Tracked Without a Trace: Linking Sessions of Users by Unsupervised Learning of Patterns in Their DNS Traffic

Matthias Kirchler
Humboldt Universität Berlin
matt.kirchler@gmail.com

Dominik Herrmann
Fakultät III, Universität Siegen
dh@exomail.to

Jens Lindemann
Universität Hamburg
lindemann@informatik.uni-hamburg.de

Marius Kloft
Humboldt Universität Berlin
kloft@hu-berlin.de

ABSTRACT

Behavior-based tracking is an unobtrusive technique that allows observers to monitor user activities on the Internet over long periods of time – in spite of changing IP addresses. Previous work has employed supervised classifiers in order to link the sessions of individual users. However, classifiers need labeled training sessions, which are difficult to obtain for observers. In this paper we show how this limitation can be overcome with an *unsupervised* learning technique. We present a modified k-means algorithm and evaluate it on a realistic dataset that contains the Domain Name System (DNS) queries of 3,862 users. For this purpose, we simulate an observer that tries to track all users, and an Internet Service Provider that assigns a different IP address to every user on every day. The highest tracking accuracy is achieved within the subgroup of highly active users. Almost all sessions of 73 % of the users in this subgroup can be linked over a period of 56 days. 19 % of the highly active users can be traced completely, i. e., all their sessions are assigned to a single cluster. This fraction increases to 40 % for shorter periods of seven days. As service providers may engage in behavior-based tracking to complement their existing profiling efforts, it constitutes a severe privacy threat for users of online services. Users can defend against behavior-based tracking by changing their IP address frequently, but this is cumbersome at the moment.

Keywords

attack; privacy; behavioral fingerprints; k-means; clustering

1. INTRODUCTION

Many citizens are losing track of their digital footprint as service providers frequently ask for personal data. More worrying, however, are efforts to monitor users behind their back without their consent. A well-known example is the practice of online behavioral advertising, which allows businesses to target their ads to individual users [12]. To infer user interests and demographics, ad networks keep track of the websites a user visits by setting cookies and "supercookies" [3] or by extracting unique device fingerprints [44].

In this paper we study a more advanced tracking technique, **behavior-based tracking**, which does not rely on cookies or client-side scripting. Behavior-based tracking exploits the fact that the online behavior of users exhibits characteristic and recurrent patterns (cf. studies on revisitation behavior [45, 53] and uniqueness of browsing behavior [46, 47]), either resulting from habits and hobbies or due to automated processes, e. g., browsers that retrieve recently opened websites upon launch, or RSS readers that contact the same set of blogs every day. As users have no means to detect behavior-based tracking on servers, it constitutes a considerable threat to privacy.

Previous work [18] considered behavior-based tracking in the context of *recursive name servers* (resolvers) in the Domain Name System (DNS): Resolvers can search for characteristic patterns in the stream of queries in order to reidentify individual users in spite of changing IP addresses. This threat is especially worrying for users of third-party resolvers, whose popularity has increased significantly in the last few years. Large operators like Google and OpenDNS receive hundreds of billions of DNS queries per day [7, 48]. Measurements of the Regional Internet Registry administering IP addresses for the Asia Pacific (APNIC) indicate that Google's 8.8.8.8 resolvers are currently (July 2016) serving more than 12 % of all DNS queries worldwide [2]. Thus, they have become an attractive vantage point for surveillance.

Behavior-based tracking does not only affect users of public DNS services. It could also be adopted by *high-reach trackers*, such as those of Google and Facebook [58], to resurrect deleted third-party cookies. In principle, the technique can be employed by *any passive observer* on a network segment between users and their DNS resolvers.

Furthermore, the small size of DNS queries (0.05 % to 0.25 % of total network traffic [14, 49]) allows them to be stored for later analysis. Thus, the technique may become a useful tool for *attribution of malicious activities* during forensic investigations [19]. Investigators could use behavior-based tracking to trace the behavior of adversaries that use multiple IP addresses over time to cover their tracks. Behavior-based tracking may also be employed by *intelligence agencies* such as NSA and GCHQ, which have already

AISec'16 October 28, 2016, Vienna, Austria

ACM ISBN 978-1-4503-4573-6/16/10.

DOI: http://dx.doi.org/10.1145/2996758.2996770

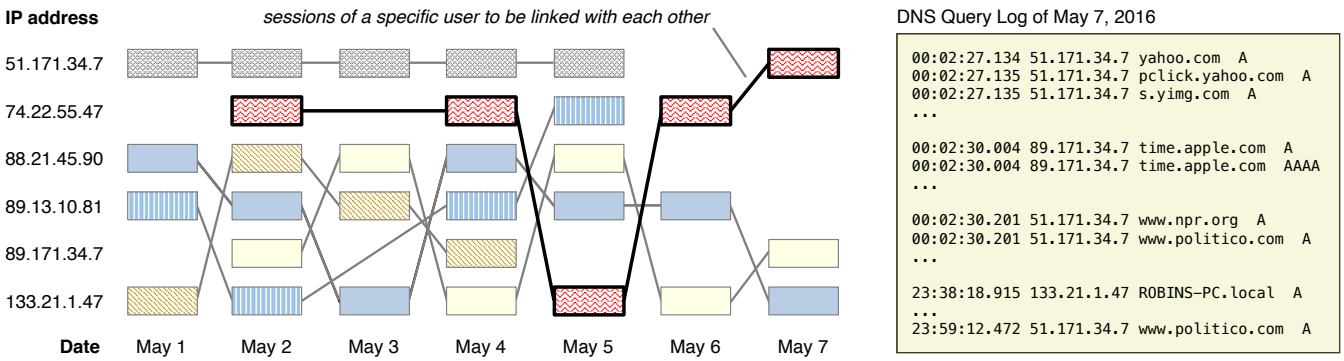


Figure 1: Scenario and objective of behavior-based tracking

leveraged tracking cookies and DNS traffic for surveillance purposes [11, 16].

Most of existing work on user re-identification employs *supervised* learning techniques, which require many labeled sessions of each user to achieve high accuracy: Yang [57] uses 300 sessions per user, Kumpošt and Matyáš [37] the traffic of a whole month. However, it is unrealistic to assume that this amount of labeled data is available in a user tracking scenario, because many Internet Service Providers (ISPs) assign short-lived IP addresses (cf. Sect. 3). On the other hand, limiting the assumed adversary to re-identifying users solely from one day to the next in a stateless fashion (cf. [18]) constrains the adversary artificially.

One obvious way to improve on existing work consists in applying more sophisticated supervised learning techniques, e. g., support vector machines. We approach the problem from a different angle in this paper and apply an *unsupervised* learning technique, which does not depend on labeled training data. We have evaluated our technique, which is based on the k-means algorithm [39], on a realistic dataset that contains the DNS queries of 3,862 users. According to our results an observer can link almost all sessions of 73 % of the more active users in our dataset over a period of 56 days. 52 % of the more active users are completely traceable over periods of seven days. Our technique can be used on its own in order to track users or it can be used as a labeling machine for a subsequently executed supervised scheme. We present two contributions in this paper:

1. To the best of our knowledge, we are the first to apply unsupervised machine learning techniques to the problem of tracking users in order to link multiple sessions at once. In contrast to previous work, our assumed observer does not need (unrealistically) large amounts of labeled training instances.

2. We present evaluation procedures and metrics that are tailored to the user tracking scenario and generate meaningful results for both observers and users.

This paper is structured as follows. We review privacy issues in the DNS in Sect. 2. After that we describe the problem setting in Sect. 3 and our unsupervised tracking technique in Sect. 4. Evaluation results are provided in Sect. 5, before we discuss limitations and future work in Sect. 6. Finally, we review related work in Sect. 7 before we conclude in Sect. 8.

2. PRIVACY ISSUES IN DNS

In the following, we briefly review the architecture of the DNS and the resulting privacy implications. The DNS (originally specified in RFCs 1034 and 1035 [42, 43]) is the most important name resolution service on the Internet. It provides access to a distributed database that is organized as a tree of domains. DNS is primarily used to look up the IP address for a given domain name, i. e., almost all communication activities start with one or multiple DNS queries.

Clients offload name resolution to dedicated servers, so-called *resolvers*. Resolvers forward queries to *authoritative name servers*, which store the actual mapping information. The vast majority of DNS queries is transmitted in the clear via UDP. All queries issued by the same user can be linked as long as the source IP address of the user stays the same and is not shared among multiple users.

The privacy implications of this design have been recognized only recently. As stated in RFC 7626 [6] it is straightforward for DNS resolvers (and any other observers on the path between client and resolver) to monitor the browsing behavior of their users. Furthermore, DNS queries issued by automated processes (e. g., mail clients, software update daemons, cloud-based services) may disclose information about installed applications, operating system, and other network services that are consumed by a client. Finally, some clients are configured incorrectly and leak internal hostnames to the resolver, either their own or hostnames of other devices in their local area network.

3. PROBLEM SETTING & ASSUMPTIONS

The aim of behavior-based tracking is to link activities of users in spite of changing IP addresses (cf. Fig. 1). We assume a passive adversary (the *observer*) that is able to obtain the DNS queries of a group of users, either by operating a DNS resolver and recording all queries or by monitoring the communication lines between clients and resolver.

Many ISPs force the broadband routers of their customers to reconnect periodically, assigning a different dynamic IP address each time. Empirical studies of large user groups indicate that IP addresses are often used for a period of 24 hours [40, 56]. Therefore, we will present and evaluate our behavior-based tracking scheme in a setting where all sessions have a fixed duration of one day. Further, we assume that every user contributes at most one session per day, that multiple users do not share a single IP address (i. e., all traffic that originates from the same source IP address during

Table 1: Structure of the data for behavior-based tracking

Session	**Day**	**User**	**Data Matrix X**
Session 1	1	1	$(0, 2, 0, 1, 0, 0, \ldots, 0, 4)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Session i	$t_i \in \{1, \ldots, D\}$	$u_i \in \{1, \ldots, k\}$	$x_i \in \mathbb{R}^d$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Session n	t_n	u_n	x_n
$n = \#\{\text{sessions}\}$	$D = \#\{\text{days}\}$	$k = \#\{\text{users}\}$	$d = \#\{\text{domains}\}$

a day belongs to the same user), and that traffic from different source IP addresses belongs to different users. These assumptions allow us to formalize the tracking problem. We defer the discussion of the ensuing limitations to Sect. 6.

An observer operates as follows. For a given time period of D consecutive days, the **Period under Consideration (PuC)**, the observer tries to link the sessions of each user by collecting them within a dedicated cluster. The ideal result would consist of as many clusters as users, each one only including all sessions of a single user. While many users will have exactly D sessions within a PuC, there may be less active users with fewer than D sessions.

Formalization.

A PuC contains n sessions (t, u, x) from an unknown number of $k \leq n$ users (cf. Table 1). The sessions are stored in a data matrix X with dimensions $n \times d$, whose rows are the vectors $x_1, \ldots, x_n$ with $n = \sum_{t=1}^{D} \#\{\text{active users on day } t\}$, and $d = \#\{\text{domains}\}$. Thus, $X_{i,j}$ is the number of times that domain j was queried in session i. Each user may be responsible for 1 to D sessions. The aim is to find a clusterer $f : \{\text{sessions}\} \subset \mathbb{R}^d \to \{\text{clusters}\} \hat{=} \{1, \ldots, k\}$ that correctly maps the single sessions into homogeneous clusters each corresponding uniquely to a user (we use $\mathbb{R}$ instead of $\mathbb{N}_0$ for reasons explained in Sect. 4). The cluster of a user u who was active on all days should contain all sessions $(1, u, *), \ldots, (D, u, *)$, but no sessions of other users.

4. TRACKING TECHNIQUE

We perform behavior-based tracking with a slightly modified k-means algorithm [39], a well established unsupervised learning technique for points in the Euclidean space $\mathbb{R}^d$. We use k-means for clustering, because of its efficiency and because we want to compare our unsupervised approach with the supervised approach of [18], which is based on the conceptually related 1NN classifier [41]. We explain our changes to the classic k-means algorithm in Sect. 4.1 and describe the implementations which we have evaluated in Sect. 4.2.

4.1 Modified k-means Algorithm

The classic k-means method consists of three steps: *initialization*, *assignment*, and *update*. For sake of clarity, we will start out with a description of the overall procedure and defer the details of initialization to Sect. 4.1.2.

4.1.1 Clustering Procedure

The initialization step sets up k cluster centroids $c_1, \ldots, c_k$ as an initial guess. Assignment and update steps are executed in a loop until the resulting clustering does not change any more, i. e., until it has converged. The *assignment step* runs through all sessions $x_1, \ldots, x_n$ and assigns them to the cluster centroid that is nearest to them with respect to a distance metric: $c(x_i) = \text{argmin}_{c_1,\ldots,c_k}\{\text{dist}(x_i, c_j)\}$. During the *update step*, every centroid c_j is updated to $c_{j;\text{new}}$ by computing the mean of all n_j sessions that are currently assigned to it:

$$c_{j;\text{new}} = \frac{1}{n_j} \sum_{i:\ c(x_i)=c_j} x_i$$

Our first modification of k-means is to replace the commonly used Euclidean distance metric with the **cosine distance** that has also been applied in previous work [18, 37]:

$$\hat{c}(x_i) = \text{argmin}_{c_1,\ldots,c_k} \left\{ \text{dist}_{\cos}(x_i, c_j) = 1 - \frac{\langle x_i, c_j \rangle}{||x_i||\ ||c_j||} \right\},$$

where $\langle \cdot, \cdot \rangle$ and $|| \cdot ||$ denote scalar product and norm in $\mathbb{R}^d$, respectively. Cosine distance is closely related to *cosine similarity*, $\text{sim}_{\cos}(x, y) = \frac{\langle x,y \rangle}{||x||\ ||y||} = 1 - \text{dist}_{\cos}(x, y)$, which equates to the cosine of the angle of two vectors, i. e., their magnitude is neglected [4]. Note that cosine distance is not a *metric* (e. g., a distance of 0 between two points does not necessarily imply equality), i. e., it is not guaranteed that k-means will converge. This *spherical k-means* algorithm [10] is known to work well with high-dimensional and sparse data.

The second modification is the introduction of a **restriction** on the maximum number of sessions assigned to a cluster. Given n sessions from D days, maximum cluster size can be restricted to D *(soft restriction)*, as we assume that a user contributes at most one session per day (cf. Sect. 3). The *strict restriction* additionally ensures that each cluster holds at most one session from each day in the PuC. The restriction is enforced after the assignment step by inserting a *reassignment step*, in which we iterate over the clusters that violate it (cf. Alg. 1). The most remote sessions in a cluster are reassigned to the nearest possible other clusters (ensuring that they do not introduce new violations there) until all violations have been resolved. Note that the *strict restriction* includes the *soft restriction*. However, we found that in some situations the *soft restriction* performs better.

4.1.2 Initialization Strategies

After having described the overall procedure, we will now explain how cluster centroids are initialized. We consider two initialization strategies that correspond to the datasets we consider during the evaluation, the *clean dataset*, in which every user is active on every day, and the *realistic dataset*, in which users may be active only on some days (cf. Sect. 5.2 for details about the difference).

Algorithm 1 k-means clustering with strict restriction

```
1: procedure KMC-STRICT(X, index, iter)        ▷ X: matrix with sessions x_1, ..., x_n; index: index assigning 1, ..., n to day_1, ..., day_D; iter: max. no. of iterations
2:     initialize cluster centroids c_1, ..., c_k
3:     l = 0
4:     while not converged and l < iter do
5:         Assign: assign all sessions x_i in X to the nearest centroid
6:         Reassign: move sessions from overfull clusters (> 1 sessions per day) to closest non-full clusters
7:         Update: recompute cluster centroids as mean of assigned sessions
8:         l = l + 1
9:     end while
10: end procedure
```

As each cluster is meant to only contain sessions of a single user and users only change their IP address once a day, it makes sense to initialize the centroids with a single session of every user. For the **clean dataset**, the initializing of the centroids is straightforward: On every day there is exactly one session for every user $u \in \{1, \ldots, k\}$. Thus, we initialize the clusters with the sessions from the last day: $c_1, \ldots, c_k = x_{n-k+1}, \ldots, x_n$ (in principle, any day can be used).

However, this approach cannot be applied to the **realistic dataset** because the total number of active users in the PuC is unknown to the observer. Therefore, we have to estimate the number of clusters k. The estimate $\overline{k}$ is determined by finding the maximum number of sessions per day within the PuC, $k_{\max}$, and reserving space for additional users by computing $\overline{k} = (1+p)k_{\max}$. The observer has to choose an appropriate value for p, for instance, by consulting historic data. In a pretest we experimented with different values $p \in \{0, 0.05, 0.1, 0.25, 0.5, 1\}$ and found that p does not have a significant influence on the resulting accuracy: The ARI scores (defined later, cf. Sect. 5.2) varied by only 0.03 points. According to the results, larger values for p are better suited for longer PuCs. We set $p = 0.1$ in our experiments.

Based on this estimate, we initialize a subset C of the centroids by choosing all $k_{\max}$ sessions from the most active day $t = t_{\max}$. Then we determine the missing $\overline{k} - k_{\max}$ centroids by selecting those sessions from the remaining period that are farthest away (in terms of cosine distance) from the centroids in C as these sessions likely belong to other users.

4.2 Implementation

In order to explore the effectiveness of our clustering technique and to compare it to previous work, we have implemented and evaluated three clustering algorithms.

First, we have implemented our **modified k-means clustering technique (KMC)**, which operates on PuCs spanning multiple days. If the scenario in Fig. 1 corresponded to a PuC ($D = 7$), KMC would predict clusters that hold all sessions occurring from May 1 to May 7.

Second, we consider a technique based on the 1-nearest-neighbor (1NN) classifier that was proposed in [18], which is, to the best of our knowledge, the state of the art in behavior-based tracking using query behavior. In contrast to our approach, this technique is stateless and limited to linking sessions on two (typically adjacent) days. Therefore, an observer that uses the 1NN classifier is bound to lose track of users that are inactive on a given day (e. g., the two users who are inactive on May 3 in Fig. 1). In the example in Fig. 1, an observer might train the classifier with the sessions from May 6 (using three arbitrarily generated class labels) and use it to predict which of the sessions on May 7 belongs to which of the class labels generated on May 6.

In order to obtain comparable results, we extend the classification approach from [18] by implementing a **1-nearest-neighbor clustering technique (1NNC)** that chains together day-to-day predictions generated by a 1NN classifier. To this end, 1NN-like classification is performed repeatedly for all pairs of adjacent days within a PuC. The procedure starts on the last day $t = D$ of the PuC and iterates backwards day by day. The clusters are initialized with the sessions of the last day. In each iteration, 1NNC loops over the sessions of day t and adds the nearest session from day $t-1$ to the respective clusters. If there are fewer sessions on day $t-1$ than on day t, the remaining clusters will be closed on day t; if there are more sessions on day $t-1$ than on day t, new clusters will be initialized with the extraneous sessions. Thus, the resulting number of clusters depends on the data and does not have to be specified in advance.

Although this basic strategy subjects 1NNC to the same limitations as the approach in [18], 1NNC turns out to be quite effective in some of our experiments. Obviously, much more complex strategies for 1NNC are conceivable. For instance, one could implement a stateful variant that keeps clusters open when no additional sessions are added to them on a given day due to intermittent inactivity of a user.

Third, we report results for a **random clustering technique (RNDC)** that assigns every session to a randomly selected cluster to serve as a primitive baseline. It is initialized with $\overline{k} = 1.1k_{\max}$ clusters (like the KMC techniques) and we apply the *strict restriction* to the clustering.

5. EVALUATION

We have evaluated our tracking technique with a realistic dataset. We describe our data (Sect. 5.1) and methodology (Sect. 5.2) before we report results in Sect. 5.3.

5.1 Dataset

We obtained the dataset used in [18] in order to evaluate the three clustering techniques. This dataset is suitable for this purpose, because it complies fairly well with the assumptions described in Sect. 3. The dataset was collected at University of Regensburg (Germany) by instructing the university's DNS resolvers to log all DNS queries that originated from the student housing network (LAN only, no queries from the WiFi network). Thus, the dataset captures a large fraction of the surfing behavior of a relatively homogenous group, i. e., students who access the Internet for studying and leisure from their room. Note that we are not guaranteed to have the whole traffic of all users, because the dataset does not contain the DNS queries that are issued while accessing the network from other parts of the campus.

Every student signs up for one unique and static IP address that allows to connect a single device in their room to the broadband network. Further, students are prohibited from sharing their address with others, i. e., by setting up network address translating routers or WiFi access points. As a result, the dataset contains the ground truth for the mapping between users and queries, which allows us to evaluate the effectiveness of our tracking technique.

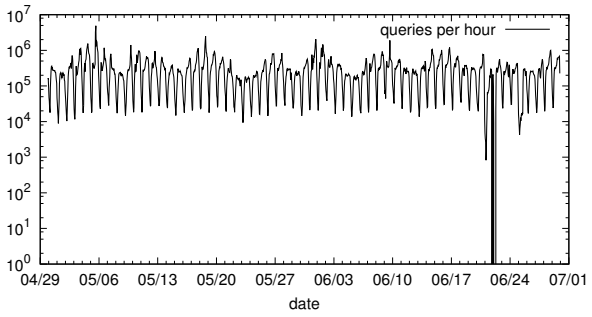


Figure 2: Time series of hourly query rate

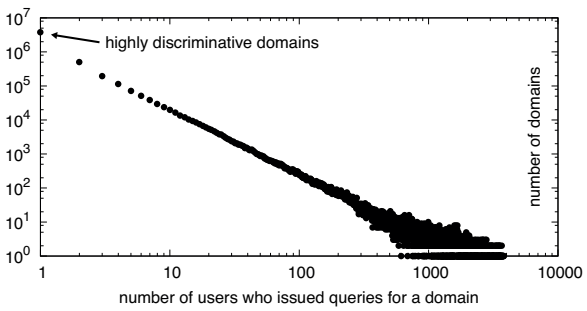


Figure 3: Discriminativeness of domains (original dataset)

In total, 431,210,371 queries for 5,010,507 distinct domains issued by 3,862 different users have been recorded between April 30, 2010 (0:00:00 am local time) and June 29, 2010 (11:59:59 pm local time). Each record consists of query time, domain, query type, and the source IP address, which has been replaced by a constant pseudonym by the computing center of the university in order to protect the identity of the students.

5.1.1 Preprocessing

We use the raw data from the DNS query log and transform it into a data matrix as described in Sect. 3. In order to obtain meaningful results, we perform a very conservative form of **feature selection** during this process. We delete all domains that have been queried in fewer than six sessions. This feature selection technique reduces the size of the data matrix considerably (but it does not have a significant impact on the results reported in this paper).

After feature selection we apply a commonly used **sublinear transformation** on the data matrix: To every cell in the data matrix we apply $f(x) = \log(1 + x)$, which limits the effect of extremely large outliers [55]. In pretests we also explored other directions, most notably different forms of *tf-idf transformations* from the text mining domain, where the term frequency is multiplied with the inverse document frequency [41]. However, these techniques did not prove effective. Future work may look into adding additional features such as the query type or leveraging temporal features like the "time of day". Another conceivable extension is to take the order of queries into account by creating n-grams [5].

5.1.2 Dataset Characteristics

Figure 2 shows a time series plot of the hourly query rate (before feature selection). The aggregated query behavior follows a diurnal pattern. As most users are students, it is not surprising that traffic peaks in the evening hours, mostly on Mondays, Tuesdays, and Wednesdays, while the weekends are less busy. Furthermore, the logging infrastructure seems to have malfunctioned between June 20 and June 22, where no or abnormally few queries were recorded.

Table 2 presents essential characteristics of the dataset before and after feature selection. While the number of distinct domains decreases considerably by 89.3 % to 534,269, the number of users remains unchanged at 3,862 and the number of queries decreases only marginally by 3.00 % to 418,288,004. The users vary considerably in terms of the extent of their online activity. The numbers of queries per user and distinct domains per user span several orders of magnitude, ranging from 1 to more than 28 million and from 1 to almost 100,000, respectively. For sessions with an assumed fixed duration of 24 hours, the median number of queries is 1,352 (IQR: 2,326) and the median number of distinct hostnames per session is 357 (IQR: 444). On the one hand, the user-domain data matrix is very sparse, because the query behavior is governed by a power law, i. e., there are few extremely popular domains followed by a long tail of domains that are queried very infrequently [26]. On the other hand, most users are very active: 75 % of the considered users are online on more than half of the days. On 75 % of the days, more than 2,053 users are active. All in all, our dataset exhibits characteristics that are in line with the results of Schomp et al. who provide a more extensive analysis of DNS client behavior [52].

5.1.3 Discriminativeness of Domains

Before applying learning techniques, we analyze the discriminative potential of domains [21]. To this end, we consider the relationship between domains and users. As shown in Fig. 3, the original dataset contains > 3.7 million domains that are only queried by a single user. Most of these domains are not useful to re-identify a particular user, because they are queried only once. However, some of them do survive feature selection (cf. row "Users per domain" in Table 2).

Further analysis revealed 273 domains (queried by 103 users) that are particularly discriminative, because in addition to being accessed by a single user only, they also *occurred in every session of this user*. In the following we present an instructive selection of such **highly discriminative domains (HDDs)**. The distinctiveness of some domains is due to unique identifiers related to hardware (MAC addresses, as in 00-02-**-**-**-**.uni-regensburg.de) and software (e. g., 6af9cf181c5****.users.storage.live.com) or generated by the user (SRV queries to a user's own chat server at _xmpp-client._tcp.****.homeip.net). Other cases, for instance SCH******-PC.uni-regensburg.de, reveal the hostname of a user's machine, which happens to be unique in our database. This behavior is not surprising, because many common operating systems issue queries for their own hostname and for hostnames found in their neighborhood. Furthermore, queries for some domains are the result of peculiar configuration choices (ntp2.ptb.de for clock synchronization) or forgotten settings (proxy.ucd.ie, the internal HTTP proxy of another university). Finally, we observed many domains for ordinary websites and weblogs, which might be periodically refreshed by RSS newsreaders or browsers in the background. Obviously, the HDDs in our dataset may not be highly discriminative in other networks.

Table 2: Statistics before (left) and after (right) preprocessing, i. e., purging queries for domains occurring in less than six sessions. Column p_i contains the statistics of the i^{th} percentile, i. e., the most active user issued 28,857,393 queries.

Descriptive Statistic	p_0	p_{25}	p_{50}	p_{75}	p_{100}	p_0	p_{25}	p_{50}	p_{75}	p_{100}
Queries per user	1	24,069	59,368	121,185	28,857,393	1	23,601	58,010	116,833	28,351,070
Domains per user	1	2,107	4,184	7,434	303,568	1	1,833	3,617	6,038	98,470
Active days per user	1	31	43	52	62	1	31	43	52	62
Queries per session	1	572	1,385	2,969	5,117,745	1	561	1,352	2,887	5,112,205
Domains per session	1	196	372	671	258,110	1	190	357	634	38,587
Queries per domain	1	1	2	4	9,781,157	6	17	34	101	9,781,157
Active days per domain	1	1	1	2	62	1	7	12	27	62
Sessions per domain	1	1	1	2	135046	6	8	14	37	135046
Users per domain	1	1	1	2	3812	1	5	8	17	3812
Active users per day	984	2,053	2,438	3,076	3,208	983	2,053	2,438	3,076	3,206

The presence of HDDs simplifies the job of the observer. If an observer was aware of the fact that a user issued queries for a HDD in every session, the observer could trivially link the affected sessions with high confidence. However, while we can determine such domains by looking at the ground truth, this is more difficult in practice. The observer would have to consider the possibility that a potentially HDD is queried by different users on different days. Nevertheless, we conjecture that classification and clustering benefit from the presence of HDDs and will study their utility in Sect. 5.3.2.

5.2 Methods and Metrics

We analyze the tracking techniques for PuC lengths $D \in \{7, 14, 28, 56\}$. For each D, we partition the first 56 days of the dataset into adjacent D-sized PuCs (cf. Sect. 3) and report the tracking accuracy for a simulated observer. Besides the **realistic dataset**, which was characterized in Sect. 5.1, we also consider a **clean dataset**. The clean dataset is derived from the realistic dataset in such a manner that there is *no user fluctuation*. It solely contains the traffic of the subset of the 134 users that are active on *all* of the first 56 days, i. e., the number of users is the same in all experiments (regardless of the value of D). This unrealistically simplified scenario provides an upper bound for tracking accuracy.

Further, we report results from three perspectives: (1) on the global level, (2) on the cluster level, and (3) on the user level. For the **global perspective** we use an established metric for external cluster evaluation. We calculate the **adjusted Rand Index (ARI)** [24], which is derived from the Rand Index (RI) and accounts for chance. RI indicates how similar two clusterings are. We use the ARI to compare a perfect clustering (based on the ground truth) with the predicted clusters. RI is calculated by obtaining all $\binom{n}{2}$ possible pairs of n sessions in a PuC and by determining the fraction of *consensually clustered* pairs. A pair of sessions is consensually clustered if both sessions are in the *same cluster* in the two clusterings or if they are in two *different* clusters in the two clusterings:

$A = \{(x_i, x_j) : i < j, \text{clustering}_1(x_i) = \text{clustering}_1(x_j)$ and $\text{clustering}_2(x_i) = \text{clustering}_2(x_j)\}$, and

$B = \{(x_i, x_j) : i < j, \text{clustering}_1(x_i) \neq \text{clustering}_1(x_j)$ and $\text{clustering}_2(x_i) \neq \text{clustering}_2(x_j)\}$.

Given these two sets ARI is defined as follows:

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}[\text{RI}]}{1 - \mathbb{E}[\text{RI}]} \text{ with } \text{RI} = \frac{\#\{A \cup B\}}{\binom{n}{2}},$$

where the expectation is taken over the uniform distribution over all possible pairs of clusterings. An ARI score of 1 indicates perfect agreement between the clusterings, a uniformly random clustering achieves a score close to 0, and negative scores indicate worse-than-random agreement.

The second perspective, the **cluster-level evaluation**, provides additional insights by looking at two quality measures that may be of interest for an observer, namely *homogeneity* h and *completeness* c [50]. h indicates whether all sessions assigned to a cluster belong to the same user, which is important if the costs of erroneously linking sessions of different users are high. c indicates whether all sessions of a user are assigned to the same cluster, which is important if the costs of missing some sessions of a user are high.

These metrics, which are normalized to the range $[0, 1]$ (larger values being more desirable for an observer), are defined as

$$h = 1 - \frac{H(U|C)}{H(U)} \text{ and } c = 1 - \frac{H(C|U)}{H(C)},$$

where $H(\cdot)$ and $H(\cdot|\cdot)$ denote the entropy and conditional entropy, respectively, U is the set of users, and C is the set of clusters. We will also refer to the v-measure, which is the harmonic mean of h and c: $v = \frac{2hc}{h+c}$. Note, however, that these measures have to be interpreted with care: First, they are not adjusted for chance, i. e., in certain situations random cluster assignment may achieve high scores. Second, considering *only* homogeneity *or* completeness can be misleading: A clustering that assigns each session to a separate cluster (i. e., there are as many clusters as sessions) achieves maximum homogeneity, but scores poorly in terms of completeness. A clustering that assigns all sessions into a single cluster achieves maximum completeness $c = 1$, but scores poorly in terms of homogeneity.

Finally, we are also interested in the traceability of individual users. This kind of analysis is facilitated by the **user-level evaluation**. For this purpose we calculate two anonymity metrics, a_m and b_m, that capture different aspects of how good a user that is active on m days can be tracked within a PuC.

Both metrics are normalized to the range $[0, 1]$. An observer strives to minimize a_m and to maximize b_m. We define

$$a_m(\mathrm{u}) = \frac{N_{\mathrm{cluster}}(u) - 1}{m - 1} \text{ and } b_m(u) = \frac{C_{\max}(u) - 1}{m - 1},$$

where $N_{\mathrm{cluster}}(u)$ equals the number of clusters a user's sessions are assigned to and $C_{\max}(u)$ equals the maximum number of a user's sessions that are assigned to the same cluster.

The informative value of the two metrics is illustrated with the following example. Assume that users u_{Alice} and u_{Bob} are online on $m = 6$ days and the cluster-assignments are $\mathrm{assign}_{\mathrm{Alice}} = \{1, 1, 2, 2, 3, 3\}$ and $\mathrm{assign}_{\mathrm{Bob}} = \{4, 4, 4, 4, 5, 6\}$, respectively. The sessions of both users are spread over three clusters. Intuitively, u_{Bob} can be traced better, because a larger number of his sessions are in one big cluster (cluster 4). The value of b_m reflects this: $b_m(u_{\mathrm{Alice}}) = \frac{2-1}{6-1} = 0.2 < 0.6 = \frac{4-1}{6-1} = b_m(u_{\mathrm{Bob}})$. Note that $a_m(u_{\mathrm{Alice}}) = 0.5 = a_m(u_{\mathrm{Bob}})$ for both users. Now let's consider a different assignment for u_{Alice} and u_{Bob} (again with $m = 6$): $\mathrm{assign}_{\mathrm{Alice}} = \{1, 1, 2, 3, 4, 5\}$ and $\mathrm{assign}_{\mathrm{Bob}} = \{6, 6, 7, 7, 8, 8\}$. In this case we obtain $b_m(u_{\mathrm{Alice}}) = 0.2 = b_m(u_{\mathrm{Bob}})$, but $a_m(u_{\mathrm{Alice}}) = 0.8 > 0.4 = a_m(u_{\mathrm{Bob}})$. This result reflects the intuition that it is more difficult to track u_{Alice}, because her sessions are spread over more clusters.

More formally, we refer to a user that is active in m sessions as *perfectly untraceable*, if his sessions get assigned to m different clusters, i. e., $N_{\mathrm{cluster}}(u) = m$ and $C_{\max}(u) = 1$) and thus $a_m(u) = 1$ and $b_m(u) = 0$ (because $a = 1 \Leftrightarrow b = 0$). Furthermore, we say a user is **completely traceable**, if all his sessions are assigned to a single cluster, i. e., $N_{\mathrm{cluster}}(u) = 1$ and $C_{\max}(user) = m$) and thus $a_m(u) = 0$ and $b_m(u) = 1$ (because $a = 0 \Leftrightarrow b = 1$).

Note that completely traceable users do not necessarily have pure clusters, i. e., their clusters may hold additional sessions that belong to other users (that are not completely traceable). Impure clusters are undesirable for observers that are interested to learn the behavioral profiles of particular users. Therefore, we will also report which fraction of users are **perfectly traceable**, which accounts for completely traceable users whose cluster is pure.

5.3 Results

We begin with results for the *clean dataset*, i. e., the traffic of 134 users that are active on every day, and proceed with results for the *realistic dataset* that contains the traffic of all 3,862 users. Note that the accuracy values published in [18] cannot be compared to our results for 1NNC.

5.3.1 Clean Dataset

We begin with results on the **global level**. Both techniques come up with clusters that are very similar to the ground truth clustering. While the ARI values of 1NNC fluctuate between 0.961 and 0.996, KMC-STRICT achieves values between 0.963 and 1 (KMC-SOFT: 0.937–0.991).

In order to determine the influence of D we averaged the ARI scores over all PuCs with the same length D. The results show that increasing D improves the quality of the clustering. The ARI scores increase from 0.981 ($D = 7$) to 0.997 ($D = 56$) for 1NNC and from 0.984 to 0.994 for KMC-STRICT, respectively. As expected, the ARI values of RNDC are close to zero for all values of D.

According to the **cluster-level** evaluation almost all clusters are homogenous and complete. In all cases, completeness and homogeneity scores were almost identical. h and c range between 0.989 and 0.999 (mean: 0.994) for 1NNC and between 0.990 and 1.000 (mean: 0.996) for KMC-STRICT ($D = 7$). For higher values of D, both scores are always higher than 0.995. The mean v-scores for RNDC are not 0, but decrease from 0.598 ($D = 7$) to 0.231 ($D = 56$).

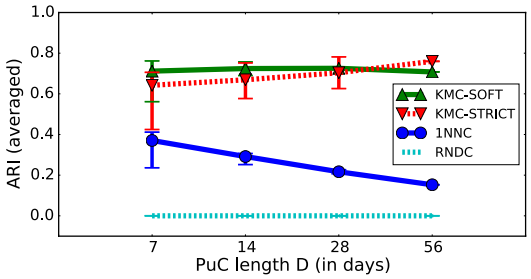


Figure 4: Averaged ARI for varying PuC length D

On the **user level** we find that most users can be tracked perfectly for all values of D. For $D = 7$, 93 % of all users are *perfectly traceable* with 1NNC. The sessions of the remaining users are spread over two clusters ($a_7 = 0.167$). For $D = 14$, 28, and 56 the fraction of perfectly traceable users decreases to 92 %, 91 %, and 91 %. The sessions of the remaining users are still spread over two clusters. KMC-STRICT performs similarly: 96 % of the users are perfectly traceable for $D = 7$. Again, the sessions of the remaining users are spread over two clusters, with the exception of three users, whose sessions are spread over 3–4 clusters in some of the PuCs. For longer PuCs, 97 %, 93 %, and 84 % of the users are perfectly traceable. The b_m values average at 0.99 or higher for all values of D for both 1NNC and KMC-STRICT.

5.3.2 Realistic Dataset

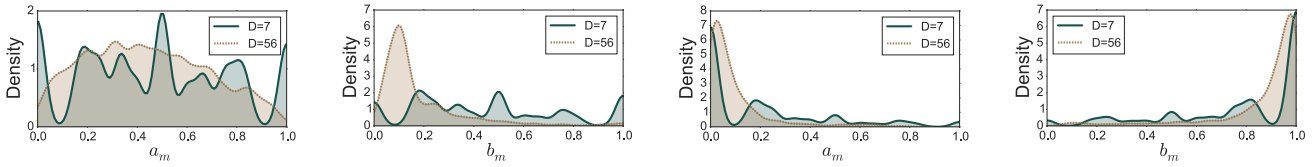
As before we begin with an evaluation on a **global level** for individual PuCs ($D = 7$). Due to the considerably larger number of users, the complexity of the clustering task is now much higher and the quality of the predicted clusterings is considerably lower. As expected, 1NNC performs worse than KMC on the realistic dataset, because it cannot deal with intermittent periods of inactivity resulting from user fluctuation (ARI scores for 1NNC: 0.236–0.412; for KMC-STRICT: 0.425–0.706). This time KMC-STRICT is outperformed by KMC-SOFT (0.561–0.762).

Again, we analyze the influence of D (cf. Fig. 4). In contrast to the clean dataset, the average number of active users is quite stable for all considered values of D in the realistic dataset. As expected, averaged ARI scores of 1NNC decrease when D is increased (from 0.371 for $D = 7$ to 0.153 for $D = 56$), because it becomes more likely that individual users are inactive on at least one day. The performance of KMC depends on the type of restriction in use. While the average ARI scores of KMC-SOFT remain stable (0.708–0.726), the scores of KMC-STRICT improve when D is increased and it eventually outperforms KMC-SOFT (from 0.642 for $D = 7$ up to 0.760 for $D = 56$), despite the number of concurrently active users increasing slightly. Again, the ARI scores of RNDC are close to zero in all cases.

The v-scores obtained in the **cluster-level evaluation** are in line with the global view. The v-score of 1NNC decreases from 0.89 ($D = 7$) to 0.77 ($D = 56$), while KMC-STRICT achieves v-scores between 0.94 ($D = 7$) and 0.92

Table 3: User-level evaluation of realistic dataset: Anonymity scores a_m and b_m for KMC-STRICT, broken down for user subgroups with different level of activity (*low:* users that are active on $\leq 40\,\%$ of days in D; *medium:* 40–70 %; *high:* $> 70\,\%$)

User subgroup	*high activity*				*medium activity*				*low activity*			
PuC length D	7	14	28	56	7	14	28	56	7	14	28	56
Users in subgroup	67 %	60 %	55 %	53 %	27 %	30 %	32 %	35 %	6 %	11 %	12 %	12 %
$a_m \leq 0.1$	52 %	62 %	68 %	73 %	47 %	34 %	43 %	46 %	46 %	32 %	21 %	16 %
$a_m \leq 0.25$	75 %	82 %	89 %	91 %	47 %	63 %	69 %	70 %	46 %	39 %	35 %	27 %
$b_m \geq 0.9$	52 %	60 %	64 %	68 %	47 %	34 %	41 %	44 %	46 %	32 %	21 %	16 %
$b_m \geq 0.75$	73 %	78 %	83 %	86 %	47 %	60 %	64 %	66 %	46 %	38 %	32 %	25 %
Completely traceable	52 %	42 %	29 %	19 %	47 %	34 %	22 %	13 %	46 %	32 %	18 %	11 %
Perfectly traceable	40 %	32 %	21 %	13 %	29 %	21 %	12 %	5 %	21 %	11 %	3 %	1 %



(a) a_m for 1NNC (b) b_m for 1NNC (c) a_m for KMC-STRICT (d) b_m for KMC-STRICT

Figure 5: Distribution of a_m and b_m of highly active users (estimated with kernel density estimator)

($D = 56$). The v-score of the RNDC baseline decreases from 0.79 to 0.55 when D is increased. While the homogeneity score remains relatively stable, the completeness score (especially of 1NNC) decreases when D is increased. In particular, for 1NNC, homogeneity only drops from 0.89 to 0.83, whereas completeness drops from 0.89 to 0.72.

Finally, we report results on the **user level**. For KMC-STRICT, the mean value of a_m (averaged over all users within a PuC and over all PuCs) decreases when D is increased (0.247, 0.210, 0.184, and 0.191), i. e., the sessions of individual users are spread over few clusters even for large values of D. The mean b_m scores remain quite stable when D is increased (0.741, 0.764, 0.778, and 0.765), i. e., sessions of individual users are mostly assigned to a single large cluster. 1NNC performs significantly worse. In contrast to KMC-STRICT the a_m values do not benefit from larger PuCs (0.492, 0.504, 0.513, and 0.514) and the b_m-values decrease with increasing D (0.456, 0.349, 0.242, and 0.161).

The quality of the clustering depends on the level of activity of a user. Therefore, we break down the results for users that are active on less than 40 % (*low*), between 40 % and 70 % (*medium*), and on more than 70 % (*high*) of the days of a PuC. We describe the main trends with the figures for *highly active users*, which make up between 53 % ($D = 56$) and 67 % ($D = 7$) of the users (cf. Table 3 for results of the other groups). Figure 5 shows the distribution of a_m and b_m for the highly active users with 1NNC and KMC-STRICT.

For $D = 7$, 1NNC's predictions make 14 % of the users completely traceable – much less than the 52 % achieved by KMC-STRICT (KMC-SOFT: 52 %). 11 % and 40 % of the users are perfectly traceable with 1NNC and KMC-STRICT, respectively (KMC-SOFT: 42 %). A score of $a_m \leq 0.25$, which corresponds to two or fewer different clusters for $m = 5$, 6, or 7, is achieved for 33 % and 75 % of the users for 1NNC and KMC-STRICT (KMC-SOFT: 75 %). Also, KMC-STRICT achieves a score of $b_m \geq 0.75$ for 73 % of the users, i. e., most sessions of a user are contained within a single large cluster (1NNC: 25 %; KMC-SOFT: 73 %)

For larger values of D, the fraction of perfectly and completely traceable users decreases, because the additional sessions increase the likelihood of at least one wrong assignment. For $D = 14$, KMC-STRICT results in perfect traceability of 32 % of users (1NNC: 5 %; KMC-SOFT: 32 %) and in complete traceability of 42 % of users (1NNC: 6 %; KMC-SOFT: 41 %). In contrast, the fraction of users that is *almost* completely traceable increases for longer PuCs: 62 % of users have $a_m \leq 0.1$ (1NNC: 12 %; KMC-SOFT: 62 %). This trend stays intact up to $D = 56$, where only 13 % of the users are perfectly traceable and only 19 % are completely traceable with KMC-STRICT, but 73 % are almost completely traceable with $a_m \leq 0.1$ (1NNC: 1 %, 1 %, and 8 %; KMC-SOFT: 8 %, 14 %, and 68 %). Further, the fraction of users with $b_m \geq 0.9$ increases to 68 %. Finally, as expected, RNDC performs very poorly: More than 99 % of the users have $a_m > 0.9$ and $b_m < 0.1$ for all values of D.

Highly Discriminative Domains.

Now we analyze to what extent traceability is improved by HDDs (cf. Sect. 5.1.3). To this end, we consider the average fraction of users that is traceable within the PuCs. We compare the fraction f_{HDD} obtained on the subset U_{HDD}, the 101 users that issue queries for HDDs in the first 56 days, with the fraction f_{none} obtained for the remaining users U_{none}. The users in U_{HDD} are not significantly more or less active than the whole user group (cf. "Number of active days within PuC" in Table 4), which is why U_{none} contains all users regardless of their level of activity.

We found that both 1NNC and KMC (cf. Table 4) benefit considerably from the presence of HDDs regardless of the value of D. For 7-day PuCs and complete traceability, we

Table 4: Fraction of completely and perfectly traceable users for the subset of users with HDDs (U_{HDD}) and the subset of all remaining users (U_{none}) obtained with KMC-STRICT

D	7		14		28		56	
Subgroup	U_{HDD}	U_{none}	U_{HDD}	U_{none}	U_{HDD}	U_{none}	U_{HDD}	U_{none}
Number of active days in PuC	5.6	5.2	10.4	10.0	19.5	19.5	36.0	38.2
Completely traceable	72 %	50 %	65 %	38 %	55 %	25 %	45 %	15 %
Perfectly traceable	59 %	35 %	52 %	26 %	43 %	16 %	30 %	8 %

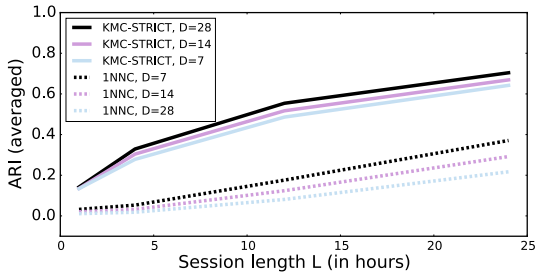


Figure 6: Averaged ARI for varying session length L

observe an improvement from $f_{\text{none}} = 17.5\,\%$ to $f_{\text{HDD}} = 36.8\,\%$ (1NNC) and from $f_{\text{none}} = 50.1\,\%$ to $f_{\text{HDD}} = 71.8\,\%$ (KMC-STRICT), respectively; for $D = 56$, the fractions are $f_{\text{none}} = 0.9\,\%$ and $f_{\text{HDD}} = 7.9\,\%$ for 1NNC and $f_{\text{none}} = 14.8\,\%$ and $f_{\text{HDD}} = 44.6\,\%$ for KMC-STRICT. Moreover, we found that users with HDDs are more likely to be perfectly traceable. For $D = 56$, the corresponding fraction improves from $f_{\text{none}} = 7.9\,\%$ to $f_{\text{HDD}} = 29.7\,\%$ for KMC-STRICT.

Continuously Traceable Users.

So far, we evaluated every PuC separately and reported averaged results, i. e., the reported fraction of 40 % perfectly traceable highly active users ($D = 7$) means that on average 40 % of these users were perfectly traceable *within each of the eight 7-day PuCs*. Based on these results we determined what fraction of users is *continuously* perfectly traceable, i. e., *in every PuC in which they are active*.

This is the case for 7.6 % of the 658 users that were highly active in all of their 7-day PuCs, i. e., all their sessions are assigned to (up to eight) pure clusters. The fraction increases with D as follows: 8.0 % of 1,170 users for $D = 14$, 9.5 % of 1,592 users for $D = 28$, and 13 % of 1,950 users for $D = 56$.

Smaller fractions are observed, if we consider *all* users (regardless of their level of activity). For $D = 7$, we found that 3.3 % of the users are continuously perfectly traceable. Again, the fraction increases with D. We observed fractions of 4.2 %, 5.8 %, and 8.5 % for $D = 14$, 28, and 56. The fractions of continuously *completely* traceable users are consistently higher: For $D = 7$, 9.1 % of the users get assigned into a single (not necessarily pure) cluster, which increases to 10.5 %, 11.7 %, and 15.6 % for larger values of D.

5.3.3 Results for Shorter Sessions

Up to this point we assumed that every user changes their IP address exactly once a day. In the following we provide selected results for smaller session lengths than $L = 24$ hours, in particular for 12-, 4-, and 1-hour sessions. Within the fixed time-frame of a PuC, the simulated observer has to link considerably more sessions than in the previous experiments, namely 14, 42, or 168 sessions for $D = 7$ days and $L = 12$, 4, and 1, respectively. Figure 6 summarizes the results. As expected, the average ARI of KMC-STRICT drops from 0.642 for $L = 24$ to 0.134 for $L = 1$. This sharp decrease is also observed when D is increased. For instance, for $D = 28$, the average ARI value drops from 0.704 ($L = 24$) to 0.142 ($L = 1$). 1NNC performs even worse: Average ARI values decay from 0.371 ($L = 24$) to 0.032 ($L = 1$) for $D = 7$ and the loss of quality is even higher for longer PuCs.

5.3.4 Computational Effort for Clustering

Our experiments were executed on an Intel Xeon E5-2695 v2 48-core (2.40 GHz) machine with a total of 377 GiB of RAM. The reported runtimes are for individual PuCs, i. e., the runtime of an experiment can be obtained by multiplying these runtimes with 8, 4, 2, and 1 for $D = 7$, 14, 28, and 56, respectively. The runtime of KMC mainly depends on D and the number of iterations. KMC never needed more than 40 iterations, most of the time much fewer. The average runtime of KMC per PuC is 16, 44, 147, and 592 minutes for $D = 7$, 14, 28, and 56, respectively. Memory consumption ranges from less than 500 MiB for $D = 7$ to 2 GiB for $D = 56$. 1NNC is considerably faster, with 0.4, 1, 3, and 5 minutes per PuC for increasing values of D and a corresponding memory usage of 300 to 800 MiB. The runtimes of our non-optimized Python code demonstrate that behavior-based tracking does not require much effort.

6. DISCUSSION AND FUTURE WORK

We now summarize our results, discuss the limitations of our study, and present possible extensions.

Generally, the results indicate that unsupervised learning techniques can be employed to build comprehensive behavioral profiles even when no labeled training sessions are available. Apparently, the clustering problem is not very challenging on the *clean* dataset. Thus, an observer that is faced with a relatively small number of users that are known to be active every day can use either 1NNC or KMC for behavior-based tracking with high accuracy. The limitations of the 1NNC approach, which corresponds to day-to-day classification of sessions, become apparent on the *realistic* dataset. KMC outperforms 1NNC by a considerable margin because it maintains a global view over all days within a PuC. KMC turns out to be most effective for users that are active on most of the days. While the best results for complete and perfect traceability are clearly obtained on short PuCs, closely matching clusterings ($a_m \leq 0.1$) can also be achieved for the longest periods we considered.

6.1 Limitations

We believe that our tracking technique can be applied in other settings. However, given our dataset we can only discuss the implications of behavior-based tracking for DNS users. While the dataset has been recorded with care, it does not capture the DNS traffic of all users perfectly (cf. Sect. 5.1 and Fig. 2). Further, multiple users *may* have shared a single IP address (cf. the user with 28,857,393 queries, Table 2), which may have degraded the results.

While we strive for a realistic evaluation, our results should not be used to draw conclusions about the effectiveness of our technique under different conditions. First, the clustering problem becomes more difficult if our assumptions (cf. Sect. 3) are relaxed. For instance, an observer that only has access to DNS traffic may find it difficult to separate the traffic of users that share the same IP address. This applies to households where multiple users are behind a router that performs network address translation. However, determined observers may still be able to separate the traffic of multiple users, either by leveraging context knowledge (such as the time of day) or by leveraging peculiarities of the query behavior to distinguish different devices or browsers. Observers that have access to the TCP traffic of users can also exploit the fact that machines can be re-identified due to their different clock skews [36].

Second, the clustering problem becomes more difficult for large user groups with millions of users. However, DNS resolvers that are accessed by users around the globe can partition their large user group according to the diurnal access patterns into smaller subgroups for different time zones. Observers can also exploit the fact that most users are spatially constrained. They could use the source IP addresses and their approximate geolocations to create even smaller partitions (losing track of users that travel large distances).

6.2 Protection Against Tracking

The results for shorter session lengths in Sect. 5.3.3 indicate that behavior-based tracking efforts can be defeated by changing one's IP address frequently. While this measure is effective in theory, implementing it in practice is cumbersome due to the "always online" paradigm. In order to obtain a new IP address more often than once a day, users will typically have to manually force their broadband router to disconnect and re-connect to their ISP, which terminates all established connections, potentially interrupting pending work. Moreover, this approach is not guaranteed to work as some ISPs re-assign the same IP address upon reconnect. The introduction of IPv6 is a chance to overcome these limitations, because ISPs could hand out a large number of IPv6 address prefixes to each of their customers [17].

Decreasing session lengths is only one of many conceivable protection techniques. For instance, multiple users could choose to share a single source IP address, for instance by using a VPN or a proxy that forwards the DNS queries to the resolver [20]. This move would prevent observers from tracking individual users. Furthermore, users could submit dummy queries to the DNS resolver in order to obfuscate their actually desired queries. However, it has been shown that using randomly selected domains for this purpose is not effective [13, 22], i. e., effective dummy traffic schemes require information about the traffic to be obscured.

As none of these techniques are in widespread use, most Internet users are vulnerable to behavior-based tracking at the moment. Note that techniques that protect against on-path eavesdroppers, e. g., DNSCrypt (https://dnscrypt.org), DNSCurve [9], or DNS over TLS [23], cannot protect against tracking by curious DNS resolvers.

6.3 Improving the Clustering

Finally, we remark that there are numerous ways to improve our technique. Deriving more features from the data (such as timing of queries), applying more sophisticated preprocessing (such as n-grams), and employing other distance metrics (such as $(\mathrm{dist}_{\cos})^p$ with $p > 1$) and clustering techniques (e. g., OPTICS [1], which infers the number of clusters from the data) are promising avenues for future work. Multiple approaches can also be combined to improve the clustering result (*consensus clustering* [15]). Future work could also analyze the security of the framework proposed in [32, 33, 34, 38] and extend the methodology to capture non-euclidean geometries [51] and multiple data types [8, 28, 29, 30, 31, 35] as well as different clustering objectives [54].

7. RELATED WORK

Most of related work assumes that an observer has access to multiple labeled sessions or that the IP address of users remains constant for a long time. Kumpošt and Matyáš [37] study NetFlow logs and re-identify users by comparing the cosine similarity of vectors that contain the number of connections to HTTP(S) and SSH hosts. While they use the same similarity measure, their approach differs from ours: Their features are derived from actual connections (which is more accurate than using DNS queries that are subject to caching), however they correspond to destination IP addresses (which are more ambiguous than domain names). Even with a full month of labeled training data, they report high false-positive rates of 21 % (SSH) and 68 % (HTTP).

Yang [57] proposes to authenticate users based on the visited websites. She considers decision trees, support vector machines, and association rule mining in her study and assumes an e-commerce provider that stores behavioral profiles of every of its customers in a database. When customers sign up, they upload the browsing history of multiple sessions to the provider. When they log in at a later time, they upload their most recent browsing history so that the provider can confirm their identity. Given 300 labeled sessions per user Yang achieves an accuracy of up to 87 % for 100 concurrent users, but reports she was unable to scale up her experiment.

More closely related is the work by Kim and Zhang [27], who derive behavioral fingerprints from DNS queries. They consider the DNS queries issued by approximately 55,000 users within two consecutive weeks. They build patterns for users that are active on at least four of the seven days within the first week, assuming that the DNS traffic of users can be attributed over a full week. They find fingerprints for a subset of 11,921 users and report that 98.74 % of these can be re-identified in the second week.

The work closest to ours is [18] by Herrmann et al. They demonstrate how sessions of users can be linked based on their DNS queries. They report an accuracy of 85.4 % using a supervised learning approach on the dataset that is also used in this paper. However, they are limited to linking sessions of two adjacent days. Further, their accuracy measure is defined in a peculiar way, because they include correct predictions about *inactive* users. The actual fraction of correctly linked sessions is considerably lower at about 70 %.

The analysis by Jain et al. [25] is related to our highly discriminative domains (cf. Sect. 5.1.3). They show that numerous identifiers are being transmitted in the clear, which can be used to link sessions of users by adversaries able to observe network traffic.

8. CONCLUSION

We have presented an unsupervised behavior-based tracking technique that outperforms the state of the art by a considerable margin. In contrast to previous work our technique is able to link a large number of sessions of individual users at once. In our experiments with the DNS queries of 3,862 users, we are able to link almost all sessions of 73 % of the 2,047 highly active users that have issued queries on at least 40 days within a period of 56 days. 13 % of these 2,047 users can be traced *perfectly*, i. e., all their sessions are assigned to a single pure cluster. Tracking is even more effective for shorter periods. An observer that considers periods of seven days can on average almost completely track 75 % of the 2,588 users that are active on at least five days, 40 % of these 2,588 users can be traced perfectly.

On the one hand, behavior-based tracking can be adopted by forensic investigators or law enforcement agencies that want to link the activities of roaming adversaries. On the other hand, we are concerned that ad networks and analytics providers will make use of behavior-based tracking to complement their existing tracking efforts, which are (supposedly) limited to explicit identifiers such as cookies and browser or device fingerprints so far. This is worrying, as behavior-based tracking takes place purely on the server side and thus cannot be detected on the client. Therefore, we hope that ISPs and manufacturers of broadband routers will cooperate to come up with usable solutions to protect customers against this threat.

Acknowledgments

We are grateful to M. Maaß, H. Pridöhl, and to the developers of *scikit-learn*. Marius Kloft acknowledges support from the German Research Foundation (DFG) award KL 2698/2-1 and from the Federal Ministry of Science and Education (BMBF) awards 031L0023A and 031B0187B.

9. REFERENCES

[1] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. In *ACM SIGMOD Int'l Conf. on Management of Data*, pages 49–60. ACM, 1999.

[2] APNIC Labs. DNSSEC Validation, 2016. http://stats.labs.apnic.net/dnssec, accessed: 2016-09-19.

[3] M. Ayenson, D. J. Wambach, A. Soltani, N. Good, and C. J. Hoofnagle. Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning. Available at SSRN: http://ssrn.com/abstract=1898390, 2011.

[4] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[5] C. Banse, D. Herrmann, and H. Federrath. Tracking Users on the Internet with Behavioral Patterns: Evaluation of Its Practical Feasibility. In *IFIP SEC*, volume 376 of *AICT*, pages 235–248. Springer, 2012.

[6] S. Bortzmeyer. DNS Privacy Considerations. RFC 7626, 2015.

[7] CircleID. Google Improving Location-Sensitive DNS Responses for Its 400B Responses-Per-Day Public DNS Service, 2014. http://www.circleid.com/posts/google_improving_location_sensitive_dns_responses/, accessed: 2016-09-19.

[8] C. Cortes, M. Kloft, and M. Mohri. Learning kernels using local rademacher complexity. In *NIPS*, pages 2760–2768, 2013.

[9] M. Dempsky. DNSCurve: Link-Level Security for the Domain Name System. Draft, RFC Editor, 2010.

[10] I. S. Dhillon and D. S. Modha. Concept Decompositions for Large Sparse Text Data Using Clustering. *Machine Learning*, 42(1/2):143–175, 2001.

[11] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten. Cookies That Give You Away: The Surveillance Implications of Web Tracking. In *WWW*, pages 289–299. ACM, 2015.

[12] Federal Trade Commission. Self-Regulatory Principles For Online Behavioral Advertising, 2009. https://www.ftc.gov/reports/federal-trade-commission-staff-report-self-regulatory-principles-online-behavioral, accessed: 2016-09-19.

[13] H. Federrath, K.-P. Fuchs, D. Herrmann, and C. Piosecny. Privacy-Preserving DNS: Analysis of Broadcast, Range Queries and Mix-Based Protection Methods. In *ESORICS*, volume 6879 of *LNCS*, pages 665–683. Springer, 2011.

[14] S. Gebert, R. Pries, D. Schlosser, and K. Heck. Internet Access Traffic Measurement and Analysis. In *Proc. Traffic Monitoring and Analysis (TMA 2012)*, volume 7189 of *LNCS*, pages 29–42. Springer, 2012.

[15] A. Goder and V. Filkov. Consensus Clustering Algorithms: Comparison and Refinement. In *Proc. 10th Workshop on Algorithm Engineering and Experiments*, pages 109–117. SIAM, 2008.

[16] C. Grothoff, M. Wachs, M. Ermert, and J. Appelbaum. NSA's MORECOWBELL, 2015. https://gnunet.org/morecowbell, accessed: 2016-09-19.

[17] D. Herrmann, C. Arndt, and H. Federrath. IPv6 Prefix Alteration: An Opportunity to Improve Online Privacy. *CoRR*, abs/1211.4704, 2012. Available at http://arxiv.org/abs/1211.4704.

[18] D. Herrmann, C. Banse, and H. Federrath. Behavior-based Tracking: Exploiting Characteristic Patterns in DNS Traffic. *Computers & Security*, 39A:17–33, Nov. 2013.

[19] D. Herrmann, K. Fuchs, and H. Federrath. Fingerprinting Techniques for Target-oriented Investigations in Network Forensics. In *Sicherheit*, volume 228 of *LNI*, pages 375–390. GI, 2014.

[20] D. Herrmann, K. Fuchs, J. Lindemann, and H. Federrath. EncDNS: A Lightweight Privacy-Preserving Name Resolution Service. In *ESORICS*, volume 8712 of *LNCS*, pages 37–55. Springer, 2014.

[21] D. Herrmann, C. Gerber, C. Banse, and H. Federrath. Analyzing Characteristic Host Access Patterns for Re-identification of Web User Sessions. In *NordSec*, volume 7127 of *LNCS*, pages 136–154. Springer, 2010.

[22] D. Herrmann, M. Maaß, and H. Federrath. Evaluating the Security of a DNS Query Obfuscation Scheme for

Private Web Surfing. In *IFIP SEC*, volume 428 of *AICT*, pages 205–219. Springer, 2014.

[23] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, 2016.

[24] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[25] S. Jain, M. Javed, and V. Paxson. Towards mining latent client identifiers from network traffic. *PoPETs*, 2016(2):100–114, 2016.

[26] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Trans. Netw.*, 10(5):589–603, 2002.

[27] D. W. Kim and J. Zhang. You are how you query: Deriving behavioral fingerprints from DNS traffic. In *SecureComm*, volume 164 of *LNICST*, pages 348–366. Springer, 2015.

[28] M. Kloft, U. Brefeld, P. Düessel, C. Gehl, and P. Laskov. Automatic feature selection for anomaly detection. In *AISec*, pages 71–76. ACM, 2008.

[29] M. Kloft, U. Brefeld, P. Laskov, K.-R. Müller, A. Zien, and S. Sonnenburg. Efficient and accurate lp-norm multiple kernel learning. In *NIPS*, pages 997–1005, 2009.

[30] M. Kloft, U. Brefeld, P. Laskov, and S. Sonnenburg. Non-sparse multiple kernel learning. In *NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, volume 4, 2008.

[31] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. Lp-norm multiple kernel learning. *JMLR*, 12(Mar):953–997, 2011.

[32] M. Kloft and P. Laskov. A poisoning attack against online anomaly detection. In *NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security*, 2007.

[33] M. Kloft and P. Laskov. Online anomaly detection under adversarial impact. In *AISTATS*, pages 405–412, 2010.

[34] M. Kloft and P. Laskov. Security analysis of online centroid anomaly detection. *JMLR*, 13(Dec):3681–3724, 2012.

[35] M. Kloft, U. Rückert, and P. L. Bartlett. A unifying view of multiple kernel learning. In *ECML-PKDD*, pages 66–81. Springer, 2010.

[36] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. *IEEE Trans. Dependable Sec. Comput.*, 2(2):93–108, 2005.

[37] M. Kumpošt and V. Matyáš. User Profiling and Re-identification: Case of University-Wide Network Analysis. In *TrustBus*, volume 5695 of *LNCS*, pages 1–10. Springer, 2009.

[38] P. Laskov and M. Kloft. A framework for quantitative security analysis of machine learning. In *AISec*, pages 1–4. ACM, 2009.

[39] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press, 1967.

[40] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband internet traffic. In *IMC*, pages 90–102. ACM, 2009.

[41] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[42] P. Mockapetris. Domain Names: Concepts and Facilities. RFC 1034, 1987.

[43] P. Mockapetris. Domain Names: Implementation and Specification. RFC 1035, 1987.

[44] K. Mowery and H. Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In *Proc. of Web 2.0 Security and Privacy (W2SP 2012)*. IEEE, 2012.

[45] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. Web Page Revisitation Revisited: Implications of a Long-Term Click-Stream Study of Browser Usage. In *CHI*, pages 597–606. ACM, 2007.

[46] L. Olejnik and C. Castelluccia. Towards Web-based Biometric Systems Using Personal Browsing Interests. In *ARES*, pages 274–280. IEEE, 2013.

[47] L. Olejnik, C. Castelluccia, and A. Janc. Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns. In *HotPETs*, 2012.

[48] OpenDNS System, 2016. https://system.opendns.com/, accessed: 2016-09-19.

[49] P. Richter, N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger. Distilling the Internet's Application Mix from Packet-Sampled Traffic. In *PAM*, volume 8995 of *LNCS*, pages 179–192. Springer, 2015.

[50] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

[51] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[52] K. Schomp, M. Rabinovich, and M. Allman. Towards a model of DNS client behavior. In *PAM*, volume 9631 of *LNCS*, pages 263–275. Springer, 2016.

[53] L. Tauscher and S. Greenberg. How people revisit web pages: empirical findings and implications for the design of history systems. *Int. J. Hum.-Comput. Stud.*, 47(1):97–137, 1997.

[54] J. E. Vogt, M. Kloft, S. Stark, S. S. Raman, S. Prabhakaran, V. Roth, and G. Rätsch. Probabilistic clustering of time-evolving distance data. *Machine Learning*, 100(2-3):635–654, 2015.

[55] I. H. Witten and E. Frank. *Data Mining. Practical Machine Learning Tools and Techniques*. Elsevier, San Francisco, 2005.

[56] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are IP addresses? In *SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 301–312. ACM, 2007.

[57] Y. Yang. Web user behavioral profiling for user identification. *Decis. Support Syst.*, 49:261–271, 2010.

[58] Z. Yu, S. Macbeth, K. Modi, and J. M. Pujol. Tracking the Trackers. In *WWW*, pages 121–132, 2016.