Electronic Communications of the EASST
Volume 080 (2021)

Conference on Networked Systems 2021
(NetSys 2021)

Browser Fingerprinting: How to Protect Machine Learning Models and Data with Differential Privacy?

Katharina Dietz, Michael Mühlhauser, Michael Seufert,
Nicholas Gray, Tobias Hoßfeld, Dominik Herrmann

16 pages

Browser Fingerprinting: How to Protect Machine Learning Models and Data with Differential Privacy?

Katharina Dietz[1], Michael Mühlhauser[2], Michael Seufert[1],
Nicholas Gray[1], Tobias Hoßfeld[1], Dominik Herrmann[2]

[1]firstname.lastname@uni-wuerzburg.de
University of Würzburg

[2]firstname.lastname@uni-bamberg.de
University of Bamberg

Abstract: As modern communication networks grow more and more complex, manually maintaining an overview of deployed soft- and hardware is challenging. Mechanisms such as fingerprinting are utilized to automatically extract information from ongoing network traffic and map this to a specific device or application, e. g., a browser. Active approaches directly interfere with the traffic and impose security risks or are simply infeasible. Therefore, passive approaches are employed, which only monitor traffic but require a well-designed feature set since less information is available. However, even these passive approaches impose privacy risks. Browser identification from encrypted traffic may lead to data leakage, e. g., the browser history of users. We propose a passive browser fingerprinting method based on explainable features and evaluate two privacy protection mechanisms, namely differentially private classifiers and differentially private data generation. With a differentially private Random Decision Forest, we achieve an accuracy of 0.877. If we train a non-private Random Forest on differentially private synthetic data, we reach an accuracy up to 0.887, showing a reasonable trade-off between utility and privacy.

Keywords: Browser Fingerprinting, Differential Privacy, Privacy-Preserving Machine Learning, Synthetic Data Generation

1 Introduction

Due to the rising complexity of enterprise networks, it is important to keep track of the network inventory to match the deployed software with Common Vulnerabilities and Exposures (CVE) databases, detecting outdated or unwanted browsers and applications. Additionally, enterprise policies might require employees to use a specific browser. To retrieve information about utilized browsers for possibly enforcing such rules, browser fingerprinting is applied. Fingerprinting describes the extraction of characteristic values or patterns by either *actively* injecting requests and analyzing responses, e. g., via JavaScript, or by *passively* observing the network traffic. The extracted information is then matched with fingerprint databases or used as input for Machine Learning (ML)-based approaches.

The aforementioned active fingerprinting approaches obtain information via dedicated browser APIs, such as a list of installed fonts and plugins [LBBA20], or hardware-level features [CLW17].

The disadvantage is that only the server hosting the website can collect the fingerprint [YC14]. If the adversary is not hosting the server but can only eavesdrop on the communication between the client and the server, active approaches are not feasible [YC14]. Furthermore, with the rise of traffic encryption, the content of packets cannot be accessed anymore. That is, commonly utilized features like the HTTP User-Agent, which contain information about the operating system (OS) and browser, cannot be extracted easily. We do not apply Deep Packet Inspection (DPI), as encrypted traffic needs to be decrypted via invasive methods, which raises privacy concerns.

Therefore, we propose a ML-based method to differentiate between two browsers solely based on packet headers; thus our *passive* approach works on encrypted traffic. However, metadata from packet headers alone could still compromise privacy. Yen et al. have shown that website fingerprinting is more effective if the browser is detected first [YHMR09]. While network monitoring is a legitimate interest in compliance with the General Data Processing Regulation (GDPR) [Eur21], e. g., to secure the network, reconstructing the browser history invokes a clear invasion of privacy. Thus, privacy needs to be protected in browser fingerprinting. This results in a trade-off between utility and privacy, as simultaneously providing privacy protection and enabling browser detection might be contradicting.

For that, we consider two different privacy protection mechanisms based on differential privacy [Dwo06]. Firstly, we use a differentially private Random Decision Forest (DPRF) to identify browsers [FI17]. Unlike non-private decision trees (DTs), where the generated tree leaks information, the output of the private ML algorithm preserves privacy [FI19]. Secondly, since this protection mechanism does not protect the raw data, we, therefore, evaluate the utility of differentially private synthetic data, i. e., we apply ML algorithms on private synthetic data. In this case, data is protected directly with QUAIL and PATE-CTGAN by creating a synthetic dataset that guarantees differential privacy [RLP+20]. The main contributions of this paper are:

1. We propose a ML-based method solely based on network traffic features to identify the used browser. The selected features for our ML model are obtained with domain knowledge and are explainable for network administrators. Our results show that Google Chrome and Mozilla Firefox can be detected with accuracy values above 0.98.

2. We incorporate two different privacy protections into browser fingerprinting. Firstly, we examine the effectiveness of a DPRF and analyze the trade-off between privacy and utility. Secondly, we create a synthetic dataset, which guarantees differential privacy, and evaluate its utility for browser fingerprinting. We show that accuracy values in both cases are still high even under the strict guarantees of differential privacy.

3. We publish the browser fingerprinting source code and our datasets [DM21]. That is, we release the initial dataset as well as our synthetic datasets. Moreover, we release the source code to generate the differentially private synthetic data.

The rest of the paper is organized as follows. Section 2 discusses related work on browser fingerprinting and provides some theoretical foundations on differential privacy. In Section 3, we explain our data collection, feature extraction, and define a non-private baseline. Section 4 and Section 5 evaluate the two privacy protection mechanisms, namely differentially private classifiers and differentially private data generation. Finally, in Section 6, we discuss the limitations of our approach, draw conclusions from our results, and provide an outlook to future work.

2 Background and Related Work

To differentiate our work from others, we provide an overview of related work concerning browser fingerprinting and classification. We highlight the restrictions of the different approaches but also point out similarities and useful aspects. In this regard, we also provide an introduction to differential privacy to make our own approach comprehensible.

2.1 Browser Fingerprinting and Classification

Early approaches on browser fingerprinting include *active* methods, which analyze installed fonts and plugins by actively injecting requests via JavaScript or similar. Eckersley [Eck10] was one of the first to identify such means. Succeeding Eckersley's work, others improved fingerprinting by using dedicated APIs like Canvas [MS12], WebGL [CLW17], or by utilizing font metrics [FE15] and JavaScript engines [MBYS11, MRH+13]. Summarizing the most commonly used features, Laperdrix et al. performed a large-scale analysis on browser fingerprinting [LRB16], and more recently, also conducted a survey regarding active approaches and online tools [LBBA20]. Further work focuses on the evolution of fingerprints over time [VLRR18].

Even though *active* fingerprinting is well-studied, it may be circumvented by disabling Flash, WebGL, and Canvas [AL17], or even imposes security risks, as JavaScript is prone to drive-by downloads and enables cross-site scripting [MRH+13]. Several browsers already implement techniques to block API-based methods [GLB18]. This process is hindered further with users' rising privacy-awareness and the introduction of encrypted traffic. New methods have to be employed, e. g., by *passively* monitoring traffic and extracting relevant information regarding characteristic patterns. Browser traffic is characterized as highly dynamic and unpredictable, as it is dependent on user input and the content of websites [BvC+17, BvC+20], and differs from other applications [vBC+20]. Zhioua [Zhi15] showcased the diverging resource scheduling of browsers. Consequently, a feature set capturing this behavior is reasonable.

An early approach to *passive* fingerprinting was proposed by Yen et al. [YHMR09]. Introducing flow-related features, such as the number of simultaneous flows and the time between consequent flows, the authors characterized the behavior of four different browsers on the basis of the Alexa Top 150 websites and achieved a precision and recall of at least 0.71. Only utilizing interval-based features related to the packet sizes, reasonable classification accuracies have been achieved for the Alexa Top 9 websites by Yu and Chan-Tin [YC14]. They compared different ML algorithms with C4.5 performing best. Samizade et al. [SSSG20] conducted similar studies regarding the feature space. In addition to the size-related features, they also included features concerned with the sheer number of packets. They trained a Neural Network (NN) and achieved high accuracies. Other algorithms, namely Support Vector Machine (SVM) and RF, performed significantly worse in comparison. Other works focused on combined browser detection with other detection problems. Mühlstein et al. [MZB+17] identified triples of OS, application, and browser. The authors attributed bursty behavior to browser traffic and, consequently, included features related to traffic peaks into their feature space. Using SVMs, they were able to achieve an accuracy over 0.96. Richardson and Garcia [RG20] focused on the detection of tuples composed of OS and browser. They, too, included burst-related features and the number of concurrent flows. They compared supervised and unsupervised approaches, with the k-nearest Neighbor (kNN) al-

gorithm and hierarchical ward clustering performing best, respectively. Husak et al. [HČJČ16] fingerprinted the unencrypted initial packets of the SSL/TLS handshake, a commonly used transport layer protocol for encryption. They reconstructed the HTTP User-Agent of clients from the generated fingerprints by matching the prints with a dictionary.

While the passive approaches avoid problems of active mechanisms, new issues emerge. With browser detection more information may be leaked, e. g., visited websites [YHMR09], which constitutes an invasion of privacy. So, further privacy mechanisms have to be employed to protect the data and the ML model, which is the focus of our work.

2.2 Differential Privacy

As already mentioned before, both privacy protections mechanisms in the browser fingerprinting use case build upon differential privacy. Therefore, we define differential privacy as formalized by Dwork [Dwo06, p. 9] – slightly adapted for (ε, δ)-differential privacy [DR14]:

Definition 1 ((ε, δ)-Differential Privacy)   A randomized function K gives (ε, δ)-differential privacy if for all data sets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(K)$,

$$Pr[K(D_1) \in S] \leq \exp(\varepsilon) \times Pr[K(D_2) \in S] + \delta \tag{1}$$

In this definition, ε is commonly referred to as the privacy budget and δ as the failure probability [JE19, DR14]. In general, smaller values of ε offer higher privacy guarantees but lower utility. Accordingly, larger values of ε reduce the privacy guarantees but increase utility. We investigate the impact of the privacy budget for browser fingerprinting in Section 4.

In this paper, we consider the application of differentially private mechanisms for the browser fingerprinting use case. Overall, we consider two research directions of differential privacy: privacy-preserving ML and privacy-preserving synthetic data generation.

In *privacy-preserving ML*, researchers design algorithms to release ML models without harming the privacy of individuals [VSBH13]. Chaudhuri et al. present privacy-preserving versions of SVMs and Logistic Regression [CMS11]. Vaidya et al. propose a private Naive Bayes classifier [VSBH13]. To guarantee differential privacy, the authors add Laplace noise to the classifier's parameters. Fletcher and Islam design a DPRF classifier using the Exponential mechanism [FI17]. That is, a scoring function assigns utility values to all possible outputs. Higher utility values are assigned to more appealing outputs [MT07]. Further, with higher utility, outputs are also returned with an exponentially increasing probability [MT07, FI17]. Later, Fletcher and Islam surveyed existing private DT classifiers [FI19]. The authors compare existing DT algorithms, e. g., in terms of accuracy.

For *privacy-preserving synthetic data generation*, researchers make use of Generative Adversarial Networks (GANs) [GPM+14]. As described by Goodfellow et al. [GPM+14], the generative and the discriminative model in the GAN framework have the following opposing functions. The generator tries to create synthetic data, which should reflect the real data distribution. Given such synthetic and real samples, the discriminator tries to identify them either as synthetic or as real. Overall, the goal of the framework is to create synthetic data that is indistinguishable from real data for the discriminator [GPM+14].

Several studies combine GANs with differential privacy. Xie et al. developed a differentially private GAN [XLW+18]. The authors ensure differential privacy with the addition of noise during training. Jordon et al. combine GANs with the Private Aggregation of Teacher Ensembles (PATE) framework to generate private synthetic data [JYv19]. Torkzadehmahani et al. propose a conditional GAN training framework, which is differentially private with the addition of Gaussian noise while optimizing the GAN's discriminator network [TKP19]. Their framework can create synthetic data but also the respective labels for supervised ML tasks. Rosenblatt et al. [RLP+20] build upon Conditional Tabular GAN (CTGAN) [XSCV19]. The authors apply the PATE framework to CTGAN to create PATE-CTGAN and add in another approach noise to the discriminator resulting in DP-CTGAN. Additionally, the authors propose the QUAIL method, which can enhance the quality of synthetic datasets for supervised ML tasks [RLP+20].

3 Dataset Description and Baseline Evaluation

In this section, we define the underlying methodology, utilized tools, and set a baseline model without privacy constraints for follow-up analyses. This includes the data collection, feature extraction from said data, and the training of an RF with optimized hyper–parameters.

3.1 Experimental Setup

We utilize a simple testbed, consisting of a Linux Desktop (Ubuntu 20.04), which automatically visits the Tranco Top 500 websites [LvTJ19] and records the resulting network traffic. The utilized browsers are Chrome (Version 89.0) and Firefox (Version 87.0). Both browsers are in their standard configuration, so no extra plugins or addons are installed. The ranking is iterated from top to bottom and is repeated five times for each browser. This results in 2500 runs each or 5000 runs in total, with runs being the webpage calls. After each run, the local cache is emptied, and unsuccessful runs are canceled after 60 seconds. The two browsers are measured in an alternating manner, e. g., if the first browser visits a website, the site is immediately afterwards visited by the second browser. This results in paired measurements, which should ensure that both measurements within one pair face similar network conditions and website content, although these influence factors could not be controlled in the measurement setup. The traffic collection stops two seconds after the *LoadEventEnd* is triggered, which is a property supported by most browsers and determines the time, when the static content is fully loaded. To ensure that there are no fragments of traffic prior to the webpage call, the head of the data is cut off according to the *NavigationStart* property, which determines the exact start of the call. Similar to the *LoadEventEnd*, this is also a property supported by most browsers. We only include traffic from the desktop to external destinations or vice-versa. So, no local traffic is included, which filters out, e. g., DNS requests. Lastly, we extract our features from this processed data.

3.2 Feature Extraction

The focus of our feature set is on the potential differences in scheduling behavior and traffic flow. The individual samples in the dataset are the webpage calls, which are described on different granularities, i. e., packet- and flow-based, briefly shown in Table 1.

Table 1: Coarse Overview of Extracted Features

	Feature	Description
Generic	Packet/Flow Count Protocol/Flag Count Protocol Entropy IP/Port Entropy	Number of packets/flows. Occurrences of transport protocols and TCP flags. Dispersion of utilized transport protocols. Dispersion of involved IPs/Ports.
Flow-based	Time between Flows Parallel Flows Flow Duration Flow Size	Time until the subsequent **non**-overlapping flow. Number of maximum coexisting other flows. Lifetime of flows. Size of flows.
Packet-based	Burst Count Burst Duration Inter-Arrival Time	Number of traffic peaks. Duration of traffic peaks. Time between packet-arrivals.

Generic features have the coarsest granularity, as they concern the whole run or are applicable on both packet- and flow-basis. They are easily extractable since they are merely simple counters. This feature group consists of the sheer number of packets and flows in a run, as well as the number of occurrences of transport protocols, i. e., TCP and UDP, which is applicable to either packets or flows. Additionally, we record the occurrences of various TCP flags, e. g., FIN or RST, since the browsers may choose to terminate their connections differently. The protocol entropy is a more compact metric to measure the transport layer protocol diversity. We also measure the IP entropy, as the diversity of addressed IPs may vary due to addressing of different, possibly browser-specific, Content Delivery Networks (CDNs). Last, the port entropy is of interest, since it captures the ratio of port 80 and 443, i. e., unsecured and secured transmission.

Flow-based features include more sophisticated features, which are no mere counters but contain statistics about the underlying distribution, i. e., mean, standard deviation, skewness, kurtosis, minimum, maximum, and median. The time between non-overlapping flows can be interpreted as the inter-arrival time (IAT) of flows and is motivated by the fact that one resource may be put on hold until another download is finished. Furthermore, since a diverging scheduling behavior is possibly more visible at the beginning, we calculate this feature not only over all flows but also for the first x flows, with $x \in \{5, 10, 20, 30, 40\}$. Moreover, we compute the maximum number of other coexisting flows for each flow, based on the fact that browsers might only establish a bounded number of connections. Since this mostly concerns connections to a specific destination, we calculate this both with and without respect to the destination. Even though the size of resources is highly dependent on the website, the browsers generally may have different priorities on which content is downloaded first. Therefore, we also collect statistics about the flow duration and size. Additionally, we calculate statistics about the size for the first x flows, since contents may be requested in varying order.

The most fine-granular features are *packet-based* features. We mainly include features capturing the burstiness of the connections, such as the burst count and the burst duration, done both on the basis of the sheer number of packets and their size. We compute this with and without respect to the corresponding flow since such a bursty behavior may only be visible when looking

(a) Ranking of Input Features (Feature Selection)   (b) Ranking of Features of Actual Output RF
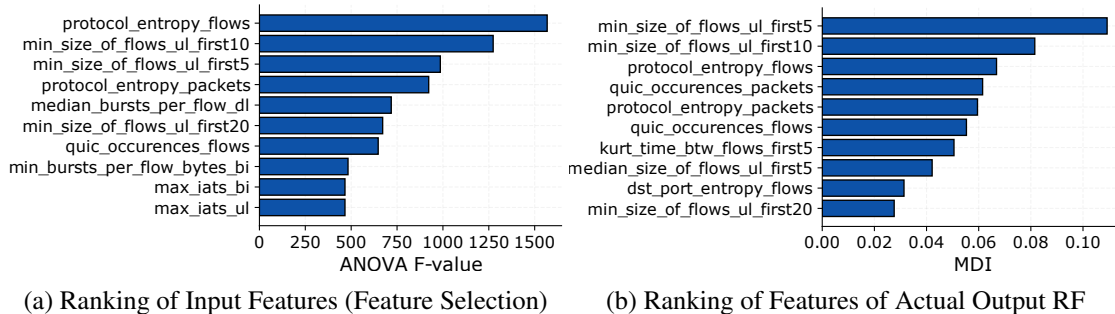
Figure 1: Feature Importances Determined by the Steps of the ML-Pipeline

at connections on their own. Lastly, the packet IAT can also be seen a measure of the burstiness when detecting highly-varying or skewed inter-arrival times.

Counting in all different variations of each feature, e. g., dividing by up- and downlink, isolating the first flows, or other mentioned constraints, this results in a total of 368 features, including all the different statistical measurements.

3.3 Browser Fingerprinting without Privacy Constraints

As a first scenario and to define an upper bound for the upcoming privacy analyses, we evaluate the potential classification accuracy with optimal parameters, based on a non-private RF classifier. RFs are often described as robust to overfitting and outliers [Bre01] and tend to perform well for classification tasks [WAF16]. We achieve the optimization via nested cross-validation (CV), which consists of a 10-fold outer and a 10-fold inner CV. The outer CV is useful to determine the robustness of the model by creating different folds for training and test set, while the inner CV is used for hyper–parameter optimization. We focus on the number of features, selected with the ANOVA F-value, and search the corresponding parameter space by doubling the number for every step, i. e., it is always a power of 2, except the last step, which considers all features.

Figure 1 illustrates the Top 10 features identified by the corresponding step in the ML-pipeline for one of the outer folds. Thus, Figure 1a depicts the features fed into the RF and their respective ANOVA F-values, while Figure 1b shows the Mean Decrease in Importance (MDI) of features in the output model, i. e., the trained RF. The protocol entropy of flows ranks high in both rankings. The same feature on packet-basis, as well as the occurrences of the Quick UDP Internet Connections (QUIC) protocol are also contained, with QUIC being a novel alternative to TCP and mainly supported by Chrome. The size of the first flows is also a feature which is reoccurring multiple times. Looking at the raw traces, Chrome establishes more connections initially but ultimately only uses one for the transmission. The Top 10 of the feature selection contain several burst-related features, including packet IATs, which confirms the observations of related work [RG20, MZB+17], whereas the Top 10 of the actual trained model only contain one feature regarding the flow IATs. Lastly, the ranking of the output model contains the destination port entropy, as Firefox tries to establish a secure connection first, i. e., over port 443 instead of port 80. Both of the shown rankings contain no mean values, but mainly minimum and maximum,
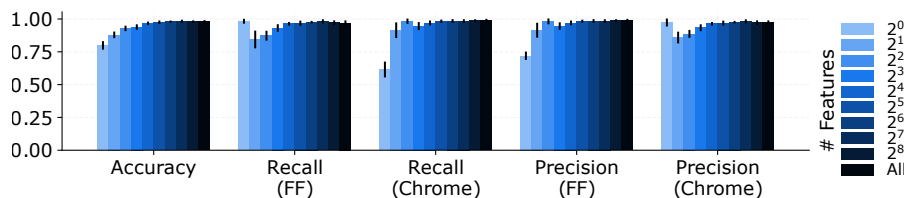
Figure 2: Results of the Hyper–parameter Optimization of One Outer Fold

attributable to both browsers loading the same content but possibly different scheduling. They identify a similar set for the Top 10, but the order differs, as the feature selection analyzes linear dependencies, while the RF itself also learns non-linear relationships.

Over the ten outer folds, we achieve a stable average accuracy of 0.982, with a standard deviation of 0.007. For the precision and recall for Chrome, we achieve values of 0.979 and 0.985, respectively. Analogous, for Firefox we achieve a precision of 0.985 and a recall of 0.978. For both browsers, these values are stable, with standard deviations not exceeding 0.014.

While the performance regarding accuracy, recall, and precision is stable, the optimal number of features varies from 64 to all features. Figure 2 illustrates the average performance of the ten inner folds of one outer fold, with error bars representing one standard deviation from the mean. The accuracy improves when selecting additional features. However, diminishing returns occur, and the difference between, e.g., 2^6 and 2^8 features is negligible. Surprisingly, the prediction works well for only one feature. Since a majority of the websites already support QUIC, this becomes a useful feature for distinguishing the two browsers by simply checking for the UDP protocol on the transport layer. This comes with low recall and precision for Chrome and Firefox, respectively, as websites not enabling QUIC are automatically classified as Firefox. This is balanced out when utilizing more features. So, utilizing extra features is desirable.

4 Browser Fingerprinting with Private Classifier

In this section, we employ a differentially private classifier to protect the resulting ML model against adversarial attacks. Firstly, we describe the rationale behind using a differentially private classifier for browser fingerprinting. Secondly, we present the results for this scenario to illustrate the trade-off between privacy and utility for the resulting ML model. Through the variation of the privacy budget ε, we can detect the best trade-off between privacy and utility.

4.1 Browser Fingerprinting with Differentially Private Classifier

In this scenario, the adversary only has access to the resulting ML model and not to the dataset. The access to the ML model is often categorized as black–box or white–box access [JE19]. In the black–box setting, adversaries can only query the model [JE19]. In the white–box model, they have access to the ML model's parameters and its structure [SSSS17]. Subsequently, the goal of the adversary is to infer something about the underlying data, e.g., with membership inference or attribute inference attacks [SSSS17, FLJ+14]. Within membership inference attacks, adversaries try to uncover if a certain data point has been part of the training dataset [SSSS17].
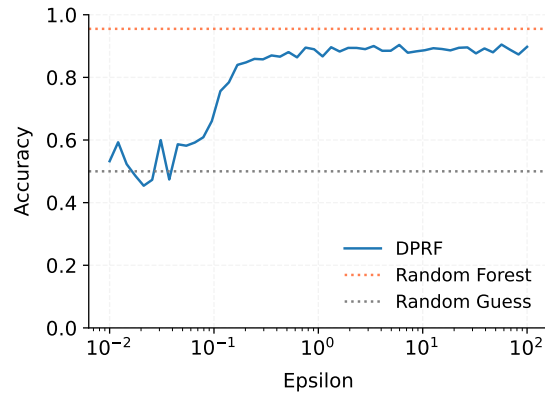
Figure 3: Privacy-Utility Trade-off for Differentially Private Random Decision Forest

In attribute inference attacks, adversaries want to infer sensitive attributes of a specific data point but have only access to the ML model and some additional non-sensitive information [JE19]. According to Shokri et al. [SSSS17], differentially private ML models reduce the probability to successfully perform membership inference attacks.

We choose the DPRF classifier by Fletcher and Islam for browser fingerprinting [FI17]. Our decision is based on a recent survey on DT classification, which shows that the DPRF reaches high accuracy values and can handle continuous attributes [FI19]. To compare the results of the DPRF classifier, we also train an unoptimized non-private RF classifier on the dataset, as we fix the number of features to 10 in the following analyses. By varying the privacy budget ε for the DPRF, we can assess the trade-off between privacy and utility and detect the best trade-off for our browser fingerprinting dataset.

4.2 Results for Differentially Private Random Forest

To protect the privacy of the data points in our ML model, we evaluate the applicability of a DPRF classifier for browser fingerprinting. With a privacy budget of $\varepsilon = 1$, the private RF classifier reaches an accuracy of 0.877 with 10 features. In comparison to the non-private classifier, accuracy drops by 8 percentage points from 0.955 to 0.877. The optimized non-private classifier performs even slightly better, reaching an accuracy of 0.982. Precision and recall are relatively stable across all browsers for non-private classifiers. With the private classifier, we observe noticeable differences between the two browsers. For Chrome, we reach a precision of 0.924 and a recall of 0.825. For Firefox, precision is 0.847, and recall is 0.930, i.e., we reach a higher precision for Chrome but a higher recall for Firefox.

Choosing a reasonable privacy budget is difficult as there is no concrete guideline; values in research often vary from 0.05 to 10 or even larger values [JE19]. As a result, there is a trade-off between the model's utility and privacy guarantees for the individual data points. Figure 3 visualizes the accuracy for different privacy budgets (0.01 to 100), which illustrates the trade-off between privacy and utility for our ML model. Additionally, Figure 3 also includes an upper bound with the non-private classifier (Accuracy = 0.955) and a lower bound with the random

guess (Accuracy = 0.5). While the model performs not much better than a random guess for $\varepsilon \leq 0.07$, accuracy values increase steadily for larger epsilons. At $\varepsilon \approx 0.11$, the accuracy is already about 0.75 and increases further up to a plateau at $\varepsilon \approx 0.51$ with an accuracy of 0.88. From there on, accuracy values fluctuate only slightly. The model reaches the maximum value at $\varepsilon \approx 56$ with an accuracy of 0.91, which is still slightly worse than the non-private classifier. Nevertheless, these results illustrate the potential of the differentially private ML model. We find the best trade-off for a relatively high privacy guarantee at $\varepsilon = 0.51$, where we can identify the browser with an accuracy of 0.88.

5 Browser Fingerprinting with Private Data

In this section, we generate differentially private synthetic datasets to protect directly the data and not only the resulting ML model. Firstly, we explain how we create the differentially private synthetic datasets. Secondly, we evaluate if those generated synthetic datasets might already be suitable for real-world applications.

5.1 Browser Fingerprinting with Differentially Private Synthetic Data

As adversaries have direct access to data in this scenario, it is insufficient to protect only the resulting ML model. To protect the raw dataset, we create a differentially private synthetic dataset for browser fingerprinting by applying PATE-CTGAN and QUAIL [RLP+20].

PATE-CTGAN Based on the PATE framework [PA17] and subsequent research on PATE-GAN [JYv19], Rosenblatt et al. proposed PATE-CTGAN [RLP+20]. As Jordon et al. have shown for PATE-GAN, differential privacy is achieved by applying a modified version of the PATE framework to the discriminator [JYv19]. In PATE-GAN, there are three components: a generator, teacher discriminators, and a student discriminator. As described by Jordon et al. [JYv19], PATE-GAN works as follows: At first, a given dataset is split into k disjoint subsets. Then, k classifiers (teachers) are trained to classify samples correctly, either as synthetic or real. Thereby, teachers receive real data only from their subset of the data but synthetic data from the generator. Subsequently, these teachers form an ensemble of ML models, which is used to train the student discriminator. That is, the student is trained on synthetic samples that have been labeled by the teachers. The authors ensure via normalization and initialization of the generator that those training samples also comprise realistic synthetic samples. The labeling process guarantees differential privacy as noise is added to the aggregation of the teachers' predictions. Based on the student's classification loss, the student model and the generator are updated via backpropagation. Overall, that means the teachers try to improve the classification of real and synthetic data, the generator tries to deceive the student discriminator, and the student discriminator tries to minimize classification loss on the dataset labeled by the teachers [JYv19]. Rosenblatt et al. modify the PATE-GAN approach slightly for PATE-CTGAN [RLP+20]. Instead of the traditional GAN framework, the authors use CTGAN in combination with the PATE framework. The main difference is that the authors use k conditional generators for each subset of the data, unlike using a single generator as in PATE-GAN [RLP+20].

Table 2: Results for Random Forest on 10 Synthetic Datasets

Case	Result	Train	Test	Accuracy	Precision		Recall	
					Chrome	Firefox	Chrome	Firefox
1	Min	Synth.	Synth.	0.922	0.932	0.914	0.906	0.937
1	Avg	Synth.	Synth.	0.951	0.955	0.948	0.945	0.957
1	Max	Synth.	Synth.	0.971	0.977	0.969	0.968	0.974
2	Min	Synth.	Real	0.551	0.746	0.529	0.153	0.802
2	Avg	Synth.	Real	0.733	0.839	0.708	0.566	0.900
2	Max	Synth.	Real	0.887	0.903	0.880	0.877	0.961

QUAIL Method   To create a differentially private synthetic dataset for supervised ML tasks, Rosenblatt et al. propose the QUAIL method [RLP+20]. The method's general idea is to make use of the composition theorem of differential privacy [DR14] with the combination of a differentially private classifier and a differentially private synthesizer [RLP+20]. On a high level, the synthesizer creates an unlabeled private synthetic dataset, which is then labeled by the ML model. To create the unlabeled dataset, the synthesizer is trained on the real dataset *without* the target class. Similarly, the classifier is trained on the real dataset *with* the target class to create a differentially private ML model. The resulting model then assigns each synthetic sample in the unlabeled dataset a label [RLP+20]. For the browser fingerprinting use case, we consider our dataset with 10 features. We choose PATE-CTGAN with $\varepsilon = 1$ and $\delta = 10^{-5}$ as synthesizer [RLP+20]. For the classifier, we choose a differentially private Logistic Regression (DPLR) with $\varepsilon = 1$ and $\delta = 0$ [CMS11]. Our implementation is based on the SmartNoise SDK [Ope21] and diffprivlib [HBML19]. Again, we consider two different cases for our evaluation. In the first case, we train and evaluate a RF classifier on private synthetic data. In the second case, a RF classifier is also trained on private synthetic data. However, the classifier is now evaluated with real data, which allows us to evaluate the utility of our synthetic datasets more realistically.

5.2   Results for Differentially Private Synthetic Data

To protect not only the resulting ML model but the data itself, we apply the QUAIL method with PATE-CTGAN and a DPLR to our dataset. As described before, we evaluate two different cases. In the first case, we apply 10-fold CV on the synthetic dataset. In the second case, we predict class labels for real data. That is, the classifier is trained on a synthetic dataset but evaluated on real data. Overall, we create 10 synthetic browser fingerprinting datasets for our evaluation. During data creation, we ensure that the dataset is relatively balanced, i. e., we select only datasets where the ratio of the frequency of the minority class and the majority class is greater or equal than 0.75. In this case, class imbalance might result from the differentially private classifier assigning the labels for the synthetic samples.

As the synthetic dataset already guarantees differential privacy, we use a non-private RF classifier to obtain the results. Table 2 shows the accuracy, precision, and recall for both cases. The results based on the synthetic data are generally higher if we evaluate the classifier on the synthetic data. We report the minimum, average, and maximum value across all 10 datasets.

On the synthetic data, we reach accuracy values ranging from 0.922 in the worst case to 0.971 in the best case. The values for precision and recall differ only slightly between Firefox and Chrome and are on the same level as the accuracy.

On the real data, we reach accuracy values from 0.551 to 0.887. We notice the same effect as with the DPRF classifier. There is a higher precision for Chrome but a lower recall. In the worst case, the recall is only 0.153 for Chrome, while the worst precision for Firefox is 0.529.

Despite the worst-case scenario, where the classifier is not much better than a random guess, there is potential for private synthetic data generation. The average accuracy of 0.733 with PATE-CTGAN and QUAIL is already relatively high compared to a random guess.

6 Conclusion

In this paper, we propose a passive browser fingerprinting method based on explainable network traffic features to differentiate between Mozilla Firefox and Google Chrome. We include a variety of features aiming to be comprehensible and incorporate novel advancements such as the UDP-based QUIC protocol in addition to the TCP-based HTTP. The robustness of these features has to be evaluated further because QUIC is already supported by some Firefox nightly builds, possibly making protocol-based features less viable. However, we also deem other features as important, so the classification is not solely dependent on the protocol-based features. We involve a wide range of websites based on the research-oriented Tranco ranking and measure both browsers under equal conditions.

To maximize the possibly achievable accuracy and define an upper bound for further analyses, we train an optimized Random Forest with hyper–parameter optimization, achieving an accuracy over 0.98. In addition to the optimized model, we also conduct various privacy evaluations, namely privacy-preserving Machine Learning and privacy-preserving synthetic data generation. With a differentially private Random Decision Forest [FI17], we achieve an accuracy of roughly 0.88, thus providing a reasonable trade-off between privacy and accuracy. The usage of differentially private synthetic data also shows promising results for real-world application. To create the synthetic datasets, we apply the QUAIL method by Rosenblatt et al. [RLP+20], where we combine PATE-CTGAN and a differentially private Logistic Regression. We reach an accuracy of 0.89 in the best-case scenario if we train on synthetic data and classify real-world data. While there are some open questions, e. g., which privacy budget should be used, differentially private synthetic data generation might be a promising approach for researchers to share sensitive data.

Similar to other fingerprinting research, there are some limitations. We consider perfect conditions for data collection, which may not be realistic. In the real world, some background noise resulting from other applications or multi-tab browsing might be present. Additionally, there are overlapping traffic streams, i. e., we would have to figure out the beginning and end of website visits. The data might also be expanded by adding more OSs, browsers or even different browser versions. We plan to tackle these challenges using the same methodology in future work.

Bibliography

[AL17]   N. M. Al-Fannah, W. Li. Not All Browsers Are Created Equal: Comparing Web Browser Fingerprintability. In *International Workshop on Security*. Pp. 105–120. Springer, 2017.

[Bre01]   L. Breiman. Random Forests. *Machine Learning* 45(1):5–32, 2001.

[BvC$^+$17]   R. Bortolameotti, T. van Ede, M. Caselli, M. H. Everts, P. Hartel, R. Hofstede, W. Jonker, A. Peter. Decanter: Detection of Anomalous Outbound HTTP Traffic by Passive Application Fingerprinting. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. Pp. 373–386. ACM, 2017.

[BvC$^+$20]   R. Bortolameotti, T. van Ede, A. Continella, T. Hupperich, M. H. Everts, R. Rafati, W. Jonker, P. Hartel, A. Peter. HeadPrint: Detecting Anomalous Communications Through Header-Based Application Fingerprinting. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. Pp. 1696–1705. 2020.

[CLW17]   Y. Cao, S. Li, E. Wijmans. (Cross-)Browser Fingerprinting via OS and Hardware Level Features. In *Network and Distributed System Security Symposium*. Internet Society, 2017.

[CMS11]   K. Chaudhuri, C. Monteleoni, A. D. Sarwate. Differentially Private Empirical Risk Minimization. *Journal of Machine Learning Research* 12:1069–1109, 2011.

[DM21]   K. Dietz, M. Mühlhauser. Private Browser Fingerprinting. 2021.
              https://github.com/wintermute-project/private-browserfingerprinting

[DR14]   C. Dwork, A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9(3-4):211–407, 2014.

[Dwo06]   C. Dwork. Differential Privacy. In *Automata, Languages and Programming*. Pp. 1–12. Springer Berlin Heidelberg, 2006.

[Eck10]   P. Eckersley. How Unique Is Your Web Browser? In *Privacy Enhancing Technologies*. Pp. 1–18. Springer, 2010.

[Eur21]   European Commission. Reform of EU Data Protection Rules. 2021.
              https://ec.europa.eu/info/law/law-topic/data-protection/reform_en

[FE15]   D. Fifield, S. Egelman. Fingerprinting Web Users Through Font Metrics. In *Financial Cryptography and Data Security*. LNCS, pp. 107–124. Springer, 2015.

[FI17]   S. Fletcher, M. Z. Islam. Differentially Private Random Decision Forests Using Smooth Sensitivity. *Expert Systems with Applications* 78:16–31, 2017.

[FI19]   S. Fletcher, M. Z. Islam. Decision Tree Classification with Differential Privacy: A Survey. *ACM Computing Surveys* 52(4), 2019.

[FLJ$^+$14] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, T. Ristenpart. Privacy in Pharma-cogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *23rd USENIX Security Symposium*. Pp. 17–32. San Diego, CA, 2014.

[GLB18] A. Gómez-Boix, P. Laperdrix, B. Baudry. Hiding in the Crowd: An Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In *Proceedings of the 2018 World Wide Web Conference*. Pp. 309–318. ACM Press, 2018.

[GPM$^+$14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*. Volume 27. Curran Associates, Inc., 2014.

[HBML19] N. Holohan, S. Braghin, P. Mac Aonghusa, K. Levacher. Diffprivlib: The IBM Differential Privacy Library. *arXiv:1907.02444 [cs]*, 2019.

[HČJČ16] M. Husák, M. Čermák, T. Jirsík, P. Čeleda. HTTPS Traffic Analysis and Client Identification Using Passive SSL/TLS Fingerprinting. *EURASIP Journal on Information Security* 2016(1):1–14, 2016.

[JE19] B. Jayaraman, D. Evans. Evaluating Differentially Private Machine Learning in Practice. In *28th USENIX Security Symposium*. Pp. 1895–1912. 2019.

[JYv19] J. Jordon, J. Yoon, M. van der Schaar. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *International Conference on Learning Representations*. 2019.

[LBBA20] P. Laperdrix, N. Bielova, B. Baudry, G. Avoine. Browser Fingerprinting: A Survey. *ACM Transactions on the Web* 14(2), 2020.

[LRB16] P. Laperdrix, W. Rudametkin, B. Baudry. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. In *IEEE Symposium on Security and Privacy*. Pp. 878–894. 2016.

[LvTJ19] V. Le Pochat, T. van Goethem, S. Tajalizadehkhoob, W. Joosen. TRANCO: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Network and Distributed Systems Security (NDSS) Symposium 2019*. 2019.

[MBYS11] K. Mowery, D. Bogenreif, S. Yilek, H. Shacham. Fingerprinting Information in JavaScript Implementations. In *Web 2.0 Security and Privacy*. P. 11. 2011.

[MRH$^+$13] M. Mulazzani, P. Reschl, M. Huber, M. Leithner, S. Schrittwieser, E. Weippl. Fast and Reliable Browser Identification with JavaScript Engine Fingerprinting. In *Web 2.0 Security and Privacy*. P. 10. 2013.

[MS12] K. Mowery, H. Shacham. Pixel Perfect: Fingerprinting Canvas in HTML5. In *Web 2.0 Security and Privacy*. P. 12. 2012.

[MT07]   F. McSherry, K. Talwar. Mechanism Design via Differential Privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. Pp. 94–103. 2007.

[MZB+17] J. Muehlstein, Y. Zion, M. Bahumi, I. Kirshenboim, R. Dubin, A. Dvir, O. Pele. Analyzing HTTPS Encrypted Traffic to Identify User's Operating System, Browser and Application. In *14th Annual Consumer Communications & Networking Conference*. Pp. 1–6. IEEE, 2017.

[Ope21]   Open Differential Privacy. SmartNoise System: Tools for Differential Privacy. 2021. https://github.com/opendp/smartnoise-sdk

[PA17]   N. Papernot, M. Abadi. Semi-Supervised Knowledge Transfer for Deep Learning from Private Training Data. In *International Conference on Learning Representations*. P. 16. 2017.

[RG20]   O. Richardson, J. Garcia. A Novel Flow-Level Session Descriptor With Application to OS and Browser Identification. In *Network Operations and Management Symposium*. Pp. 1–9. IEEE, 2020.

[RLP+20] L. Rosenblatt, X. Liu, S. Pouyanfar, E. de Leon, A. Desai, J. Allen. Differentially Private Synthetic Data: Applied Evaluations and Enhancements. *arXiv:2011.05537 [cs]*, 2020.

[SSSG20] S. Samizade, C. Shen, C. Si, X. Guan. Passive Browser Identification with Multi-Scale Convolutional Neural Networks. *Neurocomputing* 378:238–247, 2020.

[SSSS17] R. Shokri, M. Stronati, C. Song, V. Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (SP)*. Pp. 3–18. 2017.

[TKP19]   R. Torkzadehmahani, P. Kairouz, B. Paten. DP-CGAN: Differentially Private Synthetic Data and Label Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2019.

[vBC+20] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. R. Choffnes, M. van Steen, A. Peter. FlowPrint: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic. In *27th Annual Network and Distributed System Security Symposium*. The Internet Society, 2020.

[VLRR18] A. Vastel, P. Laperdrix, W. Rudametkin, R. Rouvoy. FP-STALKER: Tracking Browser Fingerprint Evolutions. In *IEEE Symposium on Security and Privacy*. Pp. 728–741. 2018.

[VSBH13] J. Vaidya, B. Shafiq, A. Basu, Y. Hong. Differentially Private Naive Bayes Classification. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. Volume 1, pp. 571–576. 2013.

[WAF16]   M. Wainberg, B. Alipanahi, B. J. Frey. Are Random Forests Truly the Best Classi-
            fiers? *Journal of Machine Learning Research* 17(1):3837–3841, 2016.

[XLW⁺18]   L. Xie, K. Lin, S. Wang, F. Wang, J. Zhou. Differentially Private Generative Adver-
            sarial Network. *arXiv:1802.06739 [cs, stat]*, 2018.

[XSCV19]   L. Xu, M. Skoularidou, A. Cuesta-Infante, K. Veeramachaneni. Modeling Tabular
            Data Using Conditional GAN. In *Advances in Neural Information Processing Sys-
            tems*. Volume 32. Curran Associates, Inc., 2019.

[YC14]   J. Yu, E. Chan-Tin. Identifying Webbrowsers in Encrypted Communications. In *Pro-
            ceedings of the 13th Workshop on Privacy in the Electronic Society*. Pp. 135–138.
            ACM, 2014.

[YHMR09]   T.-F. Yen, X. Huang, F. Monrose, M. K. Reiter. Browser Fingerprinting from Coarse
            Traffic Summaries: Techniques and Implications. In *6th International Conference
            on Detection of Intrusions and Malware, and Vulnerability Assessment*. LNCS 5587,
            pp. 157–175. Springer, 2009.

[Zhi15]   S. Zhioua. The Web Browser Factor in Traffic Analysis Attacks. *Security and Com-
            munication Networks* 8(18):4227–4241, 2015.