FedPacket: A Federated Learning Approach to Mobile Packet Classification

Evita Bakopoulou, *Student Member, IEEE*, Bálint Tillman, and Athina Markopoulou, *Fellow, IEEE*

Abstract—In order to improve mobile data transparency, various approaches have been proposed to inspect network traffic generated by mobile devices and detect exposure of personally identifiable information (PII), ad requests, etc. State-of-the-art approaches use features extracted from HTTP packets and train 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. However, training datasets from network traffic collected on user devices may contain sensitive information that users may not want to upload. In this paper, 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 also discuss privacy limitations and mitigation approaches.

Index Terms—Federated learning, machine learning, mobile devices, packet classification, privacy

1 Introduction

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. Efforts to increase online data transparency include landmark privacy laws (such as GDPR [1] and CCPA [2]) as well as technical approaches (*e.g.,* permissions [3], static and dynamic analysis [4], [5], and network-based approaches [6], [7], [8], [9]).

In this paper, 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. 2. 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 [6] and AntMonitor [8]) performed deep packet inspection (DPI) and string matching to find PII in a network packet. Mobile browsers [10] use manually-curated filter-lists [11] to match URI and other information and block ad requests.

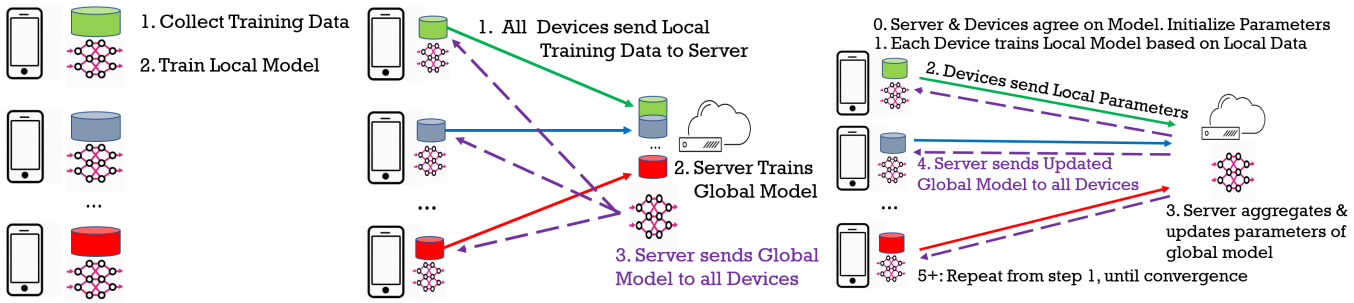
Machine learning approaches have been proposed to predict PII [7], [9], [12], ad requests [13] or tracking [14] in outgoing packets, based on features extracted from HTTP requests. These classifiers are trained using labeled packet traces, obtained either through manual/automatic testing of apps, or by devices labeling packets from real user activity and uploading them to a server. Fig. 1 depicts the spectrum of approaches for training such models. At one extreme, we have the the Local approach (Fig. 1(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. 1(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 paper, 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 [15]. An overview is depicted on Fig. 1(c). The main idea is that 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 paper, 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)-

- *All authors were with the University of California, Irvine when the work was conducted. Bálint Tillman is currently with Google, Mountain View. E-mail: {ebakopou, tillmanb, athina}@uci.edu.*

Manuscript received May 14, 2020; revised January 13, 2021.



(a) Local: share nothing (b) Centralized: share training data (c) Federated: share model parameters, not data

Fig. 1: Overview of general approaches to train machine learning models for packets from mobile devices.

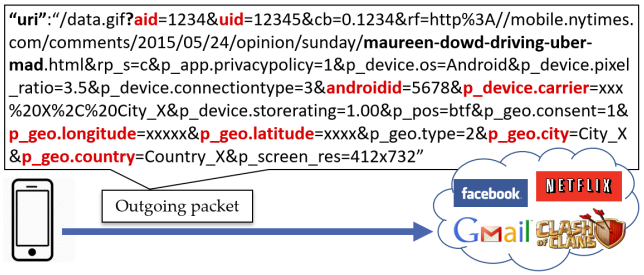


Fig. 2: Example of an outgoing HTTP packet, sent from an app on the mobile device to a remote server. The URI field alone reveals a lot of information, including various identifiers, referred domain, location, *etc.* that can be used for fingerprinting and tracking users.

based models, such as Deep Neural Networks (DNNs)) and the practice in network packet classification (previously relying on Decision Trees (DTs) [7], [9], [13], [14], 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 paper, we choose *Federated SVM* as the core of our FedPacket framework, as discussed in detail in Section 3.3.

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. 2 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 [7], [13] 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 [13], [16] and AntShield for PII exposures [12], [17]; 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 [18], [19], [20], [21], [22]. In Section 6, 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 [23], can provide effective protection against feature leakage, essentially by anonymizing the gradient updates.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 presents our methodology for FedPacket. Section 4 describes the datasets used for evaluation. Section 5 presents various scenarios and provides insights along the way. Section 6 presents privacy attacks and mitigation approaches specific to FedPacket. Section 7 concludes the paper and discusses future work. The Appendix provides additional results.

2 Related Work

There is increasing interest in understanding and controlling PII exposure and user tracking on mobile devices. Proposed approa-

ches include: permissions [3], static analysis [4], dynamic analysis [5], and network-based approaches [24], [25], [26]; our work falls within the latter which inspect mobile traffic for PII exposure, or other information (*i.e.,* tracking, malware, advertising). State-of-the-art tools include Haystack/Lumen [6], Antmonitor [8] 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 but it is orthogonal to our approach.

Ad and PII Detection via Machine Learning. There are many approaches based on manually curated blacklists [11], [27], [28] of domains on which they decide to block the whole packet destined to such domains or cookies from such domains [29]. Since blacklists are hard to maintain due to the ever-changing advertising ecosystem, additional graph analysis [26], or machine learning (ML) [13], [30] are used. NoMoAds [13] and NoMoATS [14] are state-of-the-art approaches for detecting Advertising and Tracking requests, respectively. They train classifiers, based on URL requests labeled by blacklists [11], to detect advertising and tracking; conversely, the classifiers trained this way can generalize, complement and enhance blacklists' manual curation. Recon [7] and AntShield [9] 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 [31], 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 [32], or tracking [14] detection.

Distributed & Federated Learning. The authors in [12] showed that systematic crowdsourcing where users collaborate with each other via data sharing helped to train better classifiers to detect PII exposures. However, it is assumed that users are willing to share their raw local data with a server and other users, which poses privacy risks. To leverage crowdsourcing in a more privacy-preserving way, we considered Distributed Learning. An early version was proposed in [33], 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, [33] had no averaging mechanism and the evaluation was limited. The paper that coined the term "Federated Learning" (FL) was introduced in [15], 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 [34], [35], [36], [37], [38]. In contrast to related work in the field that is using image classification or next word prediction via word/character embeddings [15], we focus on a problem where pre-trained word embeddings (*i.e.,* Word2Vec [39]) 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 [15]) are unnecessary. An earlier version of this paper, appeared on ArXiv [40].

Dealing with Heterogeneity. Dealing with system and data heterogeneity across different users, and the related notion of personalization, are open problems in FL [41]. FedProx [42] 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 [43], [44], [45], not applicable to FL due to the need of data sharing; and 2) regularization-based methods [46], [47], [48] 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 [49] 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. 5.4. 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 [46], or the need of a proxy dataset on the server [47]; these trade-offs are deferred to future work.

Privacy and Federated Learning. Several security and privacy attacks are known for machine learning systems; *e.g.,* [18], [50], [51], [52], [53] 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 [18], [19]. Examples of leakage include membership inference of data points [18], inference of certain property of train data [18] or reconstruction of train data [20], [21], [22], [51], [54], [55]. Reconstruction of local data usually requires either training auxiliary models (GANs) [51], [54] and/or having access to an auxiliary dataset [18], however, recent works [20], [21], [22], [55] introduced new methods for reconstructing local train 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) [23] or Differential Privacy [56] to offer privacy guarantees [57], [58], [59] or both [60]. In this paper, we consider such variants of FL, as orthogonal and out of scope. Our focus in this paper is on the FL framework itself: how to adapt and evaluate it specifically for mobile packet classification (as opposed to the image/text classification that is most commonly used for).

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. 6, we evaluate the sensitivity of the attack to various FL parameters.

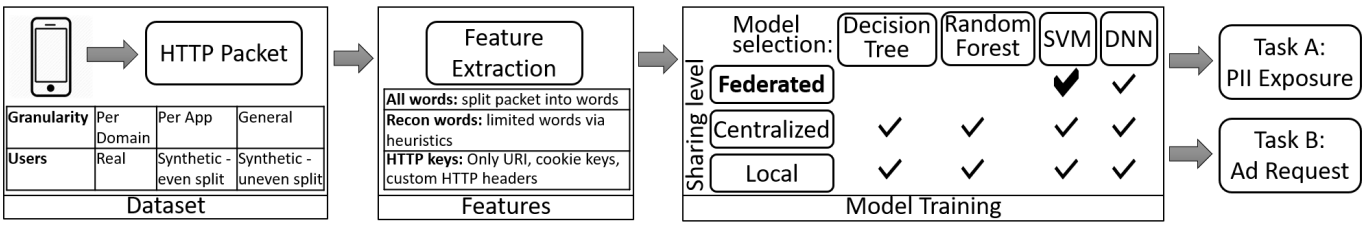


Fig. 3: Our pipeline for FedPacket. During *training*, the input is a packet trace with HTTP packets sent from mobile apps to remote destinations, labeled for the Tasks (PII Exposure, Ad Request); the output is an ML model, which is trained in a local, centralized or federated way. During *testing*, the input is an HTTP packet, and the output is a binary label (indicating the presence of PII or Ad request).

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 data in the context of mobile packet classification, as prior works focused on reconstruction of input images [20], [21], [22], [55] or text sentences [20]. The above data reconstruction attacks in case of federated packet classification are deferred to future work.

3 Methodology

3.1 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.[1] 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 [7], [9], [13] 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 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. 1(c) and described in Section 2) 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 3.2 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 3.3 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 [8], and labeling can be done using DPI [7], [9], [12], blacklists [11] or other tools [13]. Section 4 describes three real-world datasets used in this paper.

We focus on adapting and evaluating the FL framework specifically for mobile packet classification. An overview of our pipeline is provided on Fig. 3 and is described in the rest of this section.

3.2 HTTP Features

Feature extraction. We build on the Recon [7] approach which was used by follow-up classifiers [9], [12], [13]: every HTTP packet is split into words based on delimiters; the resulting words include keys and values from all HTTP packet headers. Fig. 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

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

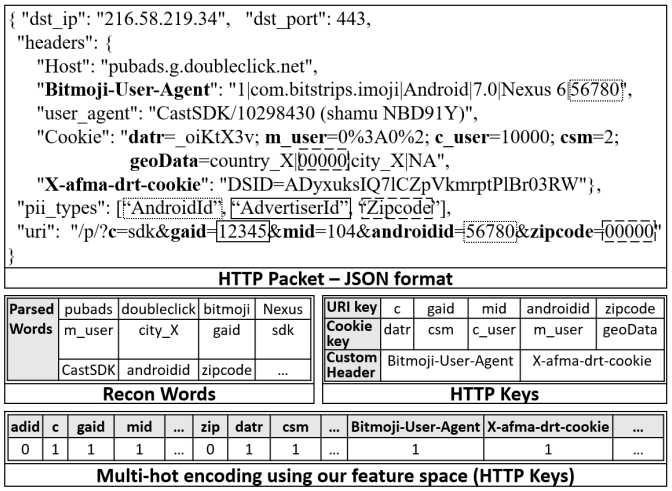


Fig. 4: An HTTP packet in JSON, where Android Id, Advertiser Id and zipcode are sent by Bitmoji app to an ad server (doubleclick.net) and thus, it would be labeled positive both for PII exposure and Ad request. Our **HTTP Keys features** are highlighted in bold: these keys are defined by the HTTP protocol 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".

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 [7], [9], [12], [13]. 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 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 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 [61] 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.

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 paper, 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. 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.

3.3 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. 1(c)) and compare it to its Centralized (Fig. 1(b)) and Local versions (Fig. 1(a)). The choices we evaluate across these two dimensions (classification model and degree of collaboration among users) are summarized under "Model Training" on Fig. 3.

Selecting SVM as the Classification Model. State-of-the-art classifiers for mobile packets have trained DTs [7], [12], [13] 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 [9], [12], [13] – 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 paper, 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 (see Appendix A.1).

Federated averaging uses models based on SGD, primarily DNNs [15]. 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 paper, 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.

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

$$f(w, X, Y) = \sum_i l(w, x_i, y_i) + \alpha||w||^2, \quad (1)$$

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 [62] 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 [15].

Algorithm 1 shows the Federated SVM algorithm: we apply the SVM-based gradient updates to the Federated Averaging [15]. Federated SVM trains an SVM model distributively over K clients (corresponding to mobile devices), where C fraction of the clients

Algorithm 1: 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 η 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*

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 [15], where users are chosen at random and uniformly. However, there is ongoing research on how client selection affects convergence [41]. The authors in [63] proposed selecting clients based on probabilities proportionate to their train data size. In this work, we show in Sec. 5.4 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) [41], [64].

The Federated SGD algorithm is a special case of Federated Averaging for $C = 1$, $E = 1$, $B = \infty$ [15] (*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. [15] 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. 1.

- Local models are trained on data available on each device, similarly to prior works [7], [9], [12], [13]. 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 [7], [9], [12], [13]. 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,

				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/(synthetic)	15,351	4,866/4,427	12,511/6,743	2,580	216	204	3,342	2,334/2,123	366
AntShield	297/(synthetic)	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 1: Summary of datasets: total features (our HTTP Keys vs. prior work), total packets, users, visited domains and classification labels.

which then aggregates, updates the global model and pushes it to all devices; the process repeats until convergence.

4 Datasets Description

We use three real-world datasets, summarized on Table 1, to evaluate the performance of our federated approach, w.r.t. two packet classification tasks: PII exposures and Ad requests.

NoMoAds dataset [13], [16]. This dataset 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 contains state-of-the-art labels for advertising and it was generated in 2017 via manual testing (interacting with each app for 5 minutes) with test accounts (no human subjects were involved).

AntShield dataset [9], [12], [17]. This dataset contains HTTP(S) packets labeled to indicate if they contain a PII exposure or not, (similarly to NoMoAds) but it is richer as it contains more apps. 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.

In-house Datasets with real users. We collected this dataset in-house from 10 real users in 2015 who contributed their packet traces for a period of 7 months[2]. The packet traces were collected by running Antmonitor [8] 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. An IRB was put in place only for restricting what to record, not the usage. In order 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. To evaluate FL, we consider the two most popular apps across all users: Chrome (8 users) and Facebook (10 users) and treat them as separate datasets.

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 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 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 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 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

2. The study was IRB approved in our institution.

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 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.

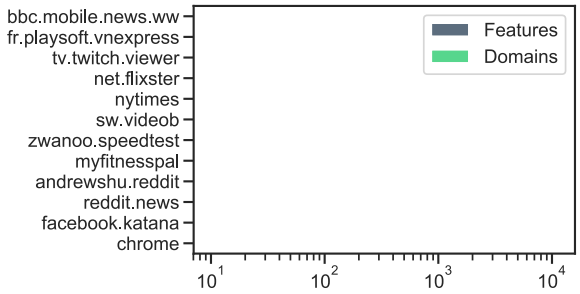


Fig. 5: Number of features and domains for the top 12 apps with most features from our in-house dataset. The number of features correlates with number of visited domains.

of each app, *e.g.,* the duration (in terms of packets), user behavior (in terms of domains visited). Fig. 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 paper, 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 global feature space is known to the server and all users in advance when they are initializing their corresponding models.

5 Results

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 3 lists the possible options for evaluation based on our pipeline defined in Section 3. 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

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 3: Parameters of the Evaluation Setup.

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 [65] 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. 5.4, that training on a subset of users can provide good classifiers for all users.

5.1 Scenario 1: Centralized Models

Setup 1. In this experiment, we use the following setup from Table 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 paper.

Results 1a: HTTP Keys vs. Recon Words features. In Table 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 prediction, since our feature space does not include information from domains that is important for this task as shown in [13]. Prior work [7], [9], [12], [13] uses domain information in addition to other potentially sensitive features, and achieves

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: **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).

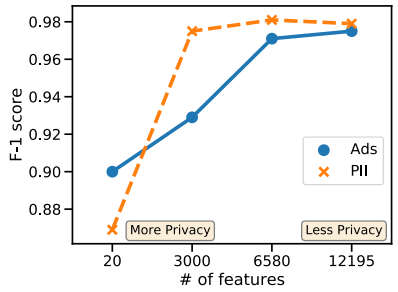


Fig. 6: Feature explosion and privacy vs. utility for top 20 important HTTP Keys (from Fig. 8), HTTP Keys (3,000), Recon Words (6,580), All Words (12,195) depicting Table 4.

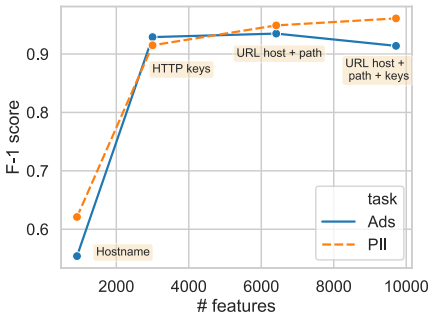


Fig. 7: Utility vs. privacy and feature explosion when using features from different parts of the URI compared to HTTP Keys feature space.

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. 6 depicts Table 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. 8) and evaluate the F1 score. F1 score decreases but remains above 0.9 for HTTP keys, considering 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.

Privacy-utility tradeoff. We show in Fig. 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 are not sufficient to achieve a high F1 score. Adding more information

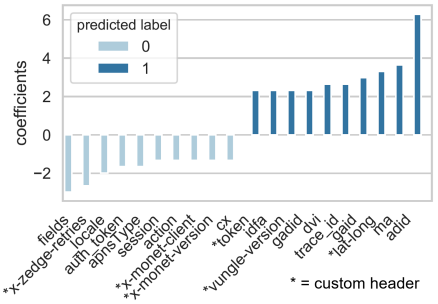


Fig. 8: Top 10 negative and positive coefficients and the corresponding features obtained from Centralized SVM with HTTP Keys for PII.

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 6. 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 compares SVM with SGD to state-of-the-art baseline models, such as Decision Tree (DT) and Random Forest (RF) (used in prior work [7], [9], [13]) 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. In Appendix A.1, we show how SVM can be as interpretable as DT by performing knowledge transfer between the two. 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, 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 $[10^{-5}, 1]$ 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. 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 [66], [67] throughout this paper unless mentioned otherwise. We discuss how

3. An example of sensitive identifiers is shown in Fig. 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.

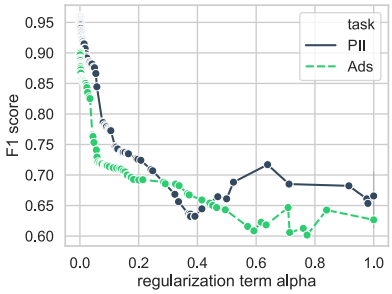


Fig. 9: Regularization term α and its effect on F1 score for Centralized SVM with SGD for both tasks.

Trained on	Tested on	Uneven split F1 score Ads	Uneven split F1 score PII	Even split F1 score Ads	Even split F1 score 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 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.

C	Uneven split B = ∞	Uneven split B = 10	Even split B = ∞	Even split 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 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.

η affects convergence in the federated setting in Sec. 5.4 and Fig. 13 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 [15], as the question of personalization in FL [42] is still an open problem and out-of-scope in this paper.

5.2 Scenario 2: NoMoAds for PII, Ad Request

Setup 2a. We use the following setup from Table 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 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.

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 [15]. 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.

Results 2b: Impact of Federated parameters. Table 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 [15] which used more complex models.

5.3 Scenario 3: AntShield for PII Prediction

Setup 3a. We use the following setup from Table 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 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

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 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 $C = 1.0, B = 10, E = 5$.

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. 10 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.

5.4 Scenario 4: In-house Datasets for PII Prediction

Setup 4. We use the following setup from Table 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. 11 shows the distribution of Chrome and Facebook packets (including labels) present across the 10 real users in our in-house dataset.

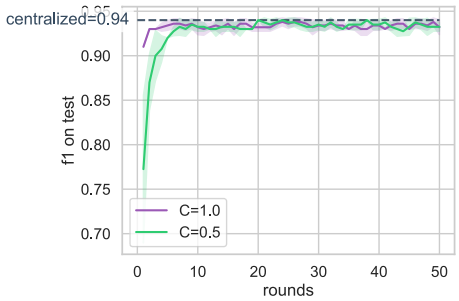


Fig. 10: **Results 3b.** Convergence of AntShield with 100 synthetic users with uneven spit when $B = 10$, $E = 1$, and varied C.

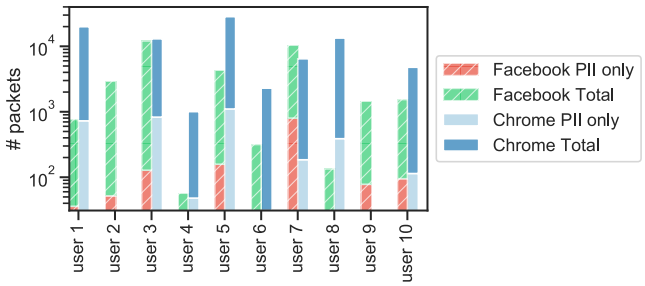


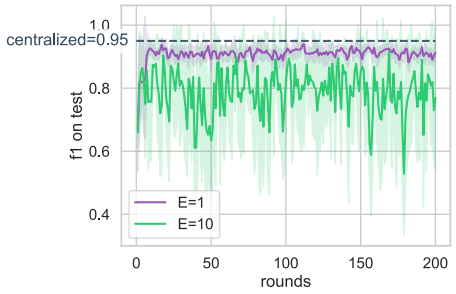
Fig. 11: Distribution of packets for Facebook and Chrome.

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]

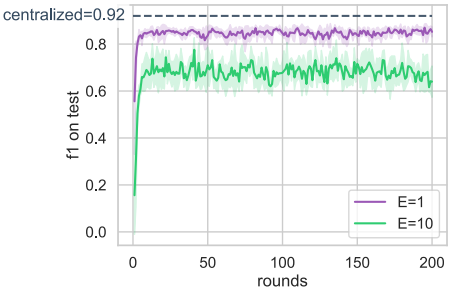
TABLE 8: **Results 4b.** We report the average [min, max] R communication rounds required to reach a target F1 score (0.94 for Facebook, 0.84 for Chrome). We vary the batch size (B) and local epochs (E) to evaluate their impact on R, with $C = 0.5$. If the target F1 score is not reached within 800 rounds over 5 runs, we assume that it does not converge.

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 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 [15] showed the opposite effect: increasing E decreases R; however, they train more complex models (DNNs) that do not overfit for those E values. Moreover, we observe that B does not significantly affect the number of rounds. In contrast, E plays an important role in the model's



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



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

Fig. 12: **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.

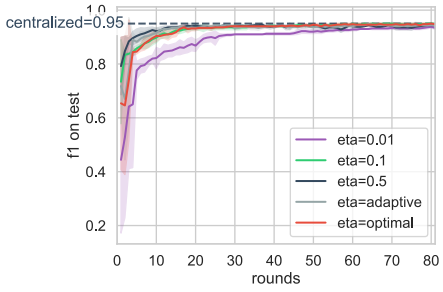


Fig. 13: **Results 4d.** Convergence of F1 score over R rounds for Facebook with $C = 0.5, B = 10, E = 1$ and learning rate η varied.

convergence, which is explored next.

Results 4c: Convergence of Federated models. Fig. 12 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 paper we used the "optimal" learning rate from scikit-learn [68] 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 [66], [67] as $t_0 = \frac{1}{\eta_0 * a}$ where a is the regularization term. Fig. 13 shows the impact of the η on the Federated model's convergence for Facebook data with $C = 0.5$, $B = 10$, $E = 1$ similarly to Fig. 12a. We observe that only the smaller learning rate (η=0.01)

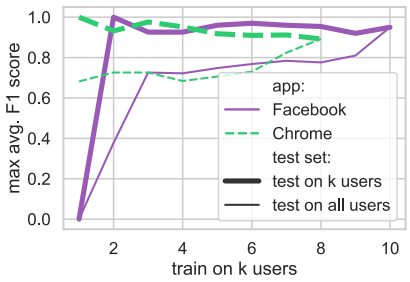


Fig. 14: **Results 4e.** Benefit of crowdsourcing with k users for Chrome and Facebook. The average F1 score is shown for all users' test data (thin line) and for test data of k users (bold line).

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 [41], [42], [46], [63], [69]. Our model has a linear kernel and the loss function is convex which guarantees convergence for both IID and non-IID data following the original FL paper [15]'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 [15] did not provide a mathematical analysis of convergence. The authors in [63] 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 [15] and [63].[4]

Results 4e: Benefit of Crowdsourcing. We ask the question: how many users need to collaborate to train a global model in order to get most of the predictive power? Fig. 14 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,

4. 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 [42], [46], [47], [48] can be considered in order to speed up further the convergence and are deferred to future work.

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 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.)

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).

Table 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 (see Appendix A.3).

5.5 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 [15]. 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 [63], 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.

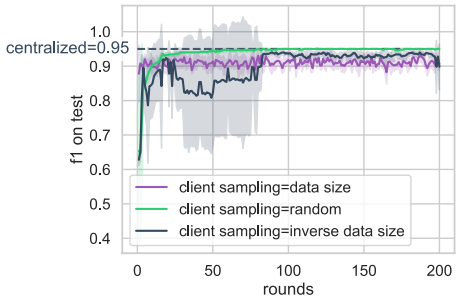


Fig. 15: **Results 5a.** Convergence of F1 score over R rounds vs. various client selection strategies for Facebook (non-IID) data with $C = 0.5, B = 10, E = 1$.

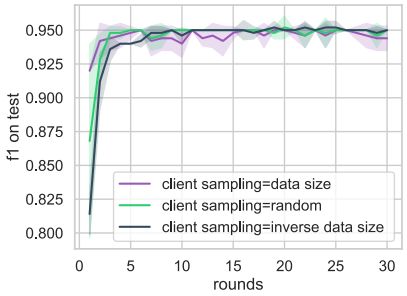
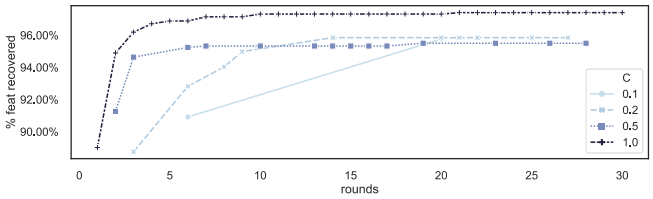


Fig. 16: **Results 5b.** Convergence of F1 score over R rounds vs. various client selection strategies with NoMoAds 20 synthetic (non-IID) users when predicting PII with $C = 0.5, B = 10, E = 1$.

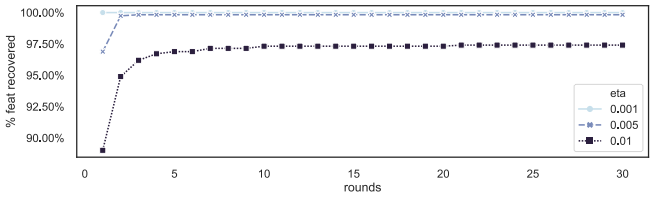
Setup 5a. We use the following setup from Table 3: *Dataset:* In-house Facebook. *Users:* 10 real users. *Classifier Granularity:* Per App. *Models:* Federated SVM. *Tasks:* PII exposure. The goal is to evaluate the convergence of the Federated model when different client selection strategies are in 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. 11 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. 15 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. 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.

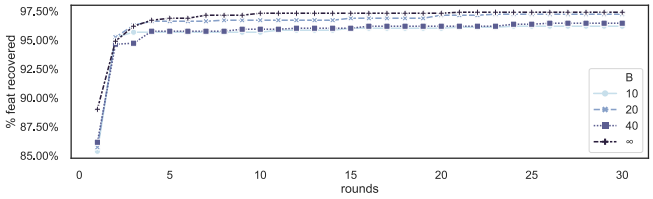
Setup 5b. We use a similar setup to Setup 5a, but with NoMoAds 20 uneven synthetic users. *Classifier Granularity:* General. *Models:* Federated SVM. *Tasks:* PII exposure, Ad request. The goal is to evaluate the convergence of the Federated model when



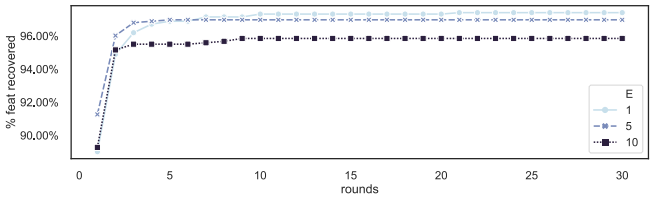
(a) % features recovered vs. C (% clients selected in a round) with $B = \infty$, $E = 1$.



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

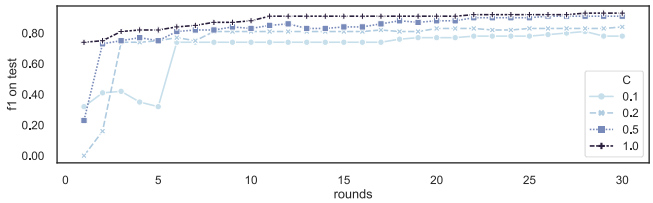


(c) % features recovered vs. batch size B with $E = 1$, $\eta = 0.01$.

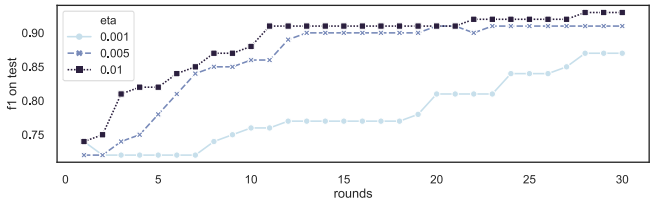


(d) % features recovered vs. local epochs E with $B = \infty$, $\eta = 0.01$.

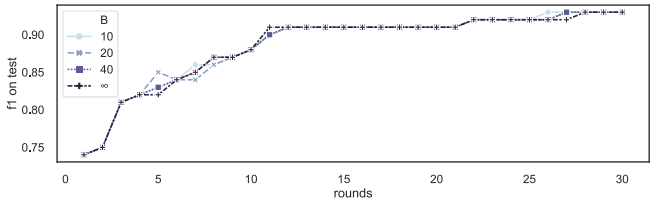
Fig. 17: Evaluating the success of our privacy attack in terms of features recovered (%) when we vary the FL parameters.



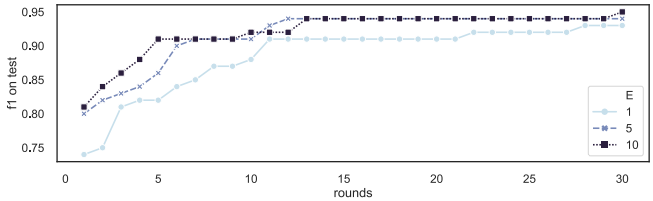
(a) F1 score vs. C (% clients selected in a round) with $B = \infty$, $E = 1$.



(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 $B = \infty$, $\eta = 0.01$.

Fig. 18: Convergence in terms of F1 score for selected FL parameters corresponding to privacy attack.

different client selection strategies are in place for IID clients.

Results 5b: IID clients. Fig. 16 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.

6 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.[5] However, federated learning, and more generally distributed learning, has its own inherent vulnerabilities to inference attacks based on observed updates [18], [19], [20], [21], [22]. 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.

5. 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.

6.1 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 [56] or Secure Aggregation [23] 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 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. 17 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. 18 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 participating clients, as shown in Fig. 17a, since the target user might not participate in every round due to random selection of clients in each round. Fig. 18a shows that selecting $C = 1.0$ speeds up 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. 17b 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. 13. We demonstrate this privacy-utility trade-off in Figures 17b and 18b: 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. 17c 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. 18c. Finally, Fig. 17d 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. 18d 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 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.

	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 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.

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,[6] 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 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) [70] filter list and the module AdblockRules from AdblockPlus [10] to label ATS domains. Figures 26 and 27 in the Appendix show the results. Some examples of non-ATS domains are the following: europa.eu, facebook.net, vox.com, while some ATS

6. The results for all 105 common domains, are presented in Fig. 26 and a zoomed-in version with top 30 (in alphabetical order) domains in Fig. 27. Both figures are moved to the Appendix due to lack of space.

domains: doubleclick.net, google-analytics.com, openx.net. Table 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).

6.2 Mitigation via Aggregation

There are two families of defense mechanisms that are usually applied on top of federated learning: differential privacy [56] (and other types of noise, including federated GANs [71]) and Secure Aggregation (SA) [23]. 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 [23] 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, 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 $I(user, feature)$). Due to lack of space, we defer details to the appendix, and outline the main idea.

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.

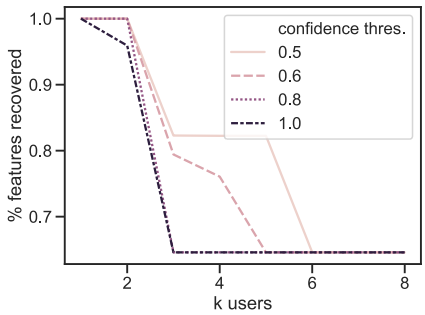


Fig. 19: Evaluating Attack Algorithm 6c, with secure aggregation on. We report the percent of the target user's recovered features, for varying k (participating users in a round) and confidence thresholds.

The 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. 19 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 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.

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, 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%.

7 Conclusion and Future Directions

This paper proposes for the first time, 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

proposed a reduced feature space (HTTP Keys), which limits the sensitive information shared by users. Then, we showed that SVM with SGD performs similarly to Decision Trees used by state-of-the-art [7], [9], [12], [13], in terms of F1 score as well as interpretability. We also showed 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 demonstrated an attack by an honest-but-curious server that can infer features and browsing history, and demonstrated that simple existing add-on mechanisms can provide significant levels of protection. The code for our experiments, will be released along with the public datasets.

In future work, we will consider additional privacy protections on top of our framework, beyond secure aggregation, *e.g.,* differential privacy (DP) [57], [58], [59], selecting a subset of gradient updates, compression of gradients or federated GANS [71]. 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 [14] or fingerprinting [32]), and beyond mobile devices (*e.g.,* for network traffic generated by different IoT devices).

ACKNOWLEDGMENTS

This work has been supported by NSF Awards 1900654, 1649372 and 1526736. E. Bakopoulou and B. Tillman have been supported by H. Samueli and Networked Systems Fellowships. E. Bakopoulou has also received a Broadcom Foundation Fellowship. We thank A. Shuba and M. Gjoka, former members of our group, for the NoMoAds and in-house datasets, used for evaluation.

REFERENCES

[1] "EU General Data Protection Regulation (GDPR)," https://eugdpr.org.
[2] "California consumer privacy act (ccpa)," https://oag.ca.gov/privacy/ccpa.
[3] 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*, vol. 2018, no. 4, pp. 33–50, 2018.
[4] 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*.
[5] 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)*, vol. 32, no. 2, p. 5, 2014.
[6] 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.
[7] 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*, ser. MobiSys '16. New York, NY, USA: ACM, 2016, pp. 361–374. [Online]. Available: http://doi.acm.org/10.1145/2906388.2906392
[8] A. Shuba, A. Le, E. Alimpertis, M. Gjoka, and A. Markopoulou, "Antmonitor: System and applications," *arXiv preprint arXiv:1611.04268*, 2016.
[9] 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.
[10] "Adblock browser," https://adblockbrowser.org.
[11] "EasyList," https://easylist.to/.
[12] 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.
[13] A. Shuba, A. Markopoulou, and Z. Shafiq, "NoMoAds: Effective and Efficient Cross-App Mobile Ad-Blocking," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 4, 2018.
[14] A. Shuba and A. Markopoulou, "Nomoats: Towards automatic detection of mobile tracking," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 2, pp. 45–66, 2020.
[15] 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. [Online]. Available: http://arxiv.org/abs/1602.05629
[16] "NoMoAds Dataset," https://athinagroup.eng.uci.edu/projects/nomoads/data/.
[17] "AntShield Dataset," https://athinagroup.eng.uci.edu/projects/antmonitor/antshield-dataset/.
[18] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning." IEEE, 2019.
[19] 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.
[20] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 747–14 756.
[21] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients–how easy is it to break privacy in federated learning?" *arXiv preprint arXiv:2003.14053*, 2020.
[22] 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.
[23] K. Bonawitz, V. Ivanov, B. Kreuter, and A. Marcedone, "Practical Secure Aggregation for Privacy Preserving Machine Learning," *Eprint.Iacr.Org*. [Online]. Available: https://eprint.iacr.org/2017/281.pdf
[24] 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.
[25] J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez, "Bug fixes, improvements,... and privacy leaks," 2018.
[26] 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.
[27] "hphosts," https://hosts-file.net/ad_servers.txt.
[28] "Adaway," https://adaway.org/hosts.txt.
[29] 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*. ACM, 2017, pp. 259–262.
[30] D. Gugelmann, M. Happe, B. Ager, and V. Lenders, "An automated approach for complementing ad blockers' blacklists," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 282–298, 2015.
[31] 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*, vol. abs/1708.06384, 2017. [Online]. Available: http://arxiv.org/abs/1708.06384
[32] 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)*, 2017, pp. 1391–1408.
[33] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 1310–1321. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813687
[34] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.
[35] S. Caldas, J. Konecnỳ, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *CoRR*, vol. abs/1812.07210, 2018. [Online]. Available: http://arxiv.org/abs/1812.07210
[36] N. Guha, A. Talwlkar, and V. Smith, "One-shot federated learning," *arXiv preprint arXiv:1902.11175*, 2019.
[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] 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.

[39] "Pretrained word2vec," https://code.google.com/archive/p/word2vec/.

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

[41] 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.

[42] 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*, vol. 2, pp. 429–450, 2020.

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

[44] 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*, 2019, pp. 11 849–11 860.

[45] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 816–11 825.

[46] 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," 2019.

[47] X. Yao and L. Sun, "Continual local training for better initialization of federated models," *arXiv preprint arXiv:2005.12657*, 2020.

[48] 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.

[49] 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*, vol. 114, no. 13, pp. 3521–3526, 2017.

[50] 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)*, Aug 2017, pp. 1–6.

[51] 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*.

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

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

[54] 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.

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

[56] C. Dwork, "Differential privacy," *Encyclopedia of Cryptography and Security*.

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

[58] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, vol. abs/1712.07557, 2017. [Online]. Available: http://arxiv.org/abs/1712.07557

[59] 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.

[60] 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.

[61] "Permanent Message Header Field Names," www.iana.org/assignments/message-headers/message-headers.xhtml#perm-headers, 2019.

[62] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for svm," *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

[63] 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.

[64] 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. [Online]. Available: http://dx.doi.org/10.1109/ICC.2019.8761315

[65] "Ui/application exerciser monkey," https://developer.android.com/studio/test/monkey.

[66] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 161–168. [Online]. Available: http://papers.nips.cc/paper/3323-the-tradeoffs-of-large-scale-learning.pdf

[67] L. Bottou, "Stochastic gradient descent (v.2)," https://leon.bottou.org/projects/sgd.

[68] 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*, vol. 12, pp. 2825–2830, 2011.

[69] C. Wang, Y. Yang, and P. Zhou, "Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity," 2020.

[70] "Mother of all adblockers," http://adblock.mahakala.is.

[71] S. Augenstein, H. B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews, and B. A. y Arcas, "Generative models for effective ml on private, decentralized datasets," 2020.

Evita Bakopoulou received her B.Sc. and M.Sc. degrees in Computer Science from Athens University of Economics and Business, Greece, in 2014 and 2016, respectively. Currently, she is a Ph.D. student in the Networked Systems Program at UC Irvine, working with Prof. Athina Markopoulou. She was a Summer Intern with Bell Labs (2017), Oath/Verizon Digital Media Services (2018) and Google (2020). Her research interests are primarily in the area of machine learning, and privacy.

Bálint Tillman received the B.Sc degree in Business Information Technology from the Corvinus University of Budapest, Hungary, in 2008; the M.Sc degree in Software Development and Technology from IT University of Copenhagen, Denmark, in 2012; and the Ph.D. degree in Networked Systems Program from UC Irvine in 2019. He was a Summer Intern with Google from 2016 to 2018, and he is currently with Google, Mountain View. He received the Henry Samueli Fellowship for Networked Systems 2015-2016. His research interests include graph algorithms and machine learning in computer networks.

Athina Markopoulou (S'98-M'02-SM'13-F'21) received the Diploma degree in Electrical and Computer Engineering from the National Technical University of Athens, Greece, in 1996, and the M.S. and Ph.D. degrees in Electrical Engineering from Stanford University in 1998 and 2003, respectively. She joined the faculty of EECS Department at UC Irvine in 2006, where she is currently a Professor and Chair. She has held short-term positions at SprintLabs, Arista Networks, and IT University of Copenhagen. She has received the NSF CAREER award in 2008, the HSSoE Faculty Midcareer Research Award in 2014, the OCEC Educator Award in 2017, and a UCI Chancellor's Fellowship in 2019. She has served as an Associate Editor for IEEE/ACM Trans. on Networking and for ACM Computer Communications Review, as the General Co-Chair for ACM CoNEXT 2016, as TPC Co-Chair for ACM SIGMETRICS 2020 and NetCod 2012. She is also a Senior Member of the ACM. Her research interests are in networking, mobile and IoT, privacy, network measurement, social networks, and network coding.