level of protection against DLG, while simultaneously improving utility; to that end, we systematically evaluate the effect of multiple federate learning parameters E, B, R, η on the convergence and success of the attack. We also propose a Diverse Batch algorithm that mobile devices can apply locally to curate their batches, so as to further protect their location privacy, without hurting utility, and while enabling data minimization. Finally, we show that the effect of multiple users participating in federated learning, w.r.t. the success of the DLG attack, depends on the similarity of trajectories of those users.

Throughout this chapter, we use two real-world datasets including: a small but dense UCI Campus LTE Dataset [25]; and a larger but sparser London metropolitan-area signal maps publicly available from the Radiocells Dataset [16]. Both datasets have been introduced in Chapter 2. It is also worth noting that we purposely do not consider additional privacy-preserving defense techniques such as Differential Privacy (DP) or Secure Aggregation (SA), in order to show that these (often expensive) mechanisms are not necessary in this context.

5.2 DLG Attack to Infer Location

Signal Maps. There has been significant interest in signal map prediction techniques based on a limited number of spatio-temporal cellular measurements. These include propagation models [144, 44] as well as data-driven approaches [59, 46, 70] and combinations thereof [104]. Increasingly sophisticated machine learning models are being developed to capture various spatial, temporal and other characteristics of signal strength [108, 55, 24] and throughput [134, 145]. Prior work has focused exclusively on minimizing the mean squared error (MSE) in predictions of raw signal strength itself [24], but the problem has been considered so far only in a centralized, not federated, way. In this work, we aim to train machine learning models to predict the signal strength of a user's device based on GPS locations via federated learning [93]. To the best of our knowledge, no prior work considered signal maps prediction in addition to the case of streaming (online) data in FL and the corresponding privacy leakage via DLG attacks.

FL & Location Privacy. Numerous works have evaluated privacy in location (or trajectory) datasets, *e.g.,* [73, 51, 105, 52], where the utility of the dataset lies in the location itself. In contrast and as pointed out in [43], the utility in mobile crowdsourcing does not lie in the location itself, but in the measurement associated with that location. In our application scenario, the measurement is the signal strength measurement, while location is only a feature in a prediction model for signal strength. In this work we focus on location privacy in crowdsourcing systems, similarly to [43, 105] but we consider the federated learning as a one of the defenses and evaluate the privacy leakage in different FL setups showing that the aggregation mechanisms of FL provide enough protection without the need of additional add-ons like DP. Moreover, we consider the case of online data. Few works [80] considered the case of online location data, although not in distributed way and with a significant different goal: next location prediction. In terms of online FL, there are few works [47, 90, 50] and none of them considered privacy leakage in such setup, but instead they focused on improving convergence in the FedAvg algorithm and tackle the bottleneck of stragglers due to device heterogeneity.

Data reconstruction based on gradients. It has been shown that exchanging gradients [147, 146] in FedSGD or model parameters [64] in FedAvg with an honest-but-curious server can enable reconstruction of the local training data. Such attacks have been demonstrated in the past in other contexts. First, *deep leakage from gradients*, (DLG) [147], reconstructed training data (images and text) and their corresponding labels, from observing a single gradient during training of DNN image classifiers, without the need for additional models (e.g., GANS [71]) or side information. The DLG attacker observes a single gradient, which is the case in FedSGD but in not FedAvg; it performs local training on at least one pass of local data before sending model updates to the server, and reconstructs either a single or several datapoints, from a single or a batch of initial dummy points, respectively. The idea of the DLG attacks is to minimize the distance of the gradient observed and the gradient of some randomly initialized data (which we call reconstructed data) and based on this optimization step, update the reconstructed data. Second, the *improved leakage from gradients* (iDLG) [146] optimized the reconstruction of the target label, thus further improving the convergence speed of the attack although it is only applicable to classification. The closest to our work is *inverting gradients in federated learning* [64] that modified the DLG attack to use a cosine-based distance instead of the Euclidean. Their goal was not to perform gradient matching finding for data reconstruction, but to find datapoints that lead to a similar change in model prediction. They also evaluated for the first time DLG against FedAvg and the impact of averaged gradients due to local epochs on the attack. Overall, prior work on DLG attacks in FL has focused exclusively, so far, on image or text data.

This Work in Perspective. To the best of our knowledge, no prior work considered signal maps prediction in addition to the case of streaming (non-static) data in FL and the corresponding location leakage via DLG attacks. We adapt the DLG attack [64] with the goal to reconstruct the average location of a data batch based on gradients of model parameters in FL, where each data batch is obtained via a time interval. We assume there are no add-ons like Differential Privacy (DP) or Secure Aggregation (SA), and we consider instead the vanilla FL. This is realistic since both DP and SA are expensive to implement so there is economic incentive for companies to use vanilla FL [93]. Moreover, the first DLG attack paper [147] discussed potential defenses including tuning training parameters such as mini-batch size. Thus, we investigate this approach here; after extensive evaluation of various FL parameters, we show how the averaging of gradients in FL provides some natural protection against location leakage via DLG attacks. Finally, we propose a clustering algorithm to limit further the effectiveness of the attack without hurting utility.

5.3 Problem Setup

Signal Maps Prediction. In this work, we are interested in training a DNN model to predict signal strength in LTE; the Reference Signal Received Power (RSRP) from location features. In particular, we denote the DNN service model as $y_i = F(x_i, w)$, which predicts RSRP value y_i based on the input feature vector $x_i = [x_{i,1}, x_{i,2}]^T$, where i denotes the i-th datapoint in each input vector x, which includes (longitude, latitude) coordinates. To evaluate the performance of the model we follow prior work and choose the Root Mean Squared Error (RMSE) as our utility metric. In order to maximize utility, we tune the DNN architecture (how deep, how wide, activations, learning rate) and hyperparameters via Hyperband tuner [81]. We conduct tuning for each cell tower, since each model is per cell tower. In practice, before FL starts, the server and the clients need to agree to a common model and in this case the server will perform tuning with a public dataset for the corresponding cell tower. This is realistic, since the operator will indeed have access to smaller datasets per cell tower which are collected either directly by the operator or via users who are willing to share their data directly with the server. In the results section, we use a DNN with two hidden layers with 224 and 640 units and ReLU and sigmoid activation functions, respectively. We also add a dropout layer after each hidden layer with 5% of units being dropped randomly to prevent overfitting.

The Online FL Framework. Federated learning [93], as already discussed in Chapter 2, allows users to keep their actual location and associated signal strength measurements on the device and only exchange model parameter updates with the server, while still collaborating to train a global model. All users and the server agree on the global model to train (in our problem we consider DNNs that predict signal strength based on location), and they operate in rounds to train it. FL has the following parameters: R number of FL rounds, B local mini-batch size (B=inf means the whole dataset is treated as a single batch), E number of local epochs, C fraction of users that participate in an FL round. These parameters control the client computation and communication between the server and clients. In every round t (out of total R), the server initializes the global parameters and asks for a local update from available users. During the local update, the user trains a local model on a single or multiple mini-batches, depending on the mini-batch size B, performs one or multiple passes on the data depending on the local epochs E, and eventually sends its new model parameters to the server. The server averages them with updates from other users, updates

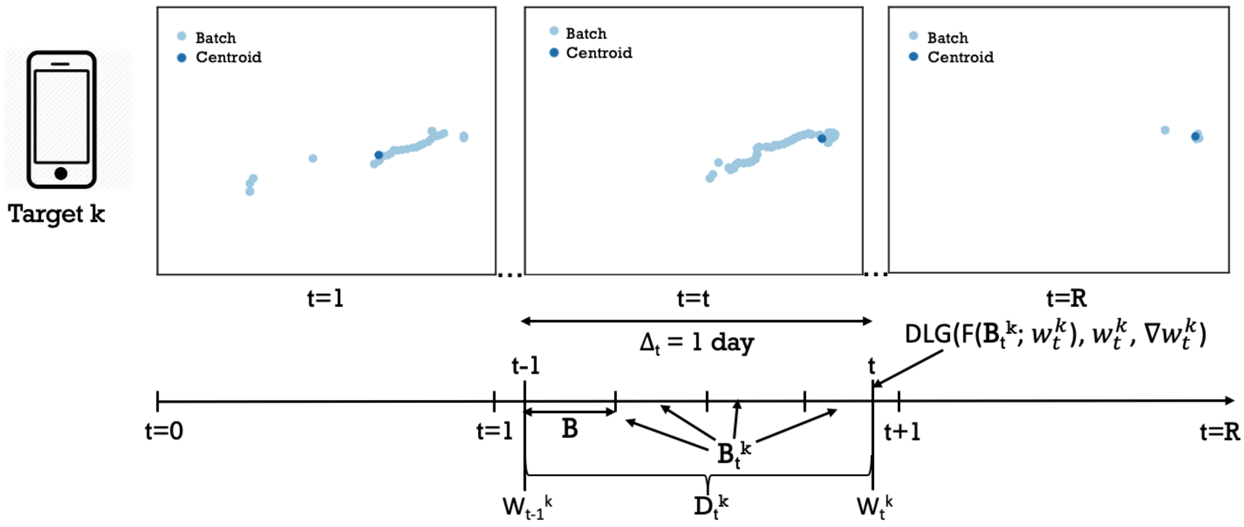


Figure 5.2: Example for 1-day rounds ($\Delta_t =$ day) in online FL that results into R total rounds. The client/target k collects D_t^k data batch (light blue) in each t round, which is used for local training to obtain the updated weights w_t^k and share them with the server. The D_t^k is split into a list B_t^k depending on the mini-batch size B. The server launches a DLG attack in each t round and aims to reconstruct those average locations (dark blue) across rounds.

the global parameters and initiates another round, until convergence.

In contrast to standard FL that was presented in Chapter 2, we consider here specifically the case of **online batch FL**, where the local data per user is not static, but it becomes available over time as the user moves around. We denote D_t^k, as the local data batch of a client k in round t, which can be partitioned further into mini-batches of size B (which will be referred as B_t^k). We would like to note that the granularity of the time interval corresponds to the number of FL rounds R, due to online data. We summarize the main parameters in Table 5.1. In every round t, only the newly acquired data D_t^k of a client k is used used for local training and the data from previous rounds (e.g., D_{t-1}^k) is discarded. An example is pictured in Fig. 5.2 for 1-day rounds; each day shows a user's trajectory. An overview of the algorithm is shown in Algorithm 4, which is also called Federated Averaging (FedAvg). A similar scheme, Federated SGD (FedSGD), has a significant difference from FedAvg: each device performs a single gradient descent step on their local data. FedSGD and FedAvg are equivalent when $E = 1, B = \infty, C = 1$. The server also launches a Deep Leakage from Gradients (DLG) attack in each round t for a target client, as described below,

Parameter	Description
x	Input features with lat, lon coordinates
y	Prediction label for RSRP
ℓ	MSE loss for RSRP prediction
η	Learning rate
t	FL round based on time interval, up to R in total
Δ_t	Interval to split data into rounds of granularity t
w_t	Global model weights at round t
w_t^k	Local model weights at round t from client k
D_t^k	Data batch of client k in round t
B	Mini-batch size; if $B = \infty$ then one mini-batch
B_t^k	List of mini-batches for a client k at round t
E	Number of local epochs in FedAvg
$\mathbb{D}$	Cosine loss in DLG
∇w_t^{target}	Gradient of target's model weights at round t
x_{DLG}	Final reconstructed location via DLG
$\bar{x}_t$	Average/centroid location of data D_t^k
$\bar{g}$	Average gradient obtained on a mini-batch
ϵ	DBSCAN parameter that controls total clusters

Table 5.1: Summary of main parameters.

aiming to reconstruct the average or centroid location (*i.e.,* the dark blue locations in Fig. 5.2).

Threat Model. We assume an honest-but-curious server who w.l.o.g. wants to infer the locations of a particular target user, or all users for that matter. Different notions of location can be inferred, *e.g.,* the trajectory at different spatio-temporal granularities, important locations, presence in points-of-interest. We purposely assume no MPC or DP add-ons, *i.e.,* the server sees the model updates coming from each individual user in every round. This is the strongest possible honest-but-curious adversary in FL: it stores updates per user and compare across successive rounds t, which allows to also calculate the gradient of model parameters from a target user, as shown in Algorithm 4, in order to launch a DLG attack. This is another key difference from the standard FL algorithm presented in Chapter 2.

DLG Attack. Algorithm 5 shows the details of the DLG attack, which is incorporated in the FedAvg algorithm 4 where the server targets a specific user and launches the attack for each round

Algorithm 4: Online FedAvg with DLG Attack.

```
Given $K$ clients (indexed by $k$); $B$ local mini-batch size; $E$ number of local epochs; $R$ number of
  global rounds based on interval; $C$ fraction of clients; $n_t^k$ is the training data size of $D_t^k$ of client $k$
  at round $t$; $n_t$ is the total data size from all users at round t and $\eta$ is learning rate; *target* is the
  client that the server aims to reconstruct their local data.
**Server executes:**
  Initialize $w_0$
  **for** *each round t = 1,2, ... $R$* **do**
      $m \leftarrow max(C \cdot K, 1)$
      $S_t \leftarrow$ (random set of $m$ clients)
      **for** *each client $k \in S_t$ in parallel* **do**
          $w_{t+1}^k \leftarrow ClientUpdate(k, w_t, t, B)$
          **if** $k$==*target* **then**
              $\nabla w_{t+1}^{target} \leftarrow w_{t+1}^k - w_t^k$
              DLG($F(x; w_{t+1}^k)$, $w_{t+1}$, $\nabla w_t^{target}$)
      $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_t^k}{n_t} w_{t+1}^k$

**ClientUpdate**($k, w, t, B$)**:**
  $B_t^k \leftarrow$ (split of local data batch $D_t^k$ at round $t$ into mini-batches of size $B$)
  **for** *each local epoch i from 1 to $E$* **do**
      **for** *mini-batch $b \in B_t^k$* **do**
          $w \leftarrow w - \eta \nabla \ell(w; b)$
  **return** $w$ *to server*
```

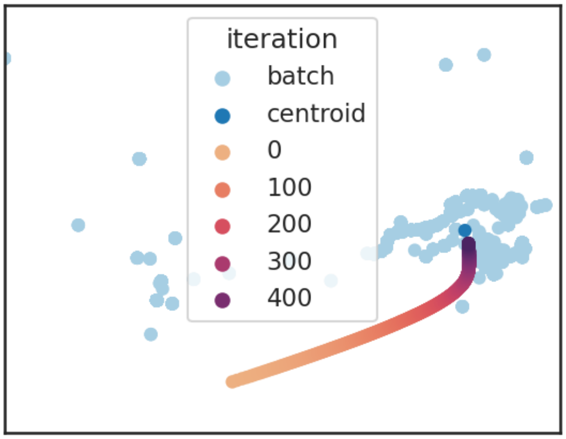


Figure 5.3: DLG attack based on the gradient obtained on a batch of ground truth locations (light blue) that reconstructs their average/centroid. The DLG attack starts with a randomly initialized location (yellow) and it gets closer to the centroid with more iterations (darker color indicated progression in iterations) by minimizing the cosine distance of the observed gradient and the gradient obtained on the current reconstructed point. The distance between the final reconstructed (dark purple) and the centroid (dark blue) is 20 meters.

Algorithm 5: DLG Attack on Latest Batch of Data.

```
Input: $F(x; w_t)$: DNN model at round $t$; $w_t$: model weights, $\nabla w_t$:model gradients, after target
  trains on a data batch of size B at round t, learning rate $\eta$ for DLG optimizer; $m$: max DLG
  iterations; $a$: regularization term for cosine DLG loss.
Output: reconstructed training data (x, y) at round t
 Initialize $x'_0 \leftarrow \mathcal{N}(0,1)$, $y'_0 \leftarrow \bar{y}$                              // mean RSRP
 for $i \leftarrow$ 0,1, ... $m$ do
     $\nabla w'_i \leftarrow \partial\ell(F(x'_i, w_t), y'_i)/\partial w_t$
     $\mathbb{D}_i \leftarrow 1 - \frac{\nabla w \cdot \nabla w'_i}{\|\nabla w\|\|\nabla w'_i\|} + \alpha$                              // cosine loss
     $x'_{i+1} \leftarrow x'_i - \eta\nabla_{x'_i}\mathbb{D}_i$, $y'_{i+1} \leftarrow y'_i - \eta\nabla_{y'_i}\mathbb{D}_i$
 return $x_{DLG} \leftarrow x'_{m+1}$
```

t the *target* client participates. An example is shown in Fig. 5.3, the ground truth data (light blue) contains visited locations of a user during a week that corresponds to a D^k_t. The DLG attack is an iterative algorithm which works as follows: 1) choose a target user and initialize a dummy location x'_0 randomly (in yellow), 2) obtain the gradient with respect to dummy location, $\nabla W'_i$, which we call dummy gradient, 3) update the dummy location towards the direction that minimizes of the cosine-based loss between the two gradients, similarly to [64], of the dummy gradient and the true gradient that was given as input. After $m = 400$ iterations, the reconstructed location is only 20 meters away from the average/centroid location. Since we consider online FL, the attacker aims reconstruct a single location in each round t that is close to the ground truth average of the batch D^k_t. Due to the online nature of data and location characteristics, there is no need to reconstruct all N locations, since the average location still reveals user's whereabouts. The attacker reconstructs both the location (lat, lon) and the target RSRP value, as we cannot use the analytical reconstruction of the label that was proposed in [146] since we focus on a regression problem instead of classification. We cannot assume the label is known by the server, since the attacker does not have access to this information and in fact, the signal strength values can actually reveal user's location. We evaluate different initializations for the location, but for the prediction label (RSRP) we initialize with the mean RSRP from the training data. This is realistic, as the attacker can have access to this information via public data or by collecting some measurements around each cell tower with its own devices. For each DLG attack we set the maximum number of iterations to 400,000, and add an early stopping condition: if the reconstructed location does not differ for 10 subsequent rounds, then we assume the attack has converged and return that reconstructed location.

5.3.1 DLG Convergence to the Average Location

We already showed in practice that the DLG attack on a gradient based on a batch of locations, B_t^k, of a user k at round t converges near the average of B_t^k. How close is the final reconstructed location to the average? This can be shown by the following theorem. Assume that the service model is a biased fully-connected neural network with L layers defined as $y_i = f(x_i)$, which predicts RSRP value y_i based on the input feature vector $x_i = [x_{i,1}, ..., x_{i,M}]^T$, where x_i and y_i represents the i-th feature vector and the corresponding RSRP value respectively, and M is the total number of features in each input vector (in this work $M = 2$). Then, the l-th layer of the service model can be represented as follows:

$$y_i^l = \sigma(w^l x_i^l + b^l),$$

where x_i^l and y_i^l is the input and output of l-th layer given the model input x_i.

THEOREM 5.1. [1] *Suppose that a data mini-batch of size B $\{(x_i, y_i)\}_{i=1}^{i=B}$ is used to update the service model $y_i = f(x_i)$ during a gradient descent step. Then, the distance between each reconstructed feature by DLG attacker and the mean value of this feature in the data batch B can be bounded by the following equation:*

$$|x_{DLG,j} - \bar{x}_{i,j}| \leq \frac{1}{2B} \sum_{i=1}^{i=B} \big((\frac{g_i}{\bar{g}} - 1)^2 + (x_{i,j} - \bar{x}_{i,j})^2 \big).$$

where $x_{DLG,j}$ is the j-th reconstructed feature, $\bar{x}_{i,j} = \frac{1}{B} \sum_{i=1}^{i=B_t^k} x_{i,j}$, $\bar{g} = \frac{1}{B} \sum_{i=1}^{i=B} g_i$, $g_i =$

[1] This theorem was first observed empirically by the author, and then proved analytically by her collaborator, Jiang Zhang.

$\frac{\partial L(f(x_i),y_i)}{\partial b_k^1}$*, and* b_k^1 *is any one of the* k*-th feature in* b^1 *satisfying* $b_k^1 \neq 0$.

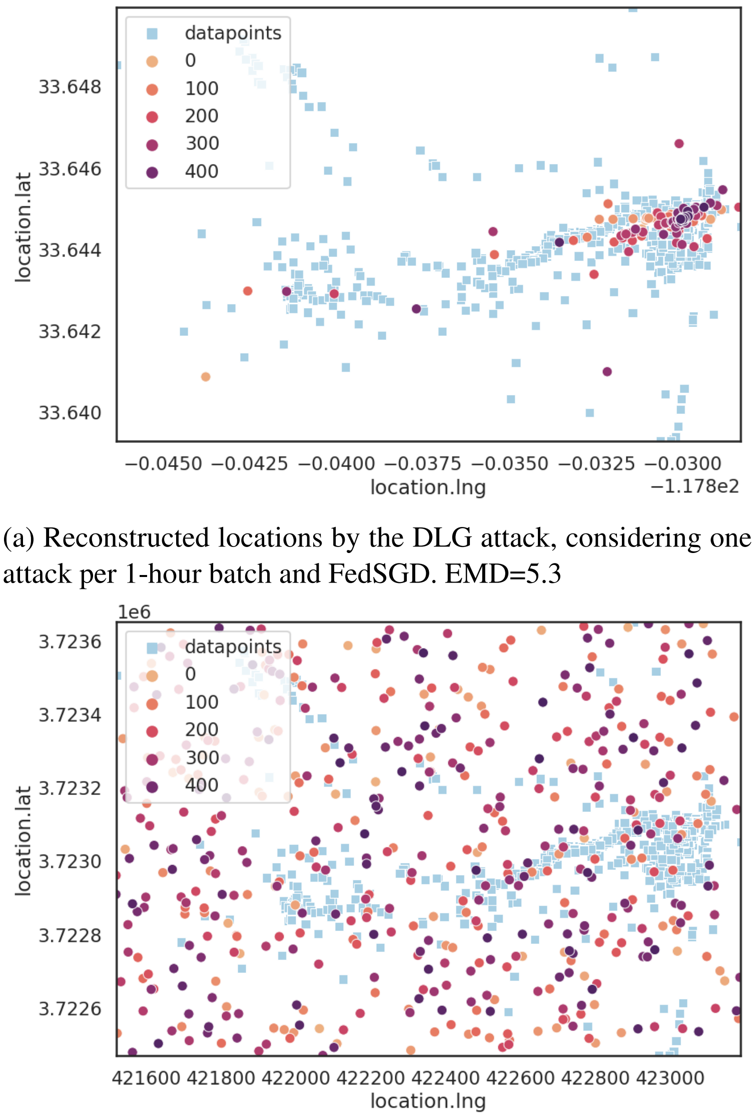
We will be using the bound in Theorem 5.1 throughout the evaluation of this work, see Results section, and we defer to the appendix for the full proof in [33]. The bound holds on all B values but when B increases the bound becomes tighter.

5.3.2 Assessing the Success of the Attack

When applied to each batch of local data B_t^k of a user k at round t, the DLG attacker essentially infers one location x_{DLG} per batch B_t^k. As the user moves around over time t, the user essentially infer a coarse (averaged) version of the trajectory of the user. Fig. 5.3.2(a) depicts that effect. In order to assess the success of the attack, we need metrics that capture how similar or dissimilar are the inferred locations from the real ones, but also how accurate the attack is.

Attack Divergence. The reconstructed locations will not always converge to the mean, as convergence depends on the location variance, initialization, whether the parameters of the DLG optimizer are tuned properly and the existence of averaging mechanisms that make the attack more challenging. We evaluate how expensive and accurate is the DLG attack in terms of average execution time and average DLG iterations per reconstructed location. Another aspect that measures the convergence of the attack is to count how many attacks have diverged, aka they are outside the attacker's defined geographical area. We refer to these locations as "diverged" and we discard them before obtaining the privacy, utility metrics.

Earth Movers Distance (EMD) [39, 60]. EMD takes into account the spatial correlations and returns the minimum cost/effort required to convert one probability distribution to another, which makes it a good privacy metric for our problem. A classic interpretation of EMD is that it treats the two probability distributions as two different ways to pile up an amount of dirt (earth) over a region D and thus, the EMD is the minimum cost required to turn one pile into the other. The cost is



(a) Reconstructed locations by the DLG attack, considering one attack per 1-hour batch and FedSGD. EMD=5.3

(b) One randomly generated location per hour (489 locations in total). EMD = 21.33 for 5 realizations (or 0.47 if normalized by the maximum distance: approx. 2km diagonal).

Figure 5.4: We consider a target user and its real locations on campus, 489 in total, depicted in light blue. The over-sampled area on the right corresponds to home location of that user. The other area on the left, corresponds to his work on campus. We see that a DLG attacker processing updates from 1h intervals can successfully reconstruct the important locations of the user: the difference between the distribution of real and the inferred locations is EMD=5.2. To put that in context, if one would randomly guess the same number of locations, the EMD would be 21.33.

defined by the amount of dirt moved times the ground distance by which it was moved. The sliced version is based on Monte Carlo approximations based on N number of projections. It is more computationally efficient than the exact EMD calculation and it is suitable for 2d distributions. It is defined as follows: $SWD_2(\mu, v) = \mathbb{E}[W_2^2(\theta\#\mu, \theta\#v)]^{\frac{1}{2}}$, where $\theta\#\mu$ stands for the pushwards of the projection $\mathbb{R}^d \in X \mapsto \langle\theta, X\rangle$.

EMD has been used in prior location privacy works, and specifically as a measure for t-closeness [83] and l-diversity [91], but not as a metric of location leakage. Another work [61] used EMD as their cost function, not for privacy but for determining if two sets of GPS locations were generated by the same individual. We use Euclidean distance as our distance function when calculating EMD on GPS coordinates that were converted to UTM. When calculating the EMD metric, we bound all reconstructed locations to a geographical area defined by the min, max longitude and latitude values from our dataset and convert them to UTM so that the distance being calculated is in meters. In case some reconstructed locations are outside these boundaries we simply discard them when computing the privacy metrics but we report the fraction of diverged locations/attacks in each scenario. In practice, the attacker will relaunch the DLG attack with a different initialization hoping it will lead to a converged reconstructed location.

Our main privacy metric is EMD; low EMD values correspond to more privacy risk, and EMD=0 indicates the two distributions are identical (maximum privacy risk). To make the EMD metric more intuitive in terms of privacy, we assume the attacker samples locations at random and uniformly, that is, he randomly guesses locations in the defined geographical area. Next, we calculate the EMD between these random locations and the ground truth locations. We call this baseline EMD_random and we perform this procedure five times in order to report the average EMD. An example is shown in Fig. 5.4b and the corresponding EMD_random is 21.33. This is how much privacy we can gain when the attacker randomly guesses. In contrast, launching the DLG attack with 1-hour rounds results to EMD=5.3, which indicates more privacy leakage. Since the interval is very fine in this case, the reconstructed locations reveal part of the user's trajectory.

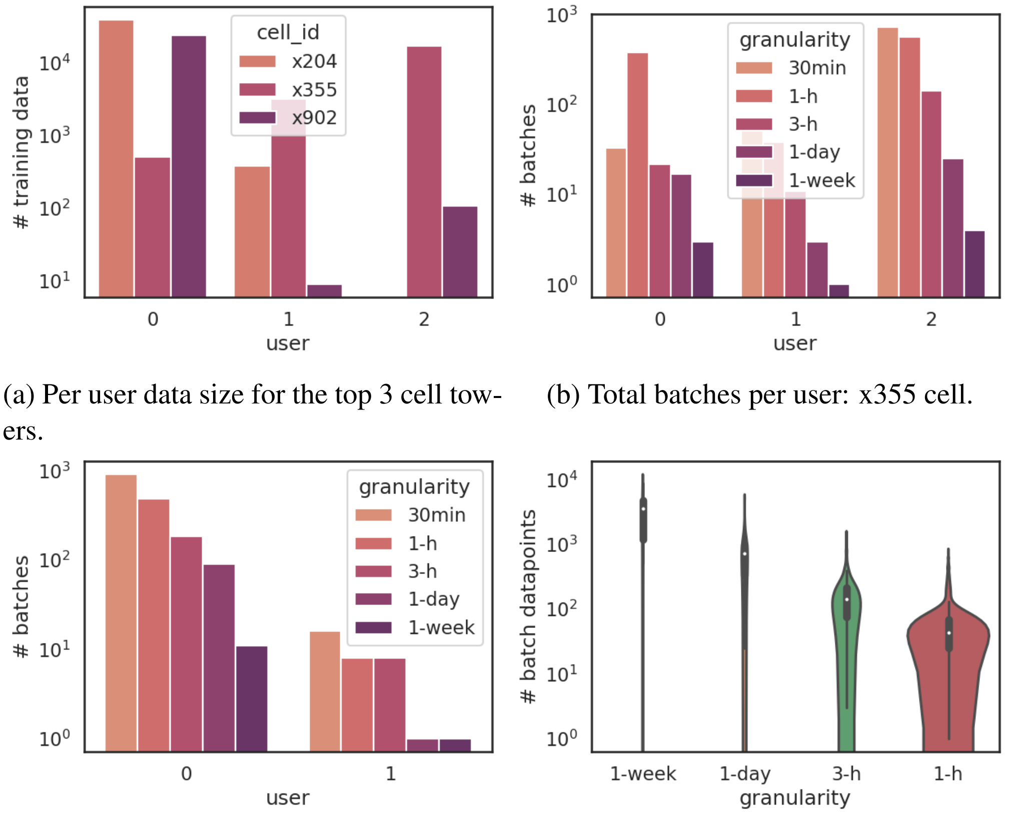
Visualization. Visualizing the reconstructed locations on top of the true user locations. This metric more intuitive and shows directly the privacy leakage in each scenario.

Distance from the Centroid. We report this metric to show how accurate the DLG attack is in each scenario. That is, how far away (in meters) is the corresponding reconstructed location in round t from the average locations of B_t^k that the attacker was aiming to reconstruct via the observed gradient. We also refer to this metric as: $||x_{DLG} - \bar{x}_t||$, where $\bar{x}_t$ is the centroid of the B_t^k and x_{DLG} is the reconstructed location at round t.

Jensen-Shannon Distance (JSD). We also considered the JSD, for each pair of reconstructed and ground truth heatmaps after normalizing and flatting each matrix to obtain the probability vectors P and Q. Although this is a well known metric for comparing distributions, it does not consider the ground distance but only the probability mass, unlike EMD.

5.4 Datasets

UCI Campus LTE dataset [25]. This dataset was introduced in Chapter 2 and we analyze it and use it as follows here. The distribution of the number of measurements for the top three cell towers with the most measurements is shown in Fig. 5.5a. Each dataset is preprocessed by removing the mean of each feature and scaling for unit variance. A 70-30 random split was chosen to create a training and test set. The pseudo-IDs were also used as proxies for user splits under federated learning. We also split the data into batches based on time intervals on varying granularities: 30 minutes, 1-3 hours or 1 week. Fig. 5.5c and Fig. 5.5b shows the resulting batches per user for the x204 and x355 cell towers respectively. For coarser granularities we obtain fewer batches but they contain significantly more datapoints, *e.g.,* for cell x204 and user 0, the average batch size for 1-week batches is 3492, for 1-day is 817, for 3-hour is 205 and for 1-hour batches is 79 datapoints. We would like to remind the reader that the granularity of the time interval corresponds to the number of FL rounds R (or global updates), when we train in online manner for every new “batch” of data that the user acquires based on the interval. For instance, for 1-week interval, the total FL rounds are 11 since there are 11 batches in total where each batch contains 1 week worth of



(a) Per user data size for the top 3 cell towers.

(b) Total batches per user: x355 cell.

(c) Total batches per user: x204 cell.

(d) Total datapoints per batch per interval: 204 cell, user 0.

Figure 5.5: Distribution statistics for Campus dataset and the top 3 cell towers.The data is split into batches based on time granularity; the finer the granularity the more batches (and more FL rounds/updates in Online Federated Learning) are obtained but each batch has few datapoints in contrast to coarser granularities which result to few but large batches/updates.

datapoints.

Radiocells [16]. We introduced this dataset in Chapter 2 and here we use it to evaluate the case of multiple users participating in FedAvg. The UCI Campus LTE Dataset, although it contains pseudo user ids, it has limited number of users. It does not contain user ids, but it does contain multiple upload files where each upload file corresponds to a single device. We focus on data from 2017 and the area of London, UK which had the most measurements and approx. 3500 upload files. Since each upload file is limited in terms of number of measurements and/or time duration, we merge multiple upload files into a single user by considering the distance of start and end locations among multiple upload files. For instance, if the start locations do not differ more than a threshold

T between two upload files and their end locations do not differ more than T, then we merge the two upload files and assume they belong to the same device/user. The motivation behind this is that the user starts logging their data from the same place (their home/work) and end logging at the same location (home/work). By adjusting T to 1 mile, we obtain 933 pseudo users, where the potential users have several weeks worth of data, so we choose per-week intervals. We also filter out the data to get only LTE measurements and we remove measurements with RSRP outside the range [-140, -44] dBm, as well as we replace collisions in the label (same location, timestamp but different RSRP value) with the average RSRP value. Fig. 5.6 shows the violin plot for per batch datapoints for each user for 1-week intervals in the selected cell tower x455 with the most measurements. For all potential target users, the average number of measurements per batch/round is around 10. The corresponding average in the Campus data was more than 1,000 measurements for user 0 and 380 measurements for user 1 when considering 1-week intervals for top cell x204.

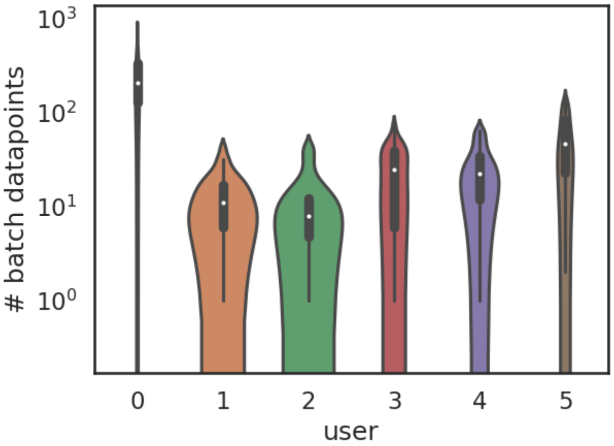
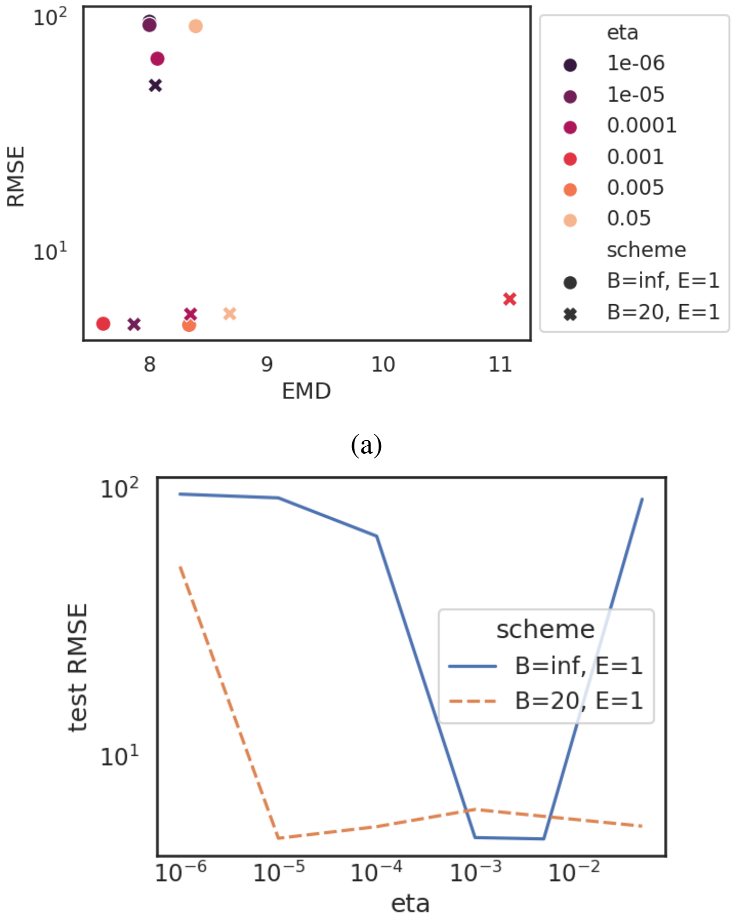


Figure 5.6: Total datapoints in each batch per user for 1-week intervals in Radiocells.

5.5 Results

In this section, we use the UCI Campus LTE Dataset and the Radiocells Dataset to evaluate the performance of online federated learning under privacy attacks

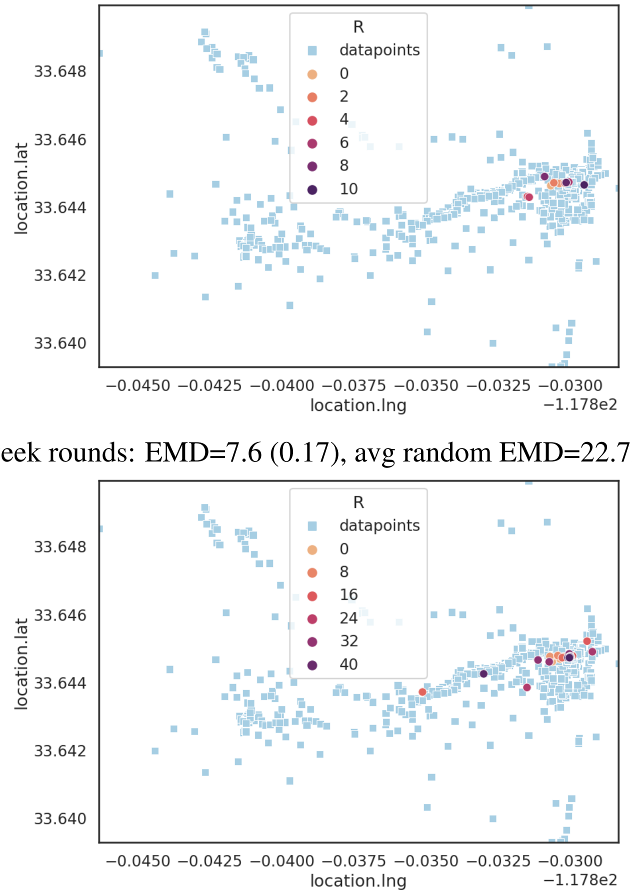


(a)

(b)

Figure 5.7: **Tuning learning rates.** We use the 1-week rounds to tune η for both B=inf and B=20 and show how utility (RMSE) and privacy (EMD) is affected. We choose $\eta = 0.001$ as our default learning rate (unless stated otherwise) and $\eta = 10^{-5}$ for B=20, since both minimize RMSE and EMD.

Learning rate η. The first parameter we need to tune is the learning rate η since SGD is sensitive to this parameter and utility is affected significantly. We performed a grid search with $\eta = 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 0.001, 0.1$ for two scenarios: FedSGD and FedAvg with E=1, B=20. To minimize RMSE, we choose $\eta = 0.001$ as our default learning rate. In case of minibatches, we observe a lower $\eta = 10^{-5}$ can slightly lower RMSE. In the following experiments, we either consider the default $\eta = 0.001$ or we lower $\eta = 10^{-5}$ in case of minibatches in order to minimize the RMSE and then measure the privacy leakage.



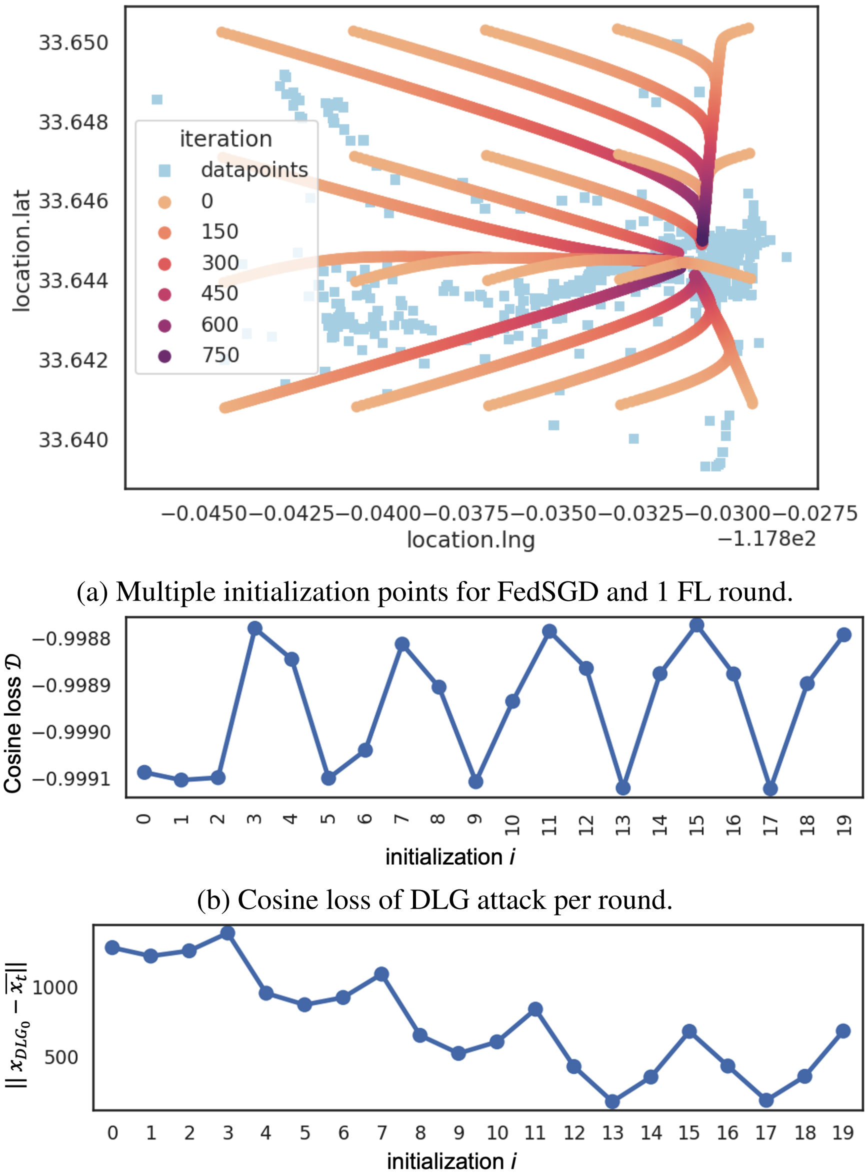
(a) 1-week rounds: EMD=7.6 (0.17), avg random EMD=22.72 (0.5).

(b) 24-hour rounds: EMD=6.4 (0.14), avg random EMD=22 (0.49).

Figure 5.8: The corresponding recovered locations in the case of strongest attack with eta=0.001 for various time granularity of FL rounds. The light blue square points are the ground truth points and the circle points are the reconstructed points for each round; the darker color represents the later rounds. RMSE is 4.93, 4.91, 5.16 for 1w, 24h, 1h respectively. Reconstruction with 1-hour rounds (Fig.5.4a) reveals user trajectories. The coarser rounds (24-h, 1-week) still reveal the frequent locations of the target, *e.g.,* their home/work locations.

5.5.1 Location Leakage with FedSGD

Strongest Attack. We consider here the case of FedSGD, which introduces minimal averaging of gradients due to $B = \infty$, E=1. That is, in each FL round the target user performs a single SGD step on their local data and sends the local model parameters to the server. We present here the strongest attack for various time granularities of rounds with $\eta = 0.001$. Fig. 5.8 shows the corresponding true locations and the reconstructed locations via DLG for 1-week and 1-day intervals. We observe that the finer the interval, *e.g.,* 1-hour as shown in Fig. 5.4a, of the rounds



(a) Multiple initialization points for FedSGD and 1 FL round.

(b) Cosine loss of DLG attack per round.

(c) Distance of each randomly initialized location to the average location.

Figure 5.9: **FedSGD for one Round.** DLG converges (visually and it terms of cosine loss) to the average location regardless of the initialization point, or how far it was initialized from the average.

the better the reconstruction of the true locations (visually but also in terms of EMD), although the utility (RMSE) is not affected significantly. However, the most visited locations are reconstructed even with coarser rounds (1-week, 24-hour) which correspond to home/work locations of the target user. In the subsequent experiments, we consider one user that participates in FL unless stated otherwise.

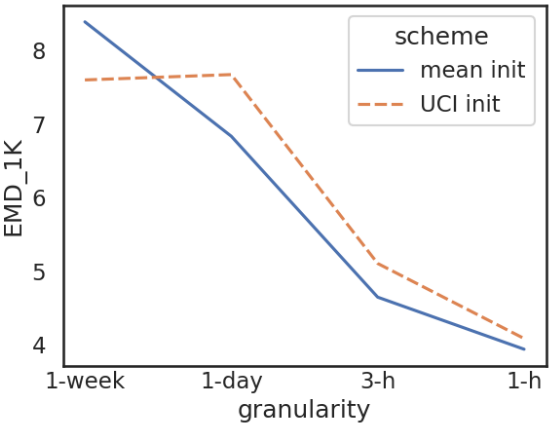


Figure 5.10: **Strongest Attack and two different initializations:** i) mean of the batch with Gaussian noise, ii) Campus center and then optimized initialization where the next batch is initialized with the previously reconstructed location. The finer the round granularity, the more leakage. Both are strong attacks regardless of the initialization (all reconstructed points converge) and result in similar behavior in terms of EMD.

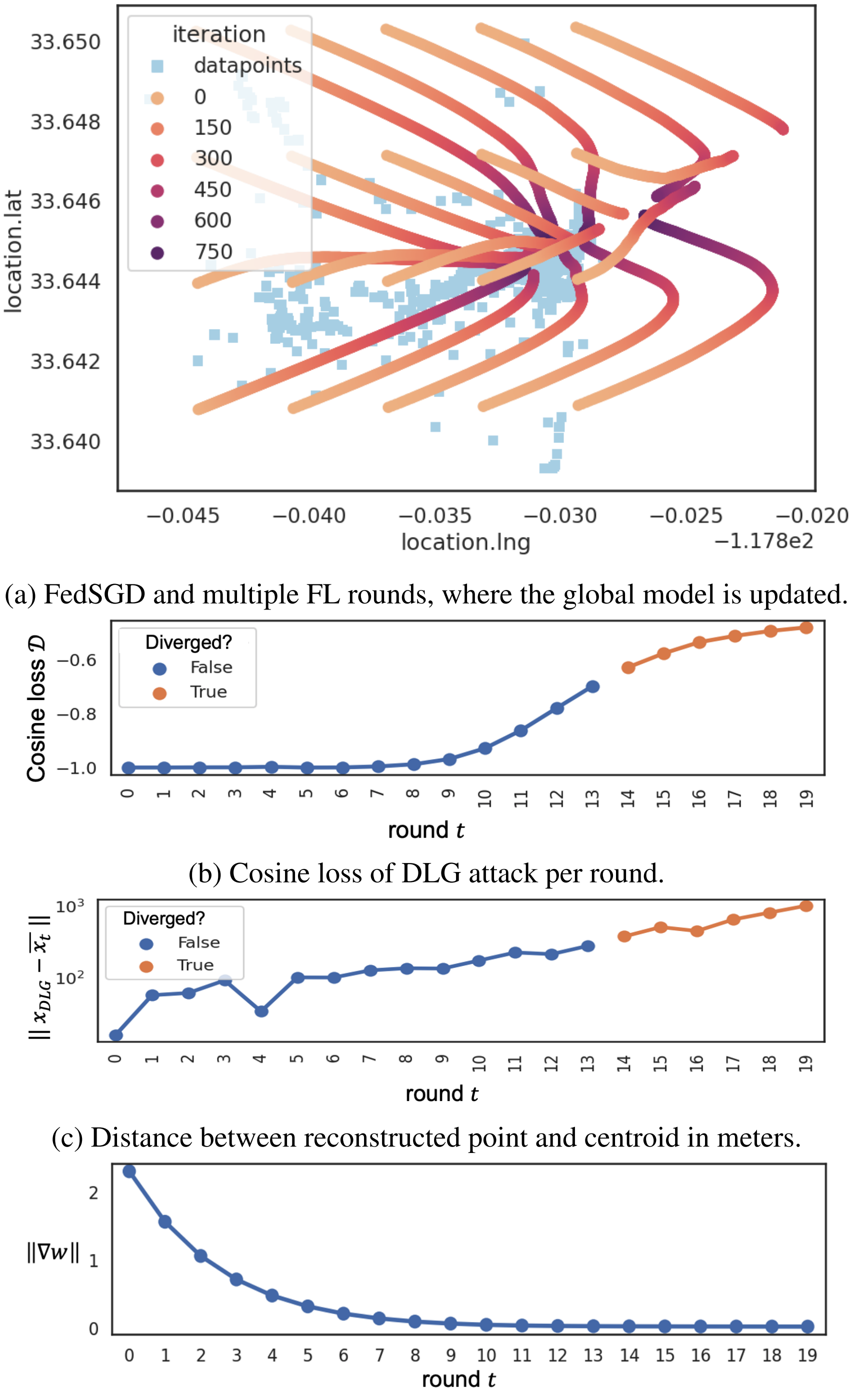
Impact of DLG Initialization. We evaluate the impact of different initializations on the convergence of the attack. We split the training data into grids of 350 meters and use the center of each grid as a potential DLG initialization point. As example, we obtained data from week 7 which will be used for local training in each round. In Fig. 5.9a there is exactly one FL round for each different initialization, that is the global model weights are initialized to the same random weights before local training and thus, we do not have any mechanism of gradient averaging. We observe that for all initializations the attack converges to the average location of the local data, regardless of the distance between the initialization point and the ground truth average of the data.

Optimized DLG Initialization. In practice the attacker can leverage the reconstructed location from a previous round and use it as the initialization point for the next attack in the next round. The reason is that in our setup there is continuity of user location patterns, especially in the finer intervals. We show in Fig. 5.10 the impact of the two initializations on the EMD metric when evaluating the strongest attack for different granularities of rounds. We consider here two options: i) the best initialization is when the attacker initializes with the mean of the batch with Gaussian noise added, and ii) fixed point and then "optimized" initialization where the subsequent reconstructed points are initialized with the previous reconstructed points. We observe that although the second initialization results to higher EMD for all intervals, the privacy leakage is still high. Both initialization do not result in any reconstructed locations outside the attacker's boundaries, and thus, we have no diverged points. In case there were such diverged points, the "optimized" initialization would not help since the initialization would be probably outside the defined boundaries/area so a random initialization would be better. For the subsequent experiment, we use the mean with added random noise as the default initialization for faster execution of the experiments. However, in practice, the attacker can start with a selected location within the defined boundaries and then use the "optimized" initialization in the subsequent rounds while also checking the cosine loss is not large in order to detect divergence.

Impact of Number of FL Rounds. On the other hand, Fig. 5.11a shows the case where the local data is the same and the same initializations are evaluated but the global model is updated in each round and in the next round the client starts local training with the updated global model. Clearly, the norm of the flattened per-layer weight matrices approaches zero after round 9 and at this point the DLG attack starts diverging; the reconstructed point is farther away from the mean of the data. In the worst case, the reconstructed point is 1km away from the mean location of the batch. This significantly lowers the attack effectiveness. Thus, even in FedSGD without any add-ons, there is some protection against the DLG attack due to multiple FL rounds and the global model converging. In the next subsection, we consider the DLG attack with Online FedAvg and evaluate how its parameters affect the attack's success due to additional averaging of gradients before the attacker receives them.

5.5.2 Location Leakage with FedAvg

In this section, we evaluate various FedAvg parameters in terms of utility (RMSE) and privacy metrics (EMD) between the true and reconstructed (via DLG) location distributions. We show that



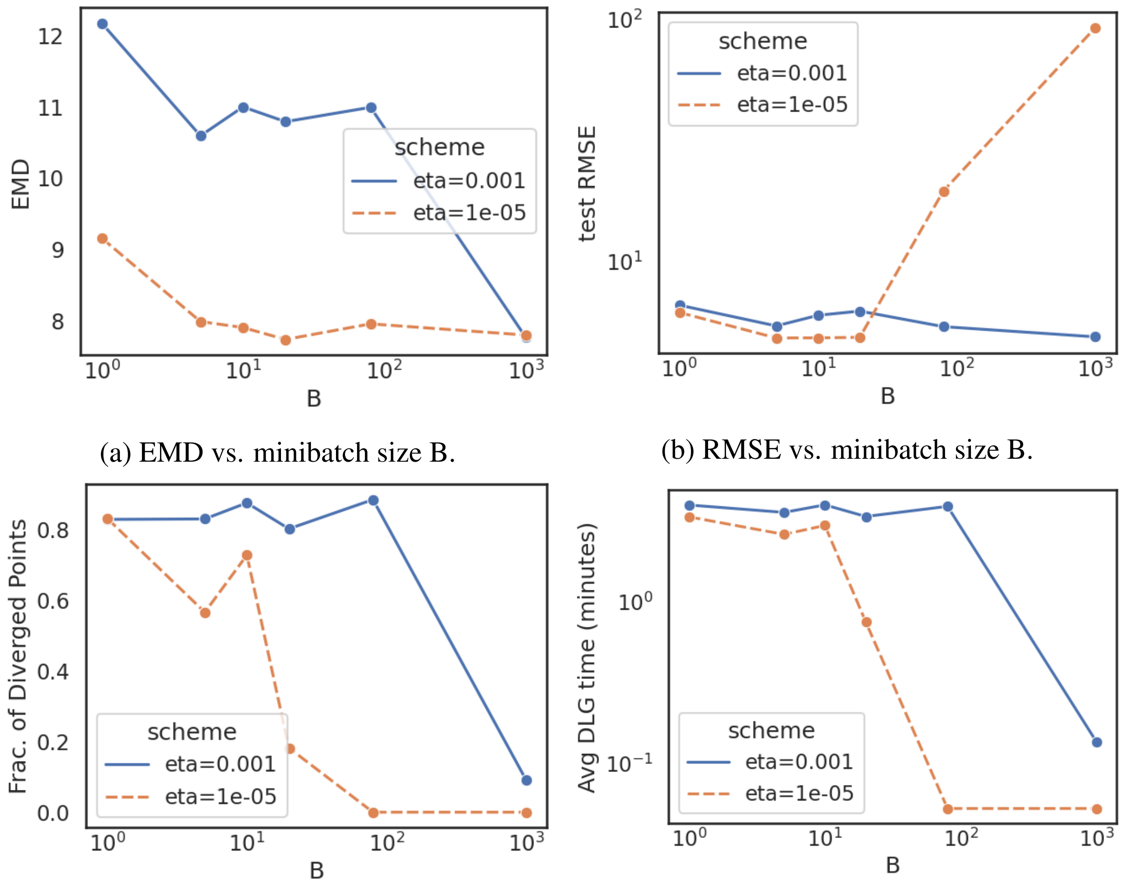
(a) FedSGD and multiple FL rounds, where the global model is updated.

(b) Cosine loss of DLG attack per round.

(c) Distance between reconstructed point and centroid in meters.

(d) Norm of gradient of flattened per-layer weight matrices between two consecutive rounds.

Figure 5.11: **FedSGD vs. Multiple Rounds.** The gradients are converging to zero with more FL rounds, which results to higher cosine loss and the reconstructed points are farther away from the centroid/average location. Here, the attacker also tries the same random initialization as in the single round case.



(a) EMD vs. minibatch size B.

(b) RMSE vs. minibatch size B.

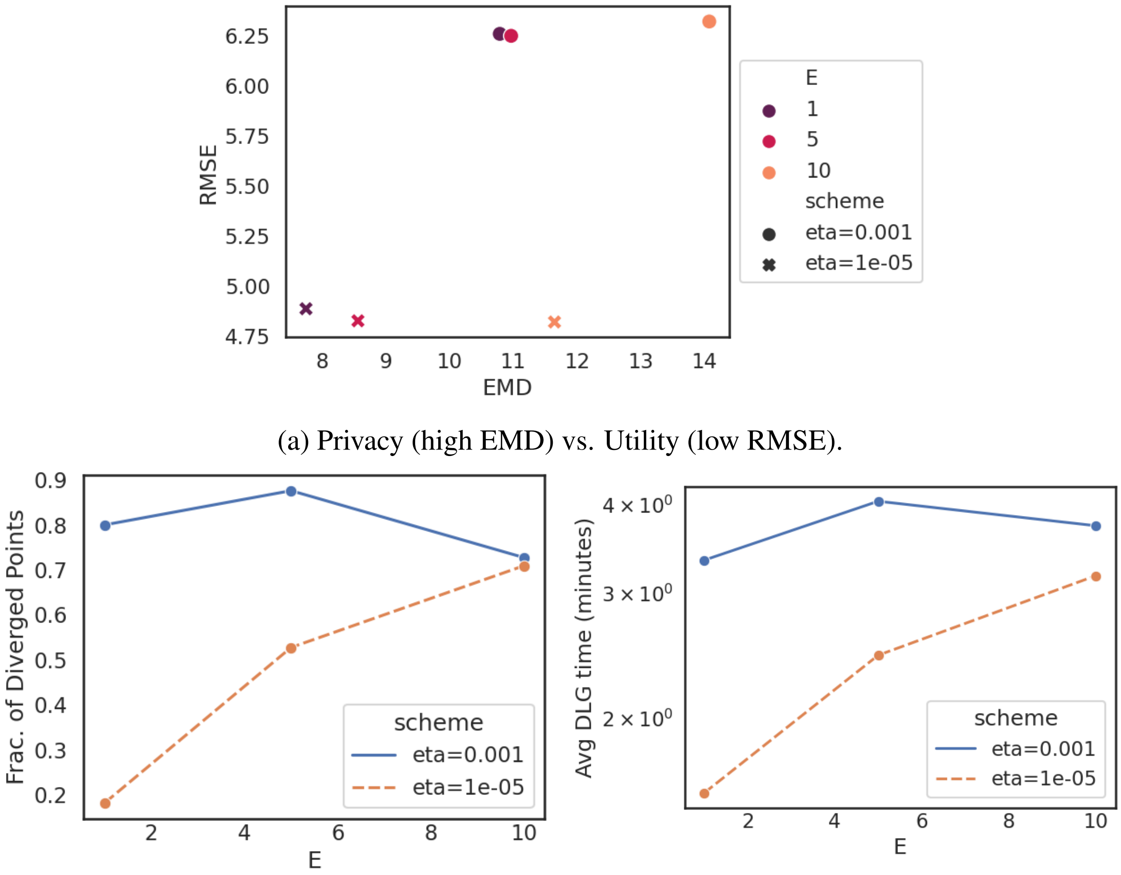
(c) DLG divergence.

(d) DLG avg execution time.

Figure 5.12: **Impact of minibatch size B.** Reducing minibatch size B introduces more averaging of the gradients which increases EMD (and privacy) and makes the attack more expensive due to divergence. B=∞ corresponds to FedSGD (minimum averaging) which leads to reduced privacy but also to higher RMSE for the lower η. Here we use 1-week rounds. The default η seems to be less sensitive to B in terms of RMSE. for B=1000, the user performs one SGD step with all available local data in each round.

FedAvg decreases the attack effectiveness due to multiple local SGD steps and the reconstructed locations are less accurate.

Impact of Minibatches. Fig. 5.12 shows the utility vs. privacy metrics when the users split their data into minibatches to perform multiple SGD steps where each FL round has 1-week worth of data. When B=1, there are as many SGD steps as the number of local datapoints for a user. This introduces maximum averaging of gradients and thus, privacy is maximum and the attack is more expensive (in terms of execution time and DLG iterations) due to higher divergence. Smaller B



(a) Privacy (high EMD) vs. Utility (low RMSE).

(b) More epochs increase divergence.

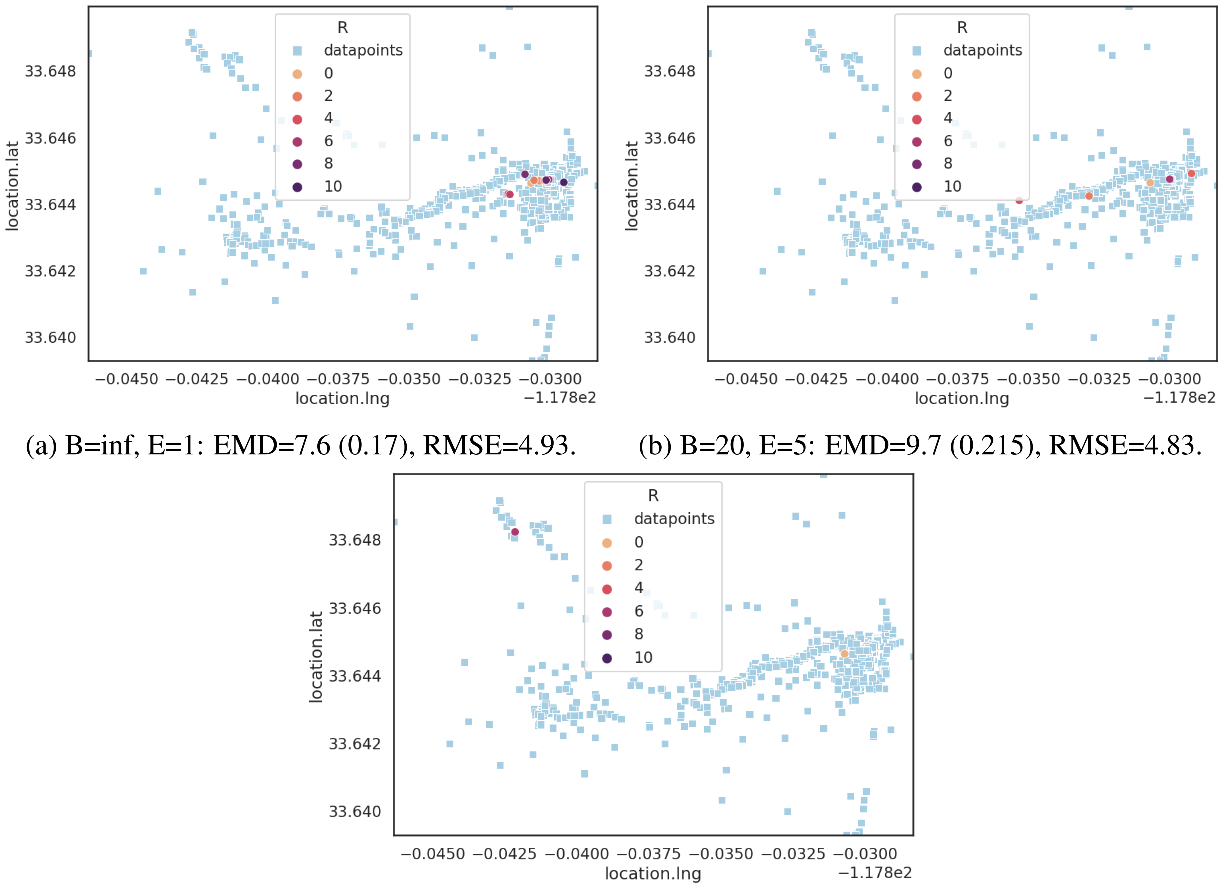
(c) More epochs increase DLG time.

Figure 5.13: **Impact of local epochs E on 1-week rounds.** We set $B = 20$ with default eta and η =1e-5 (optimized for mini-batches). Introducing more local epochs increases the EMD and thus, increase privacy, while utility is preserved. Increasing epochs also makes the attack more expensive in terms of time and less accurate (more divergence).

values are beneficial to utility especially to the lower η, but for the default $\eta = 0.001$ the RMSE is more robust. In terms of convergence of the DLG attacks, we show in Fig. 5.12c the fraction of reconstructed points that converged outside the attacker's defined boundaries. As the minibatch size B decreases, the fraction of diverged points increases due to increased SGD steps which makes the attack more expensive and less accurate. We choose $B = 20$ and the lower η =1e-05 in order to get some privacy protection (EMD slightly increases) and to maximize utility.

Impact of Local Epochs E. Another parameter of FedAvg is the number of epochs E during local training which corresponds to the number of local passes on the dataset and thus affects the number of SGD steps. In this case, we set the mini-batch size to 20, based on the previous experiment and evaluate the impact of local epochs for two learning rates η. Fig 5.13 shows how increasing the E increases the EMD and divergence in the attack thus, this additional averaging in the gradients protects privacy. On the other hand, additional passes on the local data do not seem to affect the RMSE significantly. We choose $E = 5$ on top of $B = 20$ in order to get even more privacy protection and better utility. However, we notice that even with these parameters the oversampled locations of the user can still be revealed, as shown in Fig. 5.14b. Therefore, the question is can we do better than this? We describe our algorithm for data selection next which has a data minimization effect and maintains utility.

Cumulative Online FL. So far, we considered online FL where the user train locally on the data that becomes available in the current round and discarding the local data from previous rounds. We also consider the other extreme, where the previously seen data are not discarded but they are re-used in addition to the newly acquired batch of data in each round. We call this scheme: cumulative online FL. Fig. 5.14 shows the reconstructed points in the case of the strongest attack, with averaging (B=20, E=5) and with cumulative online FL, which increases EMD to 19.2 without negatively affecting the RMSE. Although, this approach favors privacy, the oversampled location is still reconstructed (which corresponds to home/work location of the target) and also it requires the user to storage all their data which increases storage requirements and local training time. Moreover, if the user keeps re-using the older data during local training, the RMSE might decrease if the older data corresponds to older mobility patterns of the user. In this dataset, there are only 11 weeks so the user patterns are very similar from week to week. For the above reasons, we proposed our Diverse Batch algorithm, which achieves a similar RMSE and higher privacy due to higher EMD values , while using only a small subset of the data in each round without requiring storage of older data.



(a) B=inf, E=1: EMD=7.6 (0.17), RMSE=4.93.

(b) B=20, E=5: EMD=9.7 (0.215), RMSE=4.83.

(c) Cumulative online FL with B=20, E=5: EMD=19.2 (0.425), RMSE=4.85.

Figure 5.14: 1-week rounds, no averaging (B=inf, E=1) vs. B=20, E=5 vs. cumulative online FL. Without any averaging, even for coarser intervals the attacker can reconstruct accurately target's most frequent locations which correspond to home/work locations. Adding averaging results to more divergence (90%) and the converged locations are farther from the true locations. Cumulative online FL has increased EMD, although it still recovers the oversampled location (home/office) in the first round, but it also requires storage of all local data from all rounds which increases local training time.

5.5.3 Diverse Batch Algorithm for High Location Variance

So far we evaluated the impact of the magnitude of gradients on the DLG accuracy in terms of reconstruction of the average location in a batch of local data, which corresponds to the first term in Theorem 1. However, Theorem 1 is also bounded by the variance of the batch. We propose Algorithm 6 for controlling the variance of the both coordinates in a batch in order to provide more protection against DLG attacks. We evaluate our approach in practice with real-world data. Fig.

Algorithm 6: Diverse Batch algorithm for high location variance.

```
Input: $x$ training data samples; $eps$: max distance between samples in a cluster; $min\_samples$:
min samples in a cluster, $min\_B$: minimum number of batches to create
Output: list of batches with high variance
  clustered_data ← DBSCAN(x, eps, min_samples)
  centermost_points ← get_centermost(clustered_data)
  batch_list = [[centermost_points]]
  clustered_data.remove(centermost_points)
  while —batch_list— < min_B do
      centermost_points ← get_centermost(clustered_data)
      batch_list.append([centermost_points])
      clustered_data.remove(centermost_points)
end
  return batch_list
```

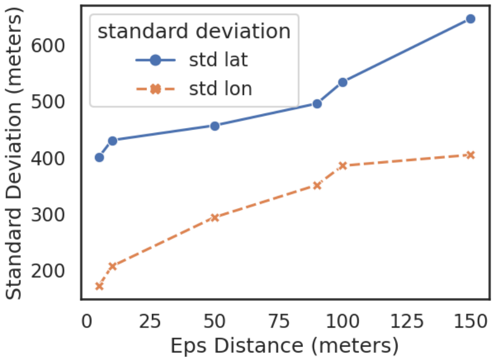


Figure 5.15: Average standard deviation (in meters) per batch for each coordinate and how it increases based on the eps parameter of DBSCAN which controls the maximum distance between points in a cluster. This motivate us to use the Algorithm 6 for creating batches with high variance. Here 1-week intervals were utilized.

5.15 shows the standard deviation for each coordinate and how it increases when we increase the eps (ϵ), which is in km, and also controls the number of clusters in the Algorithm 6. Thus, higher eps corresponds to fewer clusters obtained which also increases the location variance of the batch.

We propose Diverse Batch algorithm as shown in Algorithm 6 to create diverse batches that have high location variance via DBSCAN which also uses significantly fewer training datapoints and, thus, has a data minimization effect. In case of online FL, the algorithm is launched once for each FL round and the while loop is skipped. In particular, for each round, the user launches the algorithm on their corresponding local data and clusters their data based on the eps (ϵ) parameter, which corresponds to the maximum distance between points in a cluster. The higher the eps is, the fewer the clusters via DBSCAN and thus fewer datapoints are chosen and put in a batch. After obtaining the cluster, the center-most point from each cluster is put into the batch. Then, either a single SGD step is performed on the resulting batch or the current batch of data is split further into mini-batches and/or multiple passes on the batched data are performed due to local epochs. The algorithm can be applied multiple times within a round, or in standard offline FL, in which case the desired number of batches (min_B) must be defined and the selected center-most points will be removed before the algorithm is repeated on the remaining data. We evaluate the algorithm in terms of privacy and utility with the aforementioned metrics and compare it to scenarios without manipulated batches and to a random baseline that randomly selects the same number of points from the available data in a round as Algorithm 6 would have selected.

We would like to note that Algorithm 6 is used in the case of online learning: for each FL round the user will perform clustering of their local data and then will put the centermost points from each cluster into a single batch. Next, the constructed batches are either used as is, or they are split further into mini-batches. We demonstrate it for 1-week intervals, where for each round we obtain the batches and perform local training on them or we split them into mini-batches of 20 and perform 5 local epochs before the user sends their updates to the server. We tuned the learning rate via a grid search similarly to the non-manipulated batches and the best η was 0.001.

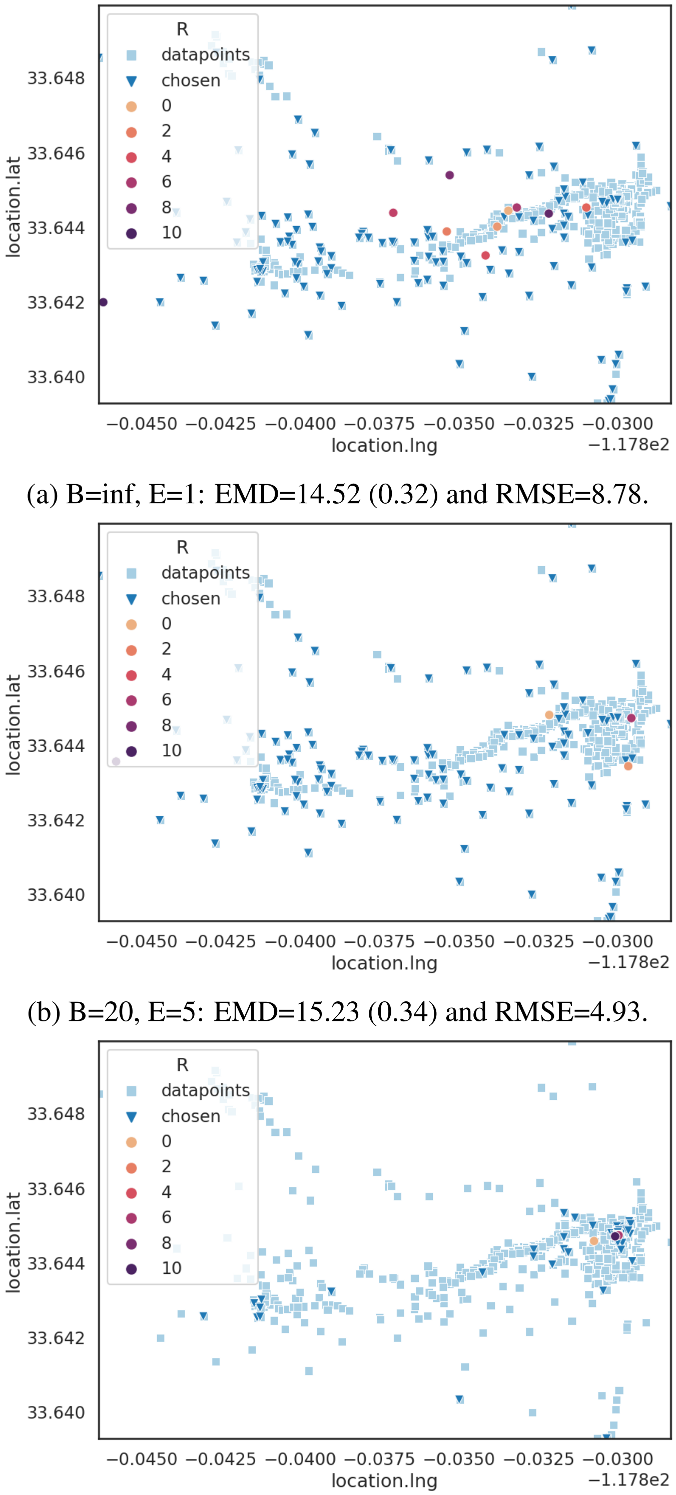
Table 5.2 presents the results. We observe that increasing eps decreases the number of training points results to smaller batches on average, due to fewer clusters and thus, fewer datapoints are selected from the available data in each round. We observe an increase in RMSE when we have no averaging, but with B=20, E=5 the RMSE is not affected based on the eps value. On the other hand, EMD is increasing for larger eps values in both cases, but with averaging the resulting EMD is higher overall, while the percentage of diverged reconstructed locations increases which makes the attack more expensive and less accurate, since the avg distance of the reconstructed locations from the corresponding centroid of the batches also increases. As a baseline, we report the RMSE and

eps	avg. B	% chosen	RMSE	EMD/B=20,E=5	% diverged	avg dist	Rnd RMSE/EMD
0.0001	239	6.84	5.24/4.876	9.61(0.21)/10.59(0.24)	18/82	139/265	4.82/7.5(0.17)
0.001	180	5.2	5.34/4.86	9.72(0.21)/10.84(0.24)	18/90	134/245	4.83/7.85(0.17)
0.005	98	2.8	5.78/4.83	10.72(0.24)/14.15(0.31)	9/57	170/290	4.82/7.9(0.18)
0.05	16	0.45	8.78/4.93	14.524(0.32)/15.23(0.34)	13/64	331/345	4.96/7.6(0.17)

Table 5.2: Batch manipulation with 1-week rounds via Algorithm 6: the DBSCAN algorithm is run in each round to obtain clusters and add the center-most points to a batch. We set $\eta = 0.001$, dropout=0.05. Default values correspond to B=inf, E=1. The second values in RMSE, EMD, % diverged, correspond to B=20, E=5. As a baseline, we choose randomly the same amount of datapoints chosen by DBSCAN per batch with E=5, B=20. Although RMSE is not impacted, the EMD is approx. half with the random method since the variance of the batches is not affected. Another baseline is to use all the data in each round with B=20, E=5 which results to RMSE=6.26, EMD=10.73 (0.24) with 72% divergence. We also report the average distance of the converged reconstructed locations.

EMD of randomly selected points in each round, after getting the number of clusters via DBSCAN in order to sample the same amount of datapoints for a fair comparison. We set $B = 20$, $E = 5$ in order to compare our algorithm with averaging. We observe that in this baseline, the EMD remains low regardless of the eps value, since there is the variance of each batch is not controlled.

Comparison of Methods. We summarize and visualize the DLG attack effectiveness in Fig. 5.16, for 1-week intervals and show how EMD increases with FedAvg with added mini-batches of size 20 and 5 local epochs, from the strongest attack with FedSGD. Next, we compare to non-online FL which results to EMD that is almost equivalent to the random EMD baseline, but still the oversampled user's location is still reconstructed. Next, we compare our Diverse Batch algorithm in Fig. 5.18 with no averaging, with averaging and the random baseline. Our approach with averaging reaches EMD comparable to random EMD, while RMSE remains low. It is also noticeable that the reconstructed locations are not revealing the oversampled user's location but instead the locations are more spread. For comparison, the random baseline still reveals the oversampled user's location which most likely corresponds to one of the most important locations of the user (home/work, etc.). Finally, we summarize the privacy vs. utility plots in Fig. 5.17 along with the additional metrics. First, we summarize here the EMD values, starting from the strongest attack (7.5), mini-batches, local epochs, and until our approach (15). Second, we show the fraction of diverged reconstructed

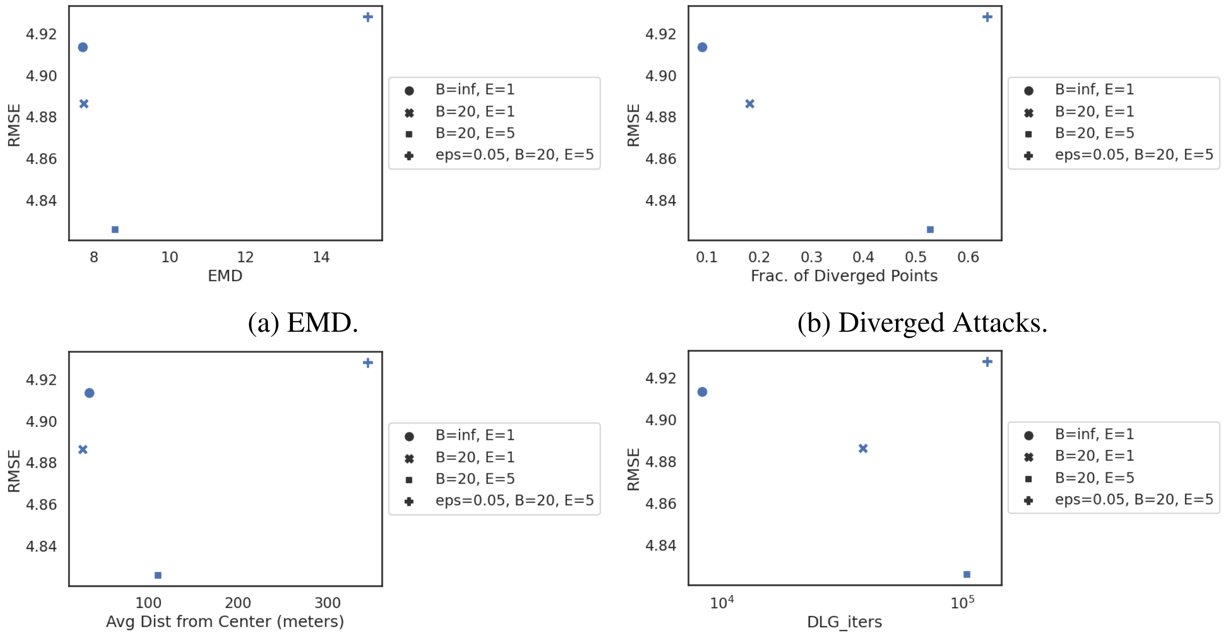


(a) B=inf, E=1: EMD=14.52 (0.32) and RMSE=8.78.

(b) B=20, E=5: EMD=15.23 (0.34) and RMSE=4.93.

(c) random baseline, B=20, E=5: EMD=7.6 (0.17) and RMSE=4.96.

Figure 5.16: **Batch Manipulation.** User's locations (light blue), chosen points (dark blue) with Diverse Batch algorithm with eps=0.05 km, and the final reconstructed points (circles). Only the converged reconstructed points are shown here. The baseline randomly samples from each round the same amount of locations as the chosen locations with DBSCAN but the high variance in the batch is not controlled which results to lower EMD. Unlike FedAvg, the oversampled location is not revealed with our algorithm which makes it a stronger defense.



(a) EMD.

(b) Diverged Attacks.

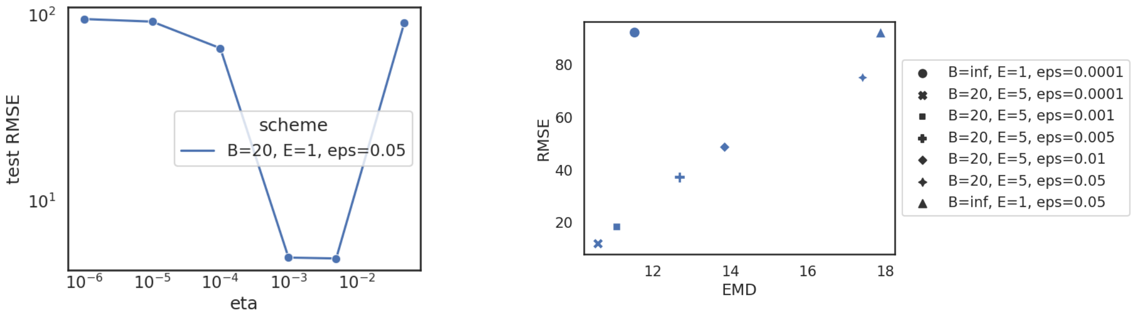
(c) Distance from centroid.

(d) Avg DLG iterations.

Figure 5.17: **Comparison for 1-week rounds.** We start with strongest attack (FedSGD) without averaging (B=inf, E=1), then add minibatches $B = 20$, then add local epochs $E = 5$. Finally, we add the manipulated batches with averaging which increase EMD but RMSE is not significantly higher. Our method increases privacy but maintains utility while only less than 1% of the data.

locations to indicate how expensive is the DLG attack in each scenario. In the strongest attack, there is almost no divergence, while with our approach the divergence increases to more than 60%, which makes the DLG attack more expensive since the attacker needs to relaunch the attack with other initializations. Fourth, we also report the avg distance between the reconstructed location and the mean location of its corresponding ground truth data, as additional privacy metric which is intuitive. With the strongest attack this is less than 30 meters, while with the discussed defenses this increases to more than 350 meters.

Fig. 5.18a how the selected learning rate minimizes RMSE in case of Diverse Batch algorithm. Fig 5.18b shows the results for a lower learning rate, which increases RMSE but the privacy patterns are similar to the tuned η.



(a) RMSE vs. various η.

(b) RMSE vs. EMD for various eps when $\eta = 10^{-5}$.

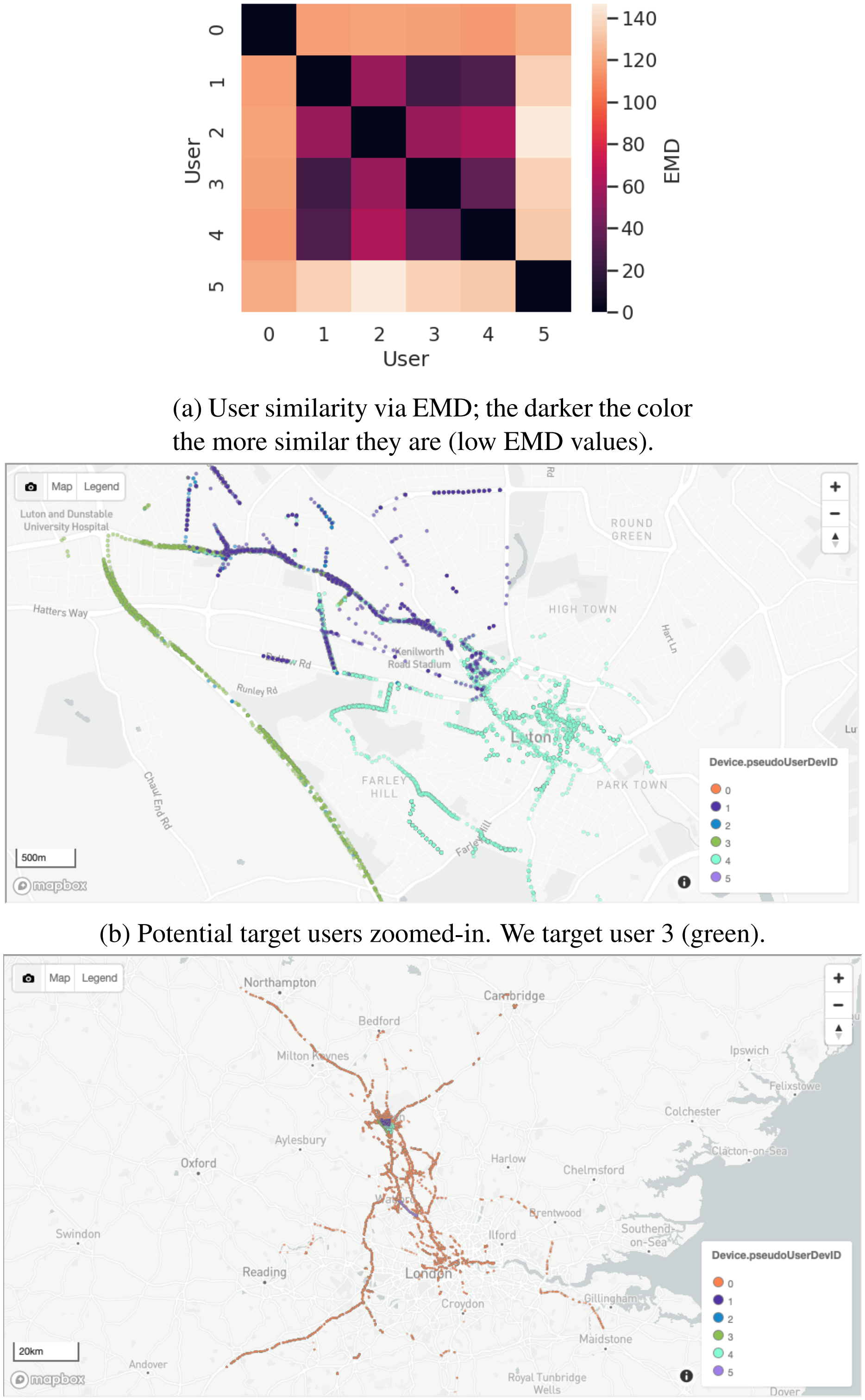
Figure 5.18: **Diverse Batch algorithm with various η.** For lower eta, the EMD behaves similarly but RMSE is higher. We choose $\eta = 0.001$.

5.5.4 Multiple Users in FL

Next, we evaluate the scenario where the attacker still targets a specific user as before, but there is at least one other user who participates in FedAvg by uploading their model updates in a round. We use the London 2017 Radiocells Dataset which was described above, as it contains significantly more users than the UCI Campus LTE Dataset. To simplify the scenario and evaluate the impact of the other user updating the global model, we assume in each FL round either the target participates or the other user(s) and they do so interchangeably.

We choose the cell x455 that contains the most measurements (169K) from 39 weeks and it has 900 users (merged into a single user 0) and 5 potential target users (user 1-5). Next, we obtain the user similarity on their EMD values when comparing to the visited locations of the target user, which is shown in Fig. 5.19a. The lighter the color the most different are the users due to a higher EMD value between them.

First, we show in Fig. 5.20 the strongest attack for user 3, and then adding more averaging via B=10 and E=5 the EMD increases from 21 to 22.2 while the RMSE slightly drops. Then, we add all users from the orange trajectory (which includes more than 900 upload files), which we consider as user 0 who participates in FL and updates the global model after locally training on the corresponding data. In this scheme, the test RMSE is obtained on the test set from all participating

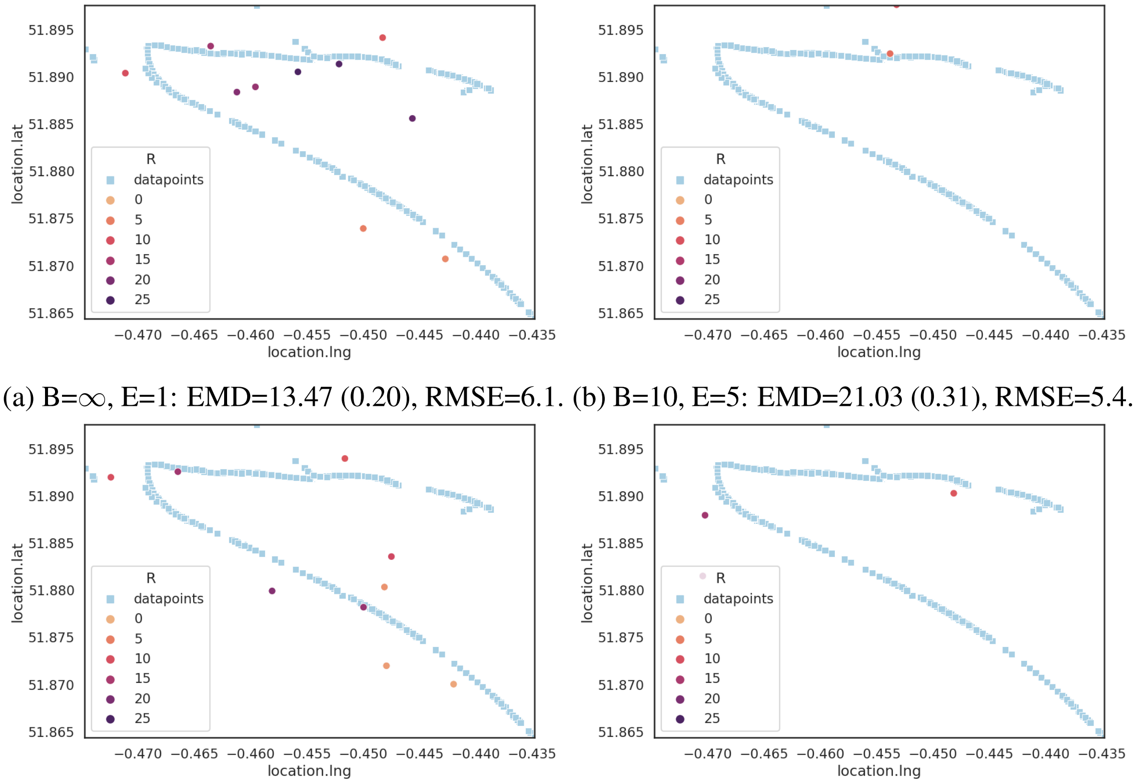


(a) User similarity via EMD; the darker the color the more similar they are (low EMD values).

(b) Potential target users zoomed-in. We target user 3 (green).

(c) All users.

Figure 5.19: Radiocells users for cell x455. User 0 contains multiple users across all London region, while users 1, 2, 3, 4, 5 are possible target users.



(a) B=∞, E=1: EMD=13.47 (0.20), RMSE=6.1.

(b) B=10, E=5: EMD=21.03 (0.31), RMSE=5.4.

(c) multi-user B=∞, E=1: EMD=22.24 (0.33), RMSE=5.77.

(d) multi-user B=10, E=5: EMD=22.515 (0.33), RMSE=5.41.

Figure 5.20: Radiocells LTE x455 with target user 3 for 1-week rounds with $\eta = 0.001$. The RMSE is evaluated on global test set when multiple users. The avg random EMD is 34 (0.5).

users, while in the single user scenario, the RMSE was obtained on the target's test data. Adding the user 0, increases the EMD and reduces the RMSE which increases privacy and utility. Adding mini-batches and local epochs when user 0 participates in FL, has a slight effect on EMD compared to B=inf, E=1. We summarize these results in Table 5.3 and report the divergence of the DLG attack. Next, we add one of the other potential target users, one at a time, to evaluate the effect of similar/dissimilar users participating in FedAvg and updating the global model with their updates. We observe that when the most dissimilar user (user 5) participates along with target user 3, then the EMD is low and there is more privacy leakage than when more similar users (user 1, 2, 4) participate. This is due to different gradients in the case of dissimilar users which slows down convergence and the attack can reconstruct more accurately the local data of the target user.

Scheme	User(s)	RMSE	EMD	% diverged
FedSGD	user 3	6.1	17.08	65
FedAvg	user 3	5.43	24.13	91
FedAvg, $\epsilon = 0.005$	user 3	5.42	25.2	97
FedAvg, $\epsilon = 0.01$	user 3	5.44	26.9	97
FedAvg	user 3, 0	5.43	22.51	90
FedAvg	user 3, 1	5.47	29.16	95
FedAvg	user 3, 2	5.43	26.15	95
FedAvg	user 3, 4	5.47	29.30	95
FedAvg	user 3, 5	5.42	23.02	92
FedAvg, $\epsilon = 0.01$	user 3, 5	5.43	30.9	95

Table 5.3: Single vs. Multi-user for Radiocells Dataset for target 3. $\eta = 0.001$ results are averaged from multiple runs. FedAvg corresponds to $E = 5, B = 10$. When ϵ (in meters) is reported, Diverse Batch algorithm was applied on the target's local data.

We add our Diverse Batch algorithm on top of FedAvg for the target user (user 3). For $\epsilon = 0.005$ meters, one of out three runs resulted to 100% divergence and the EMD could not be obtained. We average the divergence across multiple rounds and report the average, while for EMD we only report the average from the runs that did not result to 100% divergence. Increasing ϵ to 0.01 meters, results to two of of three runs with 100% divergence and the run that had some converged points had EMD=26.9 and RMSE=5.44. Finally, we evaluate the multi-users case, where only the target uses the Diverse Batch algorithm on their local data. We observe that this scheme improves further privacy protection, even in the case of dissimilar users.

5.6 Summary

In this work, we demonstrated, for the first time, the location leakage in federated learning based on gradients and on real-world signal strength maps data when the attacker is an honest-but-curious server. First, we quantify the leakage in practical scenarios and specifically in the case of online FedAvg without any privacy add-ons like Differential Privacy, where the local/private data is not static but instead the local data are streaming data based on time. In contrast to the standard image classification attacks, we showed that location leakage is limited due to the averaging mechanisms of FL which correspond to increased SGD steps due to mini-batches, local epochs. Next, we propose an algorithm for data selection in order to create batches that increase the location variance which in turn increases privacy without hurting utility although a small fraction of data is used for local training in each FL round. Finally, we evaluate the impact of multiple users who participate in FedAvg on a larger and real-world dataset.

Future Work. We considered signal maps as one application of DLG attacks on FL, however the same methodology can be extended to any learning task that requires crowdsourced (online) spatial data for training. Some interesting directions for future work are the following: (i) after the reconstruction of datapoints, how to use them to infer further information about the users? One could infer clusters of important places [73], *i.e.,* home, friend's houses, hospital, etc., ii) if secure aggregation is in-place, then identify the user based on reconstructed data similarly to "Unique in the Crowd" [51], (ii) how does personalization in FL affect DLG attacks? For instance, there are works [84] that modify the user gradients in order to not diverge from the global model updates but they have not been evaluated in terms of privacy vs. utility trade-off.

Chapter 6

Conclusion

In this thesis, we considered mobile data privacy risks, through tracking and inference, and we developed privacy enhancing mechanisms. First, we conducted a user study to assess how users perceive privacy risk in explicit mobile tracking via apps when real privacy exposures based on Personally Identifiable Information (PII) is shown from 400 apps in Android. The users seemed to be confused about how apps are sharing their information with third parties but after our short tutorial on mobile data privacy they became more aware and showed appetite for privacy control.

Next, we proposed federated learning on two mobile data applications: i) mobile packet classification in order to control/prevent explicit tracking on mobile by detecting PII exposure or Ad request in outgoing packets; and ii) signal maps prediction that trains regression model with location features in order to predict signal strength in LTE. In both applications, the data stays on the device and devices train collaboratively a global model via federated learning which raises the privacy bar compared to centralized training.

In the first FL application, we proposed FedPacket and evaluated its performance it terms of prediction performance, communication, computation in various scenarios. We showed it can achieve good classification performance and comparable to centralized state-of-the-art. However, FL can still reveal information about the local data via the exchanged gradients of model parameters. We demonstrated for the first time a privacy attack on HTTP features and showed two inference attacks and how they are affected by various FL parameters.

In the second FL application, the federated signal maps, we demonstrated privacy attack based on gradients when the data is not static but becomes available over time as the user moves around and the data are collected. We showed that averaging of gradients due to FL parameters provides natural protection against such attacks and we proposed a clustering based algorithm for data selection/minimization to further protect privacy without hurting utility.

In this thesis, we investigated specific issues related to mobile data privacy, with focus on federated learning. More generally, we are already in the all time high mobile usage era and privacy of mobile data is increasingly important over time. To provide more transparency and control to the user, a synergy of technical, such as federated learning and privacy-preserving approaches, and policy solutions are intrinsic going forward. After all, privacy is a human right.

Bibliography

[1] Adaway. https://adaway.org/hosts.txt.

[2] Adblock browser. https://adblockbrowser.org.

[3] Amazon mechanical turk. https://www.mturk.com.

[4] AntShield Dataset. https://athinagroup.eng.uci.edu/projects/antmonitor/antshield-dataset/.

[5] App annie. https://www.appannie.com.

[6] Best Practices for Unique Identifiers. https://developer.android.com/training/articles/user-data-ids.

[7] California consumer privacy act (ccpa). https://oag.ca.gov/privacy/ccpa.

[8] EasyList. https://easylist.to/.

[9] EU General Data Protection Regulation (GDPR). https://eugdpr.org.

[10] Google play. https://play.google.com/store?hl=en.

[11] hphosts. https://hosts-file.net/ad_servers.txt.

[12] Mother of all adblockers. http://adblock.mahakala.is.

[13] NoMoAds Dataset. https://athinagroup.eng.uci.edu/projects/nomoads/data/.

[14] PhoneLab, University at Buffalo. https://www.phone-lab.org/.

[15] Pretrained word2vec. https://code.google.com/archive/p/word2vec/.

[16] Radiocells Dataset. https://radiocells.org. Accessed: 2021-10-20.

[17] Speed guide: Ports database. http://www.speedguide.net/ports.php.

[18] Ui/application exerciser monkey. https://developer.android.com/studio/test/monkey.

[19] Your Apps Know Where You Were Last Night, and They're Not Keeping It Secret. `https://nyti.ms/3a5VCbp`. Accessed: 2021-10-20.

[20] Permanent Message Header Field Names. `http://www.iana.org/assignments/message-headers/message-headers.xhtml#perm-headers`, 2019.

[21] M. Abadi, U. Erlingsson, I. Goodfellow, H. B. McMahan, I. Mironov, N. Papernot, K. Talwar, and L. Zhang. On the protection of private information in machine learning systems: Two recent approches. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 1–6, Aug 2017.

[22] E. Alimpertis. *Mobile Coverage Maps Prediction*. PhD thesis, UC Irvine, 2020.

[23] E. Alimpertis and A. Markopoulou. A system for crowdsourcing passive mobile network measurements. submitted to NSDI Posters, February 2017.

[24] E. Alimpertis, A. Markopoulou, C. Butts, and K. Psounis. City-wide signal strength maps: Prediction with random forests. In *The World Wide Web Conference*, pages 2536–2542, 2019.

[25] E. Alimpertis, A. Markopoulou, and U. Irvine. A system for crowdsourcing passive mobile network measurements. *14th USENIX NSDI*, 17, 2017.

[26] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pages 11849–11860, 2019.

[27] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pages 11816–11825, 2019.

[28] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal. Your location has been shared 5,398 times!: A field study on mobile app privacy nudging. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 787–796. ACM, 2015.

[29] S. Augenstein, H. B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019.

[30] E. Bakopoulou, A. Shuba, and A. Markopoulou. Exposures exposed: a measurement and user study to assess mobile data privacy in context. *arXiv preprint arXiv:2008.08973*, 2020.

[31] E. Bakopoulou, B. Tillman, and A. Markopoulou. A federated learning approach for mobile packet classification. *arXiv preprint arXiv:1907.13113*, 2019.

[32] E. Bakopoulou, B. Tillman, and A. Markopoulou. Fedpacket: A federated learning approach to mobile packet classification. *IEEE Transactions on Mobile Computing*, 2021.

[33] E. Bakopoulou, J. Zhang, J. Ley, K. Psounis, and A. Markopoulou. Location leakage in federated signal maps. *in preparation, 2021.*

[34] M. Bastian, S. Heymann, M. Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. Icwsm*, 8, 2009.*

[35] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo. Analyzing federated learning through an adversarial lens. arXiv preprint arXiv:1811.12470*, 2018.*

[36] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers. Protection against reconstruction and its applications in private federated learning. arXiv preprint arXiv:1812.00984*, 2018.*

[37] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, et al. Towards federated learning at scale: System design. arXiv preprint arXiv:1902.01046*, 2019.*

[38] K. Bonawitz, V. Ivanov, B. Kreuter, and A. Marcedone. Practical Secure Aggregation for Privacy Preserving Machine Learning. Eprint.Iacr.Org*.*

[39] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister. Sliced and radon wasserstein barycenters of measures. Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.*

[40] L. Bottou. Stochastic gradient descent (v.2). `https://leon.bottou.org/projects/sgd`*.*

[41] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, Advances in Neural Information Processing Systems 20*, pages 161–168. Curran Associates, Inc., 2008.*

[42] L. Bottou and Y. LeCun. Large scale online learning. Advances in neural information processing systems*, 16:217–224, 2004.*

[43] S. Boukoros, M. Humbert, S. Katzenbeisser, and C. Troncoso. On (the lack of) location privacy in crowdsourcing applications. In 28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1859–1876, 2019.*

[44] Y. J. Bultitude and T. Rautiainen. IST-4-027756 WINNER II D1. 1.2 V1. 2 WINNER II Channel Models. Technical report, 2007.

[45] S. Caldas, J. Konecný, H. B. McMahan, and A. Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. CoRR*, abs/1812.07210, 2018.*

[46] A. Chakraborty, M. S. Rahman, H. Gupta, and S. R. Das. Specsense: Crowdsensing for efficient querying of spectrum occupancy. In Proc. of the IEEE INFOCOM '17*.*

[47] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In 2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE, 2020.*

[48] *S. Chitkara, N. Gothoskar, S. Harish, J. I. Hong, and Y. Agarwal. Does this app really need my location? context-aware privacy management for smartphones.* Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):42, 2017.*

[49] *A. Continella, Y. Fratantonio, M. Lindorfer, A. Puccetti, A. Zand, C. Kruegel, and G. Vigna. Obfuscation-resilient privacy leak detection for mobile apps through differential analysis. 2017.*

[50] *G. Damaskinos, R. Guerraoui, A.-M. Kermarrec, V. Nitu, R. Patra, and F. Taïani. Fleet: Online federated learning via staleness awareness and performance prediction. In* Proceedings of the 21st International Middleware Conference*, pages 163–177, 2020.*

[51] *Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility.* Scientific reports*, 3:1376, 2013.*

[52] *M. Diao, Y. Zhu, J. Ferreira, and C. Ratti. Inferring individual daily activities from mobile phone traces: A boston example.* Environment and Planning B: Planning and Design*, 43, 09 2015.*

[53] *C. Dwork. Differential privacy.* Encyclopedia of Cryptography and Security*, pages 338–340, 2011.*

[54] *M. Egele, C. Kruegel, E. Kirda, and G. Vigna. Pios: Detecting privacy leaks in ios applications. In* NDSS*, 2011.*

[55] *R. Enami, D. Rajan, and J. Camp. RAIK: Regional analysis with geodata and crowdsourcing to infer key performance indicators. In* Proc. of the IEEE WCNC*, Apr. 2018.*

[56] *W. Enck, P. Gilbert, B. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones.* ACM TOCS*, 2014.*

[57] *W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones.* ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.*

[58] *H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A First Look at Traffic on Smartphones. In* Proc. of the 10th ACM SIGCOMM Conf. on Internet Measurement*, Melbourne, Australia, Nov. 2010.*

[59] *M. R. Fida, A. Lutu, M. K. Marina, and O. Alay. ZipWeave: Towards efficient and reliable measurement based mobile coverage maps. In* Proc. of the IEEE INFOCOM '17*, May 2017.*

[60] *R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. Pot: Python optimal transport.* Journal of Machine Learning Research*, 22(78):1–8, 2021.*

[61] *C. Galbraith, P. Smyth, and H. S. Stern. Statistical methods for the forensic analysis of geolocated event data.* Forensic Science International: Digital Investigation*, 33:301009, 2020.*

[62] *J. Garcia-Reinoso et al. The 5g eve multi-site experimental architecture and experimentation workflow. In* IEEE 2nd 5G World Forum*, pages 335–340, Sep. 2019.*

[63] *K. Garimella, O. Kostakis, and M. Mathioudakis. Ad-blocking: A study on performance, privacy and counter-measures. In* Proceedings of the 2017 ACM on Web Science Conference*, pages 259–262. ACM, 2017.*

[64] *J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors,* Advances in Neural Information Processing Systems*, volume 33, pages 16937–16947. Curran Associates, Inc., 2020.*

[65] *R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective.* CoRR*, abs/1712.07557, 2017.*

[66] *C. Gibler, J. Crussell, J. Erickson, and H. Chen. Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In* Proc. of the Intl. Conf. on Trust and Trustworthy Computing*, 2012.*

[67] *D. Gugelmann, M. Happe, B. Ager, and V. Lenders. An automated approach for complementing ad blockers' blacklists.* Proceedings on Privacy Enhancing Technologies*, 2015(2):282–298, 2015.*

[68] *N. Guha, A. Talwlkar, and V. Smith. One-shot federated learning.* arXiv preprint arXiv:1902.11175*, 2019.*

[69] *A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction.* arXiv preprint arXiv:1811.03604*, 2018.*

[70] *S. He and K. G. Shin. Steering crowdsourced signal map construction via bayesian compressive sensing. In* Proc. of the IEEE INFOCOM '18*, pages 1016–1024, Apr. 2018.*

[71] *B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In* Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618. ACM, 2017.*

[72] *A. Imran, A. Zoha, and A. Abu-Dayya. Challenges in 5G: How to empower SON with big data for enabling 5G.* IEEE network*, 28(6):27–33, 2014.*

[73] *S. Isaacman, R. Becker, R. Cáceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky. Identifying important places in people's lives from cellular network data. In* International Conference on Pervasive Computing*, pages 133–151. Springer, 2011.*

[74] Q. Ismail, T. Ahmed, K. Caine, A. Kapadia, and M. Reiter. To Permit or Not to Permit, That is the Usability Question: Crowdsourcing Mobile Apps' Privacy Permission Settings. Proceedings on Privacy Enhancing Technologies*, 4(4):118–136, 2017.*

[75] Z. Jorgensen, J. Chen, C. S. Gates, N. Li, R. W. Proctor, and T. Yu. Dimensions of Risk in Mobile Applications. Proceedings of the 5th ACM Conference on Data and Application Security and Privacy - CODASPY '15*, pages 49–60, 2015.*

[76] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977*, 2019.*

[77] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.*

[78] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated Learning: Strategies for Improving Communication Efficiency. pages 1–10, 2016.

[79] E. Krijestorac, S. S. Hanna, and D. Cabric. Spatial signal strength prediction using 3d maps and deep learning. ICC 2021 - IEEE International Conference on Communications*, pages 1–6, 2021.*

[80] A. K. Laha and S. Putatunda. Real time location prediction with taxi-gps data streams. Transportation Research Part C: Emerging Technologies*, 92:298–322, 2018.*

[81] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research*, 18(185):1–52, 2018.*

[82] M. Li, T. Zhang, Y. Chen, and A. J. Smola. Efficient mini-batch training for stochastic optimization. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670, 2014.*

[83] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In 2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115, 2007.*

[84] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems*, 2:429–450, 2020.*

[85] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. On the convergence of fedavg on non-iid data. arXiv preprint arXiv:1907.02189*, 2019.*

[86] Y. Li, F. Chen, T. J.-J. Li, Y. Guo, G. Huang, M. Fredrikson, Y. Agarwal, and J. I. Hong. Privacystreams: Enabling transparency in personal data processing for mobile apps. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3):76:1–76:26, Sept. 2017.*

[87] J. Lin, N. Sadeh, S. Amini, J. Lindqvist, J. I. Hong, and J. Zhang. Expectation and purpose: Undestanding Users' Mental Models of Mobile App Privacy through Crowdsourcing. Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, page 501, 2012.*

[88] B. Liu, M. S. Andersen, F. Schaub, H. Almuhimedi, S. A. Zhang, N. Sadeh, Y. Agarwal, A. Acquisti, M. Schaarup Andersen, F. Schaub, H. Almuhimedi, S. A. Zhang, N. Sadeh, Y. Agarwal, and A. Acquisti. Follow My Recommendations: A Personalized Privacy Assistant for Mobile App Permissions. Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, (Soups):27–41, 2016.*

[89] B. Liu, B. Liu, H. Jin, and R. Govindan. Efficient Privilege De-Escalation for Ad Libraries in Mobile Apps. In Proceedings of the 13th annual international conference on mobile systems, applications, and services*, pages 89–103. ACM, 2015.*

[90] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang. Fedvision: An online visual object detection platform powered by federated learning. In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13172–13179, 2020.*

[91] M. Ma. Enhancing privacy using location semantics in location based services. In 2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*, pages 368–373, 2018.*

[92] W. Mason and S. Suri. Conducting Behavioral Research on Amazon' s Mechanical Turk. Behavior Research Methods*, 5(5):1–23, 2010.*

[93] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.*

[94] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private language models without losing accuracy. arXiv:1710.06963*, 2017.*

[95] A. Mehrotra, S. R. Müller, G. M. Harari, S. D. Gosling, C. Mascolo, M. Musolesi, and P. J. Rentfrow. Understanding the Role of Places and Activities on Mobile Phone Interaction and Usage Patterns. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–22, 2017.*

[96] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. IEEE, 2019.

[97] Y. Nan, Z. Yang, X. Wang, Y. Zhang, D. Zhu, and M. Yang. Finding Clues for Your Secrets: Semantics-Driven, Learning-Based Privacy Discovery in Mobile Apps. 2018.

[98] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. arXiv preprint arXiv:1812.00910*, 2018.*

[99] *T. Nishio and R. Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge.* ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019.*

[100] *Open Signal Inc. Mobile Analytics and Insights, June 2011.*

[101] *E. Pan, J. Ren, M. Lindorfer, C. Wilson, and D. Choffnes. Panoptispy: Characterizing audio and video exfiltration from android applications.* Proceedings on Privacy Enhancing Technologies*, 2018(4):33–50, 2018.*

[102] *G. I. Parisi and V. Lomonaco. Online continual learning on sequences. In* Recent Trends in Learning From Data*, pages 197–221. Springer, 2020.*

[103] *F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research*, 12:2825–2830, 2011.*

[104] *C. Phillips, M. Ton, D. Sicker, and D. Grunwald. Practical radio environment mapping with geostatistics.* Proc. of the IEEE DYSPAN '12*, pages 422–433, Oct. 2012.*

[105] *A. Pyrgelis, C. Troncoso, and E. D. Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In* 25th Network and Distributed System Security Symposium (NDSS)*, 2018.*

[106] *A. Rao, A. M. Kakhki, A. Razaghpanah, A. Tang, S. Wang, J. Sherry, P. Gill, A. Krishnamurthy, A. Legout, A. Mislove, and D. Choffnes. Using the Middle to Meddle with Mobile. Technical report, Northeastern University, Dec. 2013.*

[107] *A. Rao, F. Schaub, N. Sadeh, A. Acquisti, and R. Kang. Expecting the Unexpected: Understanding Mismatched Privacy Expectations Online.* the Proceedings of the Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, (Soups):77–96, 2016.*

[108] *A. Ray, S. Deb, and P. Monogioudis. Localization of lte measurement records with missing information. In* Proc. of the IEEE INFOCOM '16*, Apr. 2016.*

[109] *A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. 2018.*

[110] *A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. 2018.*

[111] *A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson. Haystack: A multi-purpose mobile vantage point in user space. 2015.*

[112] *A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson. Haystack: In Situ Mobile Traffic Analysis in User Space.* arXiv:1510.01419*, Oct. 2015.*

[113] *J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez. Bug Fixes, Improvements, ... and Privacy Leaks A Longitudinal Study of PII Leaks Across Android App Versions. (February), 2018.*

[114] *J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez. Bug fixes, improvements,... and privacy leaks. 2018.*

[115] *J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. Choffnes. Recon: Revealing and controlling pii leaks in mobile network traffic. In* Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 361–374, New York, NY, USA, 2016. ACM.*

[116] *M. S. Riazi, B. D. Rouhani, and F. Koushanfar. Deep learning on private data.* IEEE Security and Privacy Magazine*, 2018.*

[117] *S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated subgradient solver for svm.* Mathematical Programming*, 127(1):3–30, Mar 2011.*

[118] *N. Shoham, T. Avidor, A. Keren, N. Israel, D. Benditkis, L. Mor-Yosef, and I. Zeitak. Overcoming forgetting in federated learning on non-iid data.* arXiv preprint arXiv:1910.07796*, 2019.*

[119] *R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In* Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1310–1321, New York, NY, USA, 2015. ACM.*

[120] *A. Shuba, E. Bakopoulou, and A. Markopoulou. Privacy Leak Classification on Mobile Devices. In* 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2018.*

[121] *A. Shuba, E. Bakopoulou, M. A. Mehrabadi, H. Le, D. Choffnes, and A. Markopoulou. Antshield: On-device detection of personal information exposure.* arXiv preprint arXiv:1803.01261*, 2018.*

[122] *A. Shuba, A. Le, E. Alimpertis, M. Gjoka, and A. Markopoulou. Antmonitor: System and applications.* arXiv preprint arXiv:1611.04268*, 2016.*

[123] *A. Shuba and A. Markopoulou. ?NoMoATS: Towards Automatic Detection of Mobile Tracking?* Proceedings on Privacy Enhancing Technologies*, 2020(4), 2020.*

[124] *A. Shuba, A. Markopoulou, and Z. Shafiq. NoMoAds: Effective and Efficient Cross-App Mobile Ad-Blocking.* Proceedings on Privacy Enhancing Technologies*, 2018(4), 2018.*

[125] *Y. Song and U. Hengartner. PrivacyGuard: A VPN-based Platform to Detect Information Leakage on Android Devices. In* Proc. of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, Oct. 2015.*

[126] *G. Srivastava, S. Chitkara, K. Ku, S. K. Sahoo, M. Fredrikson, J. I. Hong, and Y. Agarwal. Privacyproxy: Leveraging crowdsourcing and in situ traffic analysis to detect and mitigate information leakage.* CoRR*, abs/1708.06384, 2017.*

[127] *L. Tian, S. Li, J. Ahn, D. Chu, R. Han, Q. Lv, and S. Mishra. Understanding user behavior at scale in a mobile video chat application.* Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp '13*, page 647, 2013.*

[128] *S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, and R. Zhang. A hybrid approach to privacy-preserving federated learning.* arXiv preprint arXiv:1812.03224*, 2018.*

[129] *Tutela Technologies. Manage your Mobile Experience.* `http://tutelatechnologies.com/index.html`*.*

[130] *N. Vallina-Rodriguez, A. Auçinas, M. Almeida, Y. Grunenberger, K. Papagiannaki, and J. Crowcroft. RILAnalyzer: A Comprehensive 3G Monitor on Your Phone. In* Proc. of IMC*, Barcelona, Spain, Oct. 2013.*

[131] *N. Vallina-Rodriguez, S. Sundaresan, A. Razaghpanah, R. Nithyanand, M. Allman, C. Kreibich, and P. Gill. Tracking the trackers: Towards understanding the mobile advertising and tracking ecosystem.* arXiv preprint arXiv:1609.07190*, 2016.*

[132] *M. Van Kleek, I. Liccardi, R. Binns, J. Zhao, D. J. Weitzner, and N. Shadbolt. Better the devil you know: Exposing the data sharing practices of smartphone apps. In* Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5208–5220. ACM, 2017.*

[133] *C. Wang, Y. Yang, and P. Zhou. Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity.* IEEE Transactions on Parallel and Distributed Systems*, 32(2):394–410, 2020.*

[134] *J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang. Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach. In* IEEE INFOCOM '17*, pages 1–9, May 2017.*

[135] *N. Wang, B. Zhang, B. Liu, and H. Jin. Investigating Effects of Control and Ads Awareness on Android Users' Privacy Behaviors and Perceptions. In* Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 373–382. ACM, 2015.*

[136] *Y. Wang and D. Chu. EarlyBird : Mobile Prefetching of Social Network Feeds via Content Preference Mining and Usage Pattern Analysis Categories and Subject Descriptors.* MobiHoc*, pages 67–76, 2015.*

[137] *Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning.* arXiv preprint arXiv:1812.00535*, 2018.*

[138] *W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu. A framework for evaluating gradient leakage attacks in federated learning.* arXiv preprint arXiv:2004.10397*, 2020.*

[139] *X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos. ProfileDroid: Multi-Layer Profiling of Android Applications. In* Proc. of the 18th ACM annual int. conf. on Mobile computing and networking*, Istanbul, Turkey, Aug. 2012.*

[140] *Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In* Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 329–344, New York, NY, USA, 2011. ACM.*

[141] *J. Yang, A. Varshavsky, H. Liu, Y. Chen, and M. Gruteser. Accuracy characterization of cell tower localization. In* Proc. of the ACM UbiComp '10*, pages 223–226, 2010.*

[142] *X. Yao and L. Sun. Continual local training for better initialization of federated models. In* 2020 IEEE International Conference on Image Processing (ICIP)*, pages 1736–1740. IEEE, 2020.*

[143] *J. Yoon, W. Jeong, G. Lee, E. Yang, and S. J. Hwang. Federated continual learning with adaptive parameter communication.* arXiv preprint arXiv:2003.03196*, 2020.*

[144] *Z. Yun and M. F. Iskander. Ray tracing for radio propagation modeling: Principles and applications.* IEEE Access*, 3:1089–1100, 2015.*

[145] *C. Zhang and P. Patras. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In* Proc. of the 18th ACM MobiHoc*, Mobihoc '18, pages 231–240. ACM, 2018.*

[146] *B. Zhao, K. R. Mopuri, and H. Bilen. idlg: Improved deep leakage from gradients.* arXiv preprint arXiv:2001.02610*, 2020.*

[147] *L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors,* Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.*

[148] *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.*