Se Eun Oh*, Shuai Li, and Nicholas Hopper

Fingerprinting Keywords in Search Queries over Tor

Abstract: Search engine queries contain a great deal of private and potentially compromising information about users. One technique to prevent search engines from identifying the source of a query, and Internet service providers (ISPs) from identifying the contents of queries is to query the search engine over an anonymous network such as Tor.

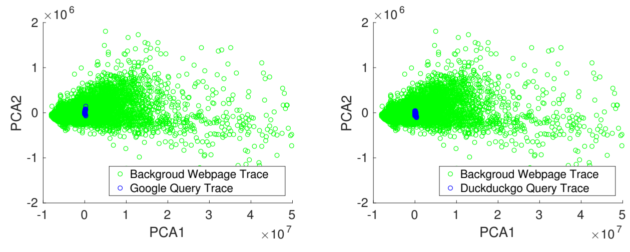
In this paper, we study the extent to which Website Fingerprinting can be extended to fingerprint individual queries or keywords to web applications, a task we call Keyword Fingerprinting (KF). We show that by augmenting traffic analysis using a two-stage approach with new task-specific feature sets, a passive network adversary can in many cases defeat the use of Tor to protect search engine queries.

We explore three popular search engines, Google, Bing, and Duckduckgo, and several machine learning techniques with various experimental scenarios. Our experimental results show that KF can identify Google queries containing one of 300 targeted keywords with recall of 80% and precision of 91%, while identifying the specific monitored keyword among 300 search keywords with accuracy 48%. We also further investigate the factors that contribute to keyword fingerprintability to understand how search engines and users might protect against KF.

DOI 10.1515/popets-2017-0048
Received 2017-02-28; revised 2017-06-01; accepted 2017-06-02.

1 Introduction

Internet search engines are the most popular and convenient method by which users locate information on the web. As such, the queries a user makes to these search engines necessarily contain a great deal of private and personal information about the user. For example, AOL search data leaked in 2006 was found to contain queries revealing medical conditions, drug use, sexual assault victims, marital problems, and political leanings [18, 38]. Thus, popular search engines such as



(a) Google **(b)** Duckduckgo

Fig. 1. Principal Components Analysis (PCA) Plot of Google and Duckduckgo query traces and background webpage traces based on CUMUL feature set

Bing and Google, and ISPs, are in a position to collect sensitive details about users. These search queries have also been among the targets of censorship [29] and surveillance infrastructures [17] built through the cooperation of state and private entities.

One mechanism to conceal the link between users and their search queries is an anonymous network such as Tor, where the identities of clients are concealed from servers and the contents and destinations of connections are concealed from network adversaries, by sending connections through a series of encrypted relays. However, even Tor cannot always guarantee connection anonymity since the timing and volume of traffic still reveal some information about user browsing activity. Over the past decade, researchers have applied various machine learning techniques to features based on packet size and timing information from Tor connections to show that Website Fingerprinting (WF) attacks can identify frequently visited or sensitive web pages [7, 14, 20, 30, 31, 44].

In this paper, we describe a new application of these attacks on Tor, Keyword Fingerprinting (KF). In this attack model, a local passive adversary attempts to infer a user's search engine queries, based only on analysing traffic intercepted between the client and the entry guard in the Tor network. A KF attack proceeds in three stages. First, the attacker must identify which Tor connections carry the search result traces of a particular search engine against the other webpage traces. The second is to determine whether a target query trace is in a list of "monitored queries" targeted for identification. The third goal is to classify each query trace correctly to predict the query (keyword) that the victim typed.

Note that while KF and WF attacks share some common techniques, KF focuses on the latter stages of

***Corresponding Author: Se Eun Oh:** University of Minnesota, E-mail: seoh@umn.edu
Shuai Li: University of Minnesota, E-mail: shuai@cs.umn.edu
Nicholas Hopper: University of Minnesota, E-mail: hoppernj@umn.edu

this attack, distinguishing between multiple results from a single web application, which is challenging for WF techniques. Figure 1 illustrates this distinction; it shows the website fingerprints (using the CUMUL features proposed by Panchenko *et al.* [30]) of 10,000 popular web pages in green, and 10,000 search engine queries in blue. Search engine queries are easy to distinguish from general web traffic, but show less variation between keywords. Hence, while direct application of WF performs the first step very well and can perform adequately in the second stage, it performs poorly for differentiating between monitored keywords. Thus, the different level of application as well as the multi-stage nature of the attack make it difficult to directly use or compare results from the WF setting.

The primary contribution of this paper is to demonstrate the feasibility of KF across a variety of settings. We collect a large keyword dataset, yielding about 160,000 search query traces for monitored and background keywords. Based on this dataset, we determine that WF features do not carry sufficient information to distinguish between queries, as Figure 1 illustrates. Thus we conduct a feature analysis to identify more useful features for KF; adding these features significantly improves the accuracy that can be achieved in the second and third stages of the attack.

Using these new features, we explore the effectiveness of KF, by considering different sizes of "monitored" and "background" keyword sets; both incremental search (e.g. Google Instant[1]) and "high security" search (with JavaScript disabled); across popular search engines — Google, Bing, and Duckduckgo; and with different classifiers, support vector machines (SVM), k-Nearest Neighbor, and random forests (k-fingerprinting). We show that well-known high-overhead WF defenses [5, 6] significantly reduce the success of KF attacks but do not completely prevent them with standard parameters. We examine factors that differ between non-fingerprintable (those with recall of 0%) and fingerprintable keywords. Overall, our results indicate that use of Tor alone may be inadequate to defend the content of users' search engine queries.

1 Google Instant is a predictive search for potential responses as soon as or before the user types keywords into the search box. By default, Google Instant predictions are enabled only when the computer is fast enough.

2 Threat Model

Our attack model for KF follows the standard WF attack model, in which the attacker is a passive attacker who can only observe the communication between the client and the Tor entry guard. He cannot add, drop, modify, decrypt packets, or compromise the entry guard. The attacker has a list of monitored phrases and hopes to identify packet traces in which the user queries a search engine for one of these phrases, which we refer to as *keywords*. We also assume that search engine servers are uncooperative and consequently users relay their queries through Tor to hide the link between previous and future queries. Thus, our threat model only monitors traces entering through the Tor network.

The attacker will progress through two sequential fingerprinting steps. First, he performs website fingerprinting to identify the traffic traces that contain queries to the targeted search engine, rather than other webpage traffic. Traffic not identified by this step is ignored. We refer to the remaining traces, passed to the next step, as *search query traces* or *keyword traces*. Second, the attacker performs KF to predict keywords in query traces. The attacker will classify query traces into individual keywords, creating a multiclass classifier. Depending on the purpose of the attack, the attacker trains classifiers with either binary or multiclass classifications. For example, binary classification will detect monitored keywords against background keywords while multiclass classification will infer which of the monitored keywords the victim queried.

We assume the adversary periodically re-trains the classifiers used in both steps with new training data; since both training and acquiring the data are resource-intensive, we used data gathered over a three-month period to test the generalizability of our results.

3 Data Set

3.1 Data Collection

We describe the Tor traffic capture tool and data sets collected for our experiment. To collect search query traces on Tor, we used the Juarez *et al.* Tor Browser Crawler [22, 39], built based on Selenium, Stem, and Dumpcap to automate Tor Browser browsing and packet capture.

We added two additional features to the crawler for our experiments. The first is to identify abnormal Google responses due to rate limiting and rebuild the Tor circuit. When an exit node is rate-limited by Google, it responds (most often) with a CAPTCHA, thus, we added a mechanism to detect these cases and reset the Tor process, resulting in a different exit node.

Among 101,685 HTML files for Google queries and 125,245 HTML files for Bing, we manually inspected those of size less than 100KB, yielding 3,315 HTML responses for Google and 9,985 for Bing, and found that a simple size threshold of 10KB for the response document was sufficient to distinguish rate-limit responses from normal results with perfect accuracy. In contrast, seemingly more sophisticated methods such as testing for the standard response URL (ipv4.google.com/sorry) or for an embedded CAPTCHA script (google.com/recaptcha/api.js) produced occasional false negatives when a rate-limited node received other abnormal results (such as permission denied).

The second additional feature is to simulate the user's keyword typing in the search box. Google supports auto-complete and Google Instant for faster searches. This can lead to unintentional incremental searches. For instance, as shown in Appendix A as Figure 7, when the user loads www.google.nl and types "kmart", the result for "kmar" will appear just before the user types 't'. Therefore, to support more realistic user actions, we choose a delay of d seconds uniformly at random from the interval $(0.1, 0.7)$ before typing each letter. Since Google Instant sometimes does not show the result as soon as typing the last letter, we enforce typing RETURN after the last letter is typed. As a consequence, traffic related to suggestion lists, auto-complete, and Google Instant is captured all together with the actual search result traffic.

Since result pages can dynamically modify their content after loading, we wait 5 seconds after the result page is loaded, and then pause 5–10 additional seconds between queries to capture all traffic related to changes made to the result webpage after loading completes. To collect query traces of monitored keywords, we recorded up to 110 *instances* of each keyword, evenly divided among 6 *batches*. Each batch starts with a list of remaining keyword instances, and repeatedly selects an instance at random from the list, queries the keyword, and removes it from the list until no instances remain. Background keyword traces were collected in 1 batch with 2 instances per keyword. On average, the time gap between batch groups was 3 days.

With these additions and settings in the crawler, we collected several sets of Tor traces to be used in our experiment, as summarized in Table 1; they were collected from March to October in 2016. We used the Tor Browser version 4.0.8. (We further collected AOL search query traces from January to February in 2017.)

Table 1. Data collection settings for Google, Bing, and Duckduckgo (inc:incremental, one:one-shot, Y:Yes, N:No)

keyword set	Google(inc)	Google(one)	Bing	Duck
Top-ranked	Y	Y	Y	Y
Blacklisted	Y	Y	N	N
AOL	Y	N	N	N

3.1.1 Keywords

We used the following three different keyword datasets to investigate the performance of KF across different sets of monitored search queries. Note that we collected "search query traces," labelled them with the corresponding keywords for supervised learning (using the classifiers discussed in section 5) and inferred "keywords" based on predicted labels returned by classifiers. For this reason, we refer to KF as keyword fingerprinting rather than search query fingerprinting.

Top-Ranked Keywords. We harvested 300 *parent* keywords by identifying the top 20 ranked keywords for each alphabet character, a–z, based on Google's auto-complete [24] (more specifically, http://keywordtool.io via Tor). We collected the top 650–750 suggested keywords for each of 100 parent keywords selected randomly from these 300 keywords. This yielded a set of about 80,000 keywords to be used as the background set. For example, if we have a parent keyword, "airline", its suggested keywords returned by auto-complete are "airline pilot" and "airline ticket purchase."

AOL Search Queries. As another source of monitored and background queries, we downloaded anonymized AOL search logs provided by Dudek [13], and extracted the queries from these logs. The logs are split among 10 files, where each user's queries appear in one log. We randomly partitioned these logs into two sets, one from which we randomly selected 100 queries to serve as monitored keywords and the other from which we randomly selected 40,000 queries to serve as background keywords.

Google Blacklisted Keywords. Several previous studies have collected lists of keywords blocked by Google. Among them, we used Google blacklisted keywords reported by www.2600.com website [16] to gather Google blacklisted keywords. For these keywords, searches using Google Instant fail. However, if submitted as normal queries, we were sometimes able to get Google query results for those blacklisted keywords, and sometimes the Google result indicated that the content was blocked. This would allow KF to be used to identify "blacklisted" keywords.

Background sets. Note that in both the "top-ranked" and AOL data sets, the maximum size of background keyword set we could use for an experiment was 80,000 queries. This is roughly 1–2 seconds worth of worldwide query traffic processed by Google [1], and in line with or larger than the size of background sets used in open-world studies of WF, which generally target coverage of a much broader background space. The use of two differently generated background sets also lends confidence that our results will generalize.

3.1.2 Two Search Query Settings

We considered the following different query settings. Note that we collected Google traces in both settings, Bing traces in the incremental query setting, and Duckduckgo traces in the one-shot query setting.

One-shot Query Setting. The Tor Browser download page [2] recommends disabling JavaScript (via configuring Noscript(S) to "Forbid scripts globally" for later versions than Tor Browser Bundle 3.5, or "about:config") in order to provide better anonymity and security. To reflect this, we collected traces of the queries introduced above assuming that there is no interaction with a search box. This was achieved for both Duckduckgo and Google by directly requesting a search query url. (e.g., google.com/search?q="keyword" for Google). Hence, collected traces contain packets for requesting search result HTML and responding with HTML, and requesting embedded web objects in HTML and responding with the corresponding embedded contents. This also mimics the process of typing a query in the search or address bar of Tor Browser when DuckDuckGo is selected as the default search engine.

Incremental Query Setting. By default, Tor browser enables JavaScript and therefore, traffic related to the user's interaction with the search box has to be captured in addition to all traffic from the previous setting. For Bing, this addition is the traffic related to the suggestion list. For Google, all traffic related to request and response in incremental search results shown in Figure 7 is additionally captured. Including results of both query types allows us to test whether incremental search improves the accuracy of KF.

3.2 Data Preparation

We reconstructed full TLS records using Tshark [40], similar to T. Wang and Goldberg's work [44]. In addition, we removed faulty packets if they were empty, lacked TLS segments, or if the capture file was cut short in the middle of a packet. The latter is a result message from Tshark when a capture file is not yet flushed out, before a copy is made, in which case the end of the file is not a proper record, since we saved all capture files during a live capture. To correct for occasional out-of-order delivery and re-transmitted packets, we re-ordered all packets according to the TCP sequence number.

4 Feature Analysis

After collecting the query traces, we extracted features from the traces to use in classifier experiments. We computed many features previously identified as useful for website fingerprinting, including total number of incoming and outgoing packets and cells, Tor cell traces, rounded TCP and TLS traces, unique packet sizes, outgoing burst data, and cumulative TLS records; for a full description of these features, see Appendix B.

4.1 Additional Features

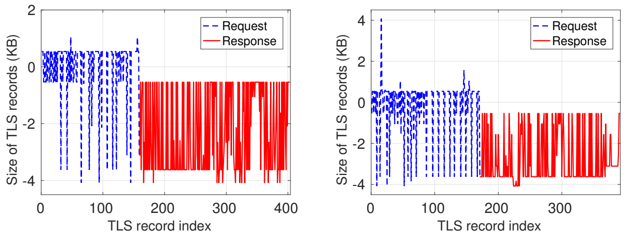
Since search engine result pages often embed CSS, script and advertisement elements, search query traces typically contain multiple request and response pairs; however, the number of such pairs is on average less than those in popular webpages carrying interactive and multimedia content. Additionally, in the incremental setting, after the web browser requests each character in the query, the server responds accordingly to show an updated suggestion list. Subsequently, the web browser requests an HTML document with a keyword and the server responds. Then the web browser requests any embedded web objects such as images. If Google Instant is enabled for Google searches, these interactions are repeated several times. Since the HTML responses are generated by a single programmatic template, the overall timing and size patterns do not have much distinguishing power. As a result, the KF phase will need access to more fine-grained features based on individual TLS records. Thus our experimental results in Section 6 consider the following sets of additional features.

Burst of incoming packets. Based on the concept of "burst" suggested by T. Wang *et al.* [43], but not exactly following their approach, we defined a burst of incoming packets as a sequence of incoming packets comprising more than 2 incoming packets in which there are no outgoing packets. For each such burst, we computed the total number of packets, mean, maximum, and sum of the TLS record sizes (burstIncoming).

Cumulative TLS data in the response for embedded objects (Resp). All query traces in our dataset include a giant sequence of incoming packets, which is

Table 2. Comparison of the request and response portions of Search Query Traces

Metric	Google		DuckDuckgo	
	RQ	RP	RQ	RP
Avg of # of packets	140	223	102	193
Max # of packets	288	559	251	801
Avg of total payload(KB)	115	496	89	434
Max of total payload(KB)	350	1246	295	1669
SVM Accuracy(%)	13.9	17.2	14.7	20.8



(a) Keyword pizza **(b)** Keyword craigslist

Fig. 2. TLS records in two Google query traces. (+) indicates outgoing packets and (-) indicates incoming packets

red in Figure 2 and occupies more than 50% of packets in the trace (see Table 2). We identify this sequence as the largest incoming burst in a query trace, and call it the "response" portion of the trace, while the sequence before the response portion is the "request" portion.

In the request portion, we are able to capture traffic related to requesting and downloading the HTML response, and requesting the embedded objects. Furthermore, the request portion includes traffic generated by the user's interaction with the search box, e.g. suggestion lists and preliminary HTML results in the case of Google Instant. However, since search query traces from a single web application follow a very similar HTML template and have similar traffic pattern for the interaction with the search box and the predicted search if they are same keywords, we expect this portion of the sequence to be less informative about queries than the response portion. Table 2 details results of a small-scale study to confirm this intuition, in which we collected 100 traces for 100 keywords and trained multiclass classifiers on the request and response portions of the traces, respectively. The response portion achieved higher accuracy, 17% compared to 14% of the request portion in Google. When reversing the sequence, as discussed in the next section, the distinction becomes much greater.

Based on this observation, we extracted feature sets from the response portion: we created the tuple RespTotal consisting of the total number of TLS records, maximum TLS record size, average TLS record size, and sum of TLS record sizes; the sequence of TLS record sizes (RespTLS); the sequence of cumulative sizes of TLS records (cumulRespTLS); and the sequence of the corresponding number of Tor cells (cumulRespTorCell).

For example, if the response portion of a query trace consists of three TLS records of sizes 2080, 3108, and 1566, then the RespTLS feature vector is $(-2080, -3108, -1566)$ (following the convention that sign indicates packet direction). The corresponding cumulRespTLS feature vector is $(-2080, -5188, -6754)$ and cumulRespTorCell is $(4, 10, 13)$. In the following sections, we refer to features extracted from the response portion by prefixing them with Resp; as we show in Section 6, feature sets in Resp as well as aggregated feature sets including Resp outperform existing feature sets for the new, second and third stage classification.

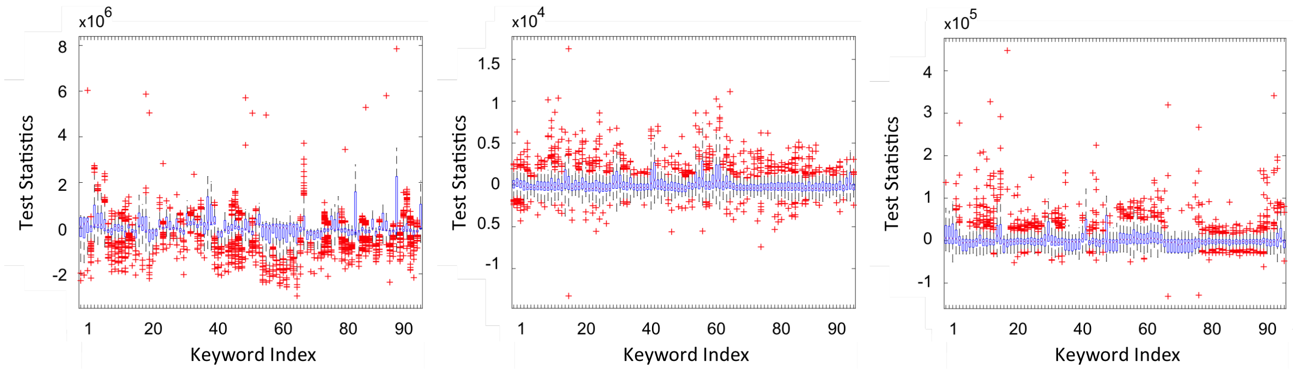
4.2 Preprocessing

Sequence Reversal. Since both SVM and k-NN classifiers require all input vectors to have the same dimension, we additionally reversed the sequence of cumulative record sizes and tor cells, so that truncation would preserve the end of the sequence, which cumulatively includes information about the earlier portions of the sequence. To show that this improves accuracy versus early truncation, for cumulRespTLS, we computed the accuracy of an SVM classifier when trained on the first 140 packets of both the original and reversed cumulative traces, for a test set of 100 instances of 100 keywords. As a result, the reversed sequences, RcumulRespTLS, gave us better accuracy (53.79%, compared to 21.33% when using truncated cumulRespTLS). Therefore, we used RcumulRespTLS and RcumulRespTorCell for aggregated feature sets, rather than cumulRespTLS and cumulRespTorCell.

4.3 Feature Evaluation

There are several statistical methods to compare the distributions of two sample populations. The Kolmogorov-Smirnov two-sample test decides if two datasets are from the same distribution by comparing their empirical distribution functions and the Mann-Whitney U test supports the comparison of two groups of continuous, non-normally distributed data. In contrast, the the Kruskal-Wallis H Test [25] is a widely used non-parametric technique to test for statistically-significant differences between *multiple* groups of continuous data, using ranks for each feature instead of actual values.

Since comparing features extracted from keyword traces involves comparing more than two groups, we decided to use the Kruskal-Walls H test to determine



(a) Aggr4 (b) roundedTCP (c) cumulTLS

Fig. 3. Mean Ranks Distribution of Aggr4, roundedTCP, and cumulTLS. Computation of these results is explained in Appendix B.

what set of features to use for classification in the KF phase. We applied it to various combinations of the feature vectors described in the previous section, for a data set consisting of 100 instances of 100 keywords. We ran the Kruskal-Wallis H Test for each feature set, which eventually returns an ANOVA table consisting of sum of squares, degrees of freedom, H, and p-value for each keyword group. Since there are g entries in the ANOVA table, (where g is the number of keywords in the data set) H is treated as a χ^2 statistic with $g-1$ degrees of freedom to determine the p-value. Given the translation into ranks, H is computed as

$$H = \frac{12}{N(N+1)} \times \Sigma \frac{TR_g^2}{N_g} - 3 \times (N+1) \text{ , where}$$

– N is the total number of features
– TR_g is the rank total for each group
– N_g is the number of features in each group

Because of the translation to rank data, mean ranks of groups are comparable across feature sets. For features that are well-separated, we expect more variation in mean rank across groups (since the features within a group will all have similar rank), and for features that have low distinguishing power we expect the mean ranks of each keyword group to be more similar.

The full results of this test, for the dataset consisting of 100 instances each of 100 keywords, appear in Appendix B as Table 16; most of the features gave a p-value of less than 0.01, but H values for some classic features were not as high as for the new features described above. Based on these results, we decided to test different combinations of feature sets whose H (after scaling for differences in dimensionality) was higher than 6,000. As expected, when we included new feature sets Resp and burstIncoming, we found a higher variance between each group. To illustrate this, we computed TR_g^2/N_g, for each keyword group in three feature sets: Aggr4, which combines the Total, cumulRespTLS, RespTotal, and cumulRespTorCell features; roundedTCP; and cumulTLS). As shown in Figure 3, it is clear that compared to Figure 3a, all keyword groups in Figure 3b and Figure 3c are close to each other, which prevents differentiating keyword groups based on this feature.

In the end, we found that it is least likely that samples in each group in Aggr4, aggregated based on Total, RespTotal, RcumulRespTLS, and RcumulRespTorCell, are from the same distributions, which leads to better keyword classification results.

4.4 Feature Dimensions

Because of the need to truncate feature vectors to a fixed dimension and because longer feature vectors linearly increase the computational cost of training both k-NN and SVM classifiers, we ran a separate experiment to determine the dimension, n_{best}, that gives the best tradeoff between accuracy and training time, similarly to Panchenko *et al.* [30].

As Aggr4 was determined to show the highest variation across keywords, we used this feature set to see the relationship between the number of features and corresponding computational cost. To discover n_{best}, we varied the number of features composing Aggr4 by varying the number of features in RcumulRespTLS and RcumulRespTorCell before aggregation since Total and RespTotal have fewer than 5 features. We trained the SVM with 10,000 top-ranked keyword traces and used 24 python workers, which will be discussed in Section 5, to parallelize the 10-fold cross validation (Note that we used 24 CPUs and 32GB memory for this analysis). This work is similar to Panchenko *et al.*'s work [30]; however, we additionally compared training time to get a more specific result and determine the trade-off between performance and computational cost.

Finally, based on the results shown in Figure 8 in Appendix D, we decided to use 247 features as it gave the best accuracy as well as acceptable running time, since increasing the dimension of the feature vectors above 250 did not yield better accuracy, while linearly increasing the running time. In addition, note that since training classifiers is an offline process and we use a three-month period training (as discussed in Section 7), 188 seconds is acceptable for training.

5 Classification

5.1 Classifiers

We tested three different classifier algorithms for the second and third phase of KF attacks: Support Vector Machines (SVM), k-Nearest Neighbor (k-NN), and k-Fingerprinting (k-FP). All three are supervised learning techniques, which train classifiers on labelled data using fixed-dimensional feature vectors. We evaluated all three classifiers using the Aggr4 feature set with dimension 247 and 10-fold cross-validation during testing; the results are summarized in Table 3.

Support Vector Machines Many researchers have used SVMs to construct effective fingerprinting attacks [7, 30, 44]. The SVM algorithm finds the maximum-margin hyperplane in a high dimensional space to which we map our samples, which gives the largest distance to the nearest training-data point for all classes.

In our experiment, we used a non-linear classifier with a radial basis function (RBF) and n-fold cross-validation to determine the C and γ leading to the highest accuracy, which are inputs of the RBF. We varied C between 0.0078125 and 128, and γ from 0.03125 to 4. The cross-validation accuracy refers to computing the number of examples in each fold that were correctly classified. In Section 6, we ran 10-fold cross validation to avoid over-fitting as well as to compute the overall metrics more correctly during testing. In addition, since the cross validation is the most expensive operation, we parallelized it using multiple python workers supported by the Libsvm library [9]. We used 16 workers when the size of background classes was up to 10,000 and 24 workers when it was more than that.

k-Nearest Neighbors T. Wang *et al.* [43] used k-NN classification with weighted ℓ_1 distance to conduct website fingerprinting attacks. In this technique, a set of labelled training points are first used to learn weights for the distance metric d. Then a set of labelled classification points are used to classify new feature vectors by finding the k closest classification points and comparing the labels of these neighbors; Wang *et al.* classify a page as belonging to class C only if all k neighbors belong to C. We used the same training parameters as in Wang *et al.*'s work [43] with our feature vectors to compare classifier performance.

k-Fingerprinting Hayes and Danezis [19] developed a novel technique that they refer to as k-Fingerprinting. In k-Fingerprinting a random forest ensemble of decision trees is first trained on labelled feature vectors. Then each labelled training trace is translated to a fingerprint vector by taking the classification produced by each tree in the forest. Finally, a new trace is classified by computing its fingerprint vector, and then finding the k nearest neighbors among the training fingerprints. We used the same training and classification parameters that Hayes and Danezis used in their work [19] with our feature vectors to compare classifier performance for KF.

Table 3. Closed-world accuracy (Acc), TPR, FPR, and within-monitored accuracy (WM-acc) comparing to existing classifiers. (all results in %)

Metric	Acc	TPR	FPR	WMAcc
cumulTLS[30]	18.7	35.0	3.9	8.9
k-FP($k=1$)[19]	40.3	65.4	0.03	35.8
k-NN[43] ($k=1$)	44.5	88.2	22.9	41.1
k-NN ($k=2$)	42.3	32.9	4.70	24.5
k-NN ($k=3$)	43.7	18.7	1.7	16.1
svmResp	64.0	82.6	8.1	56.5

5.2 Binary Classification

Binary classification is used to classify instances into one of two possible outcomes. However, since we have a multi-labeled dataset, we investigate two approaches. The first is to train classifiers based on binary-label learning by converting all data to be labeled with 1 or -1. The second is to train classifiers with class labels, and convert all monitored labels output by the classifier to 1, and the unmonitored label to -1. In this setting, when we compute the True Positive Rate (TPR) and False Positive Rate (FPR), we ignore the confusion between monitored query traces. In other words, we recognize it as a TP because, although the classifier classifies a monitored query trace into a different keyword in monitored keywords, the attacker is still able to determine that it is a monitored keyword. We used both binary-label learning and multi-label learning in the binary classification stage for closed and open world experiments.

If we assume that the censor's goal is to block or detect monitored keywords, the precision indicates how many innocent users are misclassified as searching for "monitored keywords" and recall explains how many

Table 4. Google trace identification

background	40k	80k	100k
TPR(%)	99.2	98.6	98.6
FPR(%)	0	0	0
precision(%)	100	100	100

Table 5. Bing trace identification

background	40k	80k	100k
TPR(%)	99.7	99.9	99.8
FPR(%)	0	0	0
precision(%)	100	100	100

Table 6. Duck trace identification

background	40k	80k	100k
TPR(%)	99.6	99.6	99.6
FPR(%)	0	0	0
precision(%)	100	100	100

guilty users evade detection by being misclassified as typing "background keywords". If the censor is interested in a particular decision among multiple choices to determine if it is a particular keyword, they should focus on reducing false positives (FPs), which would result in increased precision. If the censor is more interested in determining whether any monitored keyword is queried, they will focus on false negatives (FNs), which are measured by the recall.

5.3 Multiclass Classification

Multiclass classification is used to classify instances into one of multiple possible outcomes. To support the multiclass classification using SVMs, we can reduce the problem to multiple binary classification problems with two different strategies: One-against-one (OAO) and One-against-all (OAA). The OAO approach is more popularly used with SVMs since it is faster than the OAA. OAO classification trains $\frac{k(k-1)}{2}$ classifiers (if we have k labels) per all possible pairs of labels while OAA trains k classifiers since its purpose is to classify a single label against all remaining labels. It is well known that OAA is more accurate than OAO in most cases when we use SVMs [8]. For our experiment, we decided to use OAO, yielding more reasonable computational cost for multi-class classification as other researchers have done [7, 22, 30, 44].

To solve the issue of how to count the confusion between monitored query traces as either FPs or FNs, we additionally suggest a new metric, "within-monitored accuracy", which computes the accuracy within monitored query traces. This better represents the probability of the correct classification of individual keywords between monitored query traces. Therefore, in the open-world experiment, we measure success using "within-monitored accuracy."

6 Experiments

In this section, we evaluate the feasibility of the KF attack through a series of experiments. First, we show that query traces of a targeted search engine can be identified with nearly perfect accuracy against other webpage traces. With the traffic identified as traces of the target search engine, we evaluate KF in both closed-world and open-world settings across several different experimental conditions.

In particular, we investigate the extent to which our new feature sets described in Section 4 outperform existing WF feature sets [7, 14, 20, 30, 31, 44] to identify keywords. Furthermore, we focus on investigating whether both identifying monitored keyword traces and differentiating keyword traces from a single search engine are feasible with our new task-specific feature sets, even with 80,000 background keywords. In addition, to achieve this new task, fingerprinting keywords, we consider new variables affecting the performance of classifiers such as different Tor Browser settings and search engines. We also evaluate different classifiers on search query traces to suggest the best method for keyword fingerprinting and evaluate KF under two WF mitigation mechanisms, BuFLO [5] and Tamaraw [6]. Lastly, we further explore the open questions of what factors affect the fingerprintability of keywords.

Throughout this section, we use a variety of metrics to evaluate classifiers:

- **FPR**: The fraction of traces from background keywords classified as monitored keywords.
- **TPR** and **Recall**: The fraction of traces from monitored keywords classified as monitored keywords, a more useful metric in the case that most traces are from the set of monitored keywords.
- **Precision**: The fraction of positive classifications that are correct, a more useful metric in the case that most traces are not from the monitored set.
- **Accuracy**: The overall fraction of traces that are correctly classified.
- **Within-monitored (WM) Accuracy**: The number of monitored keyword traces classified with the correct label over the total number of monitored traces. This metric is only used for multi-class classification, as discussed in Section 5.3.

With these metrics, we try to answer the following research questions regarding how various factors impact the performance of KF attacks:

- **Q1**: Is it achievable to sift target search engine traces from all webpage traces? (Section 6.1)
- **Q2**: How much is the performance of KF improved with Resp feature sets? (Section 6.2)

Table 7. Closed-world accuracy of various feature sets. (Note that Aggr2 is Total+RcumulRespTLS, Aggr3 is Aggr2+RespTotal, and Aggr4 is Aggr3+RcumulRespTorCell)

feature	Accuracy(%)
Total	35.5
torCell	7.5
roundedTCP	12.7
roundedTLS	15.2
burstIncoming	26.7
cumulTLS	18.7
RespTotal	26.1
RespTLS	17.2
RcumulRespTorCell	53.4
RcumulRespTLS	53.8
Aggr2	62.2
Aggr3	63.4
Aggr4	64.0

- **Q3**: What kind of label learning should attackers choose according to their goal? (Section 6.3.2)
- **Q4**: What Tor browser settings are recommended to protect users against KF attacks? (Section 6.3.5)
- **Q5**: Which search engine(s) provide the strongest anonymity against KF attacks? (Section 6.3.6)
- **Q6**: How much do existing WF defenses degrade the success of KF? (Section 6.3.7)

6.1 Search Query Trace Identification

We trained SVM classifiers using Panchenko *et al.*'s cumulTLS features [30], drawing positive examples from our search engine traces and negative examples from 111,884 webpage traces provided by Panchenko *et al.* [30] without Google, Bing, or Duckduckgo search queries.

The monitored set consisted of 100 instances of each of 100 different keywords, while the size of the background set was varied from 40,000 to 100,000. As shown in Tables 4, 5, and 6 even with 100,000 background queries, we were able to detect query traces with a minimum of 98.6% TPR and 0% FPR. This result is plausible since we are distinguishing one specific class of page from all other traffic, and query responses follow a more restricted format with fewer embedded objects than other webpages. This distinction is illustrated by the PCA plot shown in Figure 1. In the following sections, we restrict the input traces to query traces from the target search engine.

6.2 Closed World Accuracy

In the closed world scenario, we assume that the victim may query 100 keywords and seek to classify which of those 100 keywords a given Google query trace represents. As other researchers have pointed out [22, 33], the closed world scenario relies on unrealistic assumptions. However, accuracy in the closed-world setting is a minimal requirement for plausibility. We performed multiclass classification with each feature set in Table 7, labeling each trace according to its keyword.

We used 10-fold cross-validation — partitioning the traces into 10 folds of 1,000 — to get the best C and γ for each feature set and to ensure that the training sets and testing sets did not overlap. Finally, we obtained 1,000 predictions for each fold (for a total of 10,000 predictions). As shown in Table 7, feature sets using information from the Response portion of a query trace outperformed all previously used WF feature sets [7, 14, 20, 30, 31, 44] for KF. In particular, our best feature set (Aggr4) showed much better performance than the feature set (cumulTLS) used for WF by Panchenko *et al.* [30], achieving a closed-world accuracy of 64.0% compared with only 18.7%. These results clearly show that feature sets tailored to keyword fingerprinting can improve the quality of KF attacks.

Table 8. TPR, FPR, and Precision when we use 100 instances of 100 monitored Google query traces and 1 instance of 10,000 background traces.

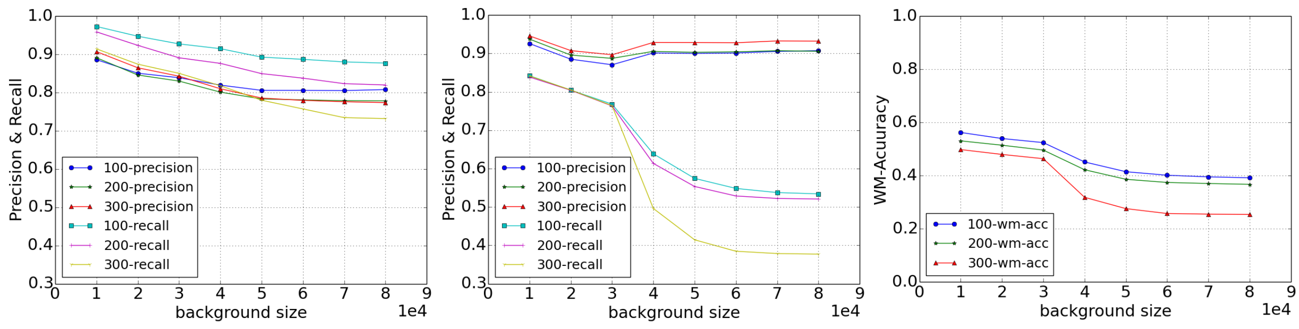
Metric	Binary-label	Multi-label
TPR(%)	93.1	82.6
FPR(%)	14.9	8.1
Precision(%)	86.3	91.1

6.3 Open World Scenario

In the open-world scenario, the attacker maintains a set of monitored keywords to identify, while the victim may query arbitrary keywords. This is a more realistic scenario in the real world, because if the attacker tries to capture victims' search query traces to infer user-typed keywords, the collected data will be expected to include more background keywords than monitored keywords. This section evaluates the performance of several variations of KF in an open-world setting.

6.3.1 Classifier Comparison

First, we compared k-NN [43], CUMUL [30], and k-FP [19] with svmResp to determine the best classifier for KF. In this experiment, we used 100 traces for each of 100 keywords from the top-ranked keyword set, and used 1 trace for each of 10,000 keywords from the background set. We then trained and evaluated classifiers using 10-fold cross validation in the *multi-label learning* setting, where each monitored trace was labeled with its



(a) Binary-Label Learning **(b)** Multi-Label Learning **(c)** Within-monitored accuracy

Fig. 4. Precision and recall for binary classification and within-monitored accuracy for multiclass classification when varying the number of monitored and background keywords

corresponding keyword, and all background traces were assigned a single "background" label. Note that we used the Aggr4 feature set for k-NN and k-FP.

The results are summarized in Table 3. As expected, CumulTLS based on the first 104 features had worse performance than svmResp because the previously used features do not vary enough between query traces from a single search engine. The k-FP classifier was able to differentiate between monitored and un-monitored keywords quite well, achieving a FPR of just 0.03%, but did not do well in identifying the precise monitored keyword for a given query trace. We hypothesize that bagging based on subsets of features discards too much useful sequencing information in the RcumulRespTLS and RcumulRespTorCell feature vectors.

Although k-NN with $k = 1$ had the highest TPR, its FPR was the highest as well. As expected, with higher $k \in \{2, 3\}$, we see a reduced FPR but TPR significantly decreases as well. Note that the FPR is much higher than observed when applying k-NN to website fingerprinting; combined with the observed low FPR of the k-FP classifier, this suggests that the feature vectors of query traces are too densely packed in ℓ_1 space for k-NN to take advantage of the multi-modality of keyword classes. svmResp produced the best within-monitored accuracy rate, which is important in the keyword identification phase of KF. Thus we continue to use svmResp as the classifier for the remainder of the paper.

6.3.2 Effect of label learning.

We also evaluated KF in the *binary-label learning* setting, where each trace was given a binary label according to whether it belonged to the monitored or background set. As shown in Table 8, multi-label learning achieved higher recall (93% vs. 83%) while binary-label learning resulted in better precision (91% vs. 86%).

Furthermore, as shown in Figure 9 in Appendix E, multi-label learning continues to ensure better precision while binary-label learning results in higher recall across different C and γ pairs. This is because the probability a trace is classified as a FN is lower in binary-label learning since we converted all monitored keyword labels into a single label. Thus classifiers were trained with more instances of the monitored label than those in multi-label learning. For instance, if we select 30 instances for each of 100 monitored keywords, the binary-label classifier is trained with 3,000 instances for a single monitored label. In contrast, the probability a trace is classified as a FP is lower in multi-label learning since the classifier is trained with fewer instances for each monitored label than in binary-label learning.

Based on these results and the discussion in Section 5, if the attacker wants to correctly identify individual monitored keywords, multi-label learning will have better performance, while also ensuring fewer incidences of censoring non-monitored keywords. If the attacker's goal is to restrict access to keywords in the monitored group, binary-label learning is better since it is more important to ensure that fewer targeted queries evade filtering.

6.3.3 Effect of monitored and background set size.

To determine how the sizes of the monitored and background sets change our results, we selected monitored sets of 100, 200 and 300 keywords, collecting 80 traces for each keyword, and varied the size of the background set between 10,000 and 80,000. In a sense, increasing the size of the background set represents an attempt to capture the large variation in non-monitored search engine queries. Figure 4 shows that for binary-label classification, the precision has almost no variation with the size of the monitored set in binary-label learning and minimal variation in multi-label learning, but decreases for

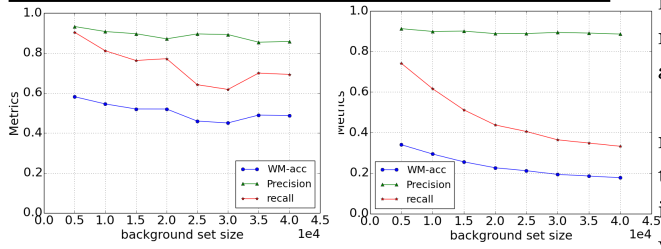
Table 9. Precision(P), recall(R), and within-monitored accuracy(W) (%) to detect 3,000 and 8,000 traces of top-ranked and AOL search keywords and 3,000 traces of Google blacklisted keywords using binary and multiclass classification against 50k–80k background(B) keyword traces. Variance in all figures was less than 0.1%.

B	Top(3,000)			Black(3,000)			AOL(3,000)		
	P	R	W	P	R	W	P	R	W
50k	83.2±.1	31.6±.1	21.6±.1	76.8±.1	24.2±.1	24.2±.1	70.2±.1	12.2±.1	11.6±.1
60k	82.4±.1	28.7±.1	20.0±.1	76.7±.1	23.1±.1	23.1±.1	71.9±.1	9.5±.1	9.1±.1
70k	84.1±.1	27.9±.1	19.2±.1	76.8±.1	22.4±.1	22.3±.1	75.2±.1	8.1±.1	7.6±.1
80k	84.7±.1	28.5±.1	19.8±.1	76.1±.1	22.1±.1	22.0±.1	75.7±.1	6.0±.1	5.7±.1

B	Top(8,000)			AOL(8,000)		
	P	R	W	P	R	W
50k	90.0±.1	57.5±.1	41.4±.1	80.4±.1	31.2±.1	28.8±.1
60k	90.1±.1	54.8±.1	40.1±.1	79.5±.1	27.5±.1	25.6±.1
70k	90.6±.1	53.8±.1	39.5±.1	79.7±.1	24.3±.1	22.9±.1
80k	90.7±.1	53.4±.1	39.1±.1	81.5±.1	20.2±.1	19.2±.1

Table 10. Analysis of HTML and search results screenshot for Top-ranked keywords and AOL search queries. We counted the number of content types among text links, images, videos, SNS, and maps, and computed the fraction of HTML responses (t-html), that consist only of text links and include no other contents such as images.

Dataset	HTML size(KB)	# of contents	t-html(%)
Top-ranked	429±131	3.5±0.7	12.0
AOL	384±98	1.2±0.8	63



(a) Incremental Query Setting (b) One Shot Query Setting

Fig. 5. Within-monitored accuracy, precision, and recall to detect 8,000 top-ranked Google keyword traces when varying Tor browser settings (JS enabled vs. disabled)

both settings with a larger background set. The recall always decreases by increasing the size of either set and is more sensitive to the size of background set in multi-label learning. For multiclass classification, Figure 4c shows that increasing the size of either set results in a decrease in within-monitored accuracy. However, all of the metrics seem to stabilize with background sets of size 50,000 suggesting these experiments accurately capture the variability in search engine result fingerprints. We note that Figures 4b and 4c show a significant change in precision, recall and WM-accuracy between background sets of size 30,000 and 40,000. This was due to the random inclusion of an unusual number of outliers in the 4th partition of background traces, as discussed in Appendix F.

6.3.4 Effect of monitored keyword set.

To determine how the set of monitored keywords impacts our results, we further constructed two additional monitored sets using AOL search queries and Google blacklisted keywords. Using 30 instances for each of 100 top-ranked, AOL, and Google blacklisted keywords, Table 9 illustrates that KF is more effective for top-ranked keywords. However, with more training data using 80 instances, KF is still able to distinguish between 100 monitored AOL keywords with recall of 31% and WM-accuracy of 29%.

Thus, we can make KF work adequately on different monitored datasets with sufficient training data while the specific set of monitored keywords does somewhat impact the performance of classifiers. To further explore why top-ranked keywords are better targets for KF, we examined search result screenshots and HTML responses for top-ranked keywords and AOL dataset. According to Table 10, we found that top-ranked keyword traces contained more diverse types of contents as well as larger embedded objects than AOL search queries and these impact the performance of KF. In Section 6.5, we furthermore intensively investigate the relationship between specific types of contents and fingerprintability.

6.3.5 Effect of different query setting.

For users who disable JavaScript in Tor Browser as discussed in Section 3, we also investigated KF in the "one-shot query setting," which includes neither interaction with the search box nor the incremental results returned by Google Instant. We classified 100 instances of 100 top-ranked keywords with varying background set sizes. As Figure 5 shows, all metrics except precision in one-shot query traces were worse than those in incremental search query traces. In particular, within-monitored ac-

Table 11. Binary classification (TPR). We did not report the standard deviation, which is less than 0.1.

Back	Google	Bing	Duck
10k	81.2±.1	78.4±.1	75.2±.1
20k	77.2	74.9	71.4±.1
30k	71.8	66.2	60.0±.2
40k	67.2±.1	61.2±.2	56.2±.1

Table 12. Multiclass classification (WM-accuracy). We did not report the standard deviation, which is less than 0.1.

Back	Google	Bing	Duck
10k	54.5±.1	44.3±.2	44.4±.1
20k	52.0±.1	42.0±.1	41.9±.1
30k	45.1	37.7±.1	35.8±.1
40k	48.2±.1	34.9	33.7

curacy was significantly lower. (17% vs. 48% for 100 monitored and 40,000 background keywords)

The main reason is that incremental traces carry additional rich information such as traffic for autocomplete and Google Instant search results, which are highly likely to be consistent in the same keyword group. Additional regular traffic makes Aggr4 more distinguishable than features only based on incoming traffic for embedded objects in HTML responses returned by the one-shot query. Thus, the incremental query setting is more vulnerable to KF, indicating that disabling JavaScript also helps to mitigate the KF attack.

6.3.6 Effect of search engine.

We evaluated classifiers for binary and multiclass KF trained using 100 monitored top-ranked keywords and varying background sets across three different search engines, Google (Instant), Bing, and Duckduckgo. As shown in Tables 11 and 12, for binary classification, TPR is higher with Google as the size of background set increases. For multiclass classification, WM accuracy is consistently highest for Google, due to the extra information leaked by incremental search. Overall, this study shows that our approach can be applicable to most search engines since their query traces follow a similar format, containing a large and informative response portion in their TLS record sequences.

6.3.7 Effect of WF defenses.

We evaluated the effect of two WF defenses on KF in the closed-world setting: BuFLO [5], and Tamaraw [6]. BuFLO enforces packet sizes and inter-packet timing to be constant and pads with dummy packets until the total number of packets reaches a threshold; Tamaraw allows different inter-packet timing for incoming and outgoing traffic and pads incoming and outgoing packets to the nearest multiple of the "padding length" parameter.

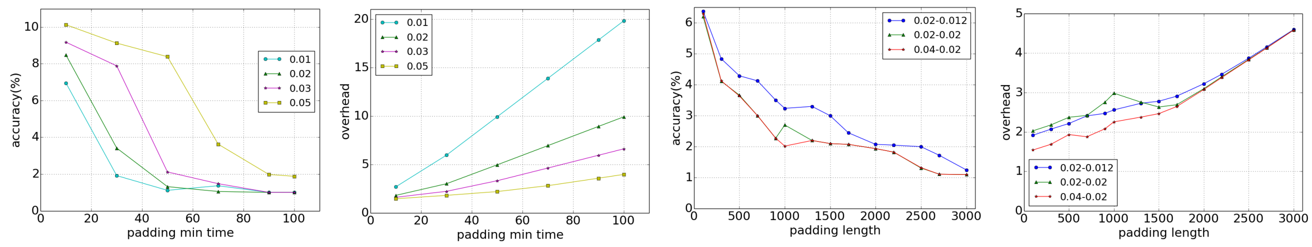
Table 13. Accuracy under Tamaraw when varying incoming (row) and outgoing (column) padding intervals as well as padding lengths (100–500)

interval	0.04	0.02	0.01
0.005	5.86±0.25	5.73±0.25	5.89±0.31
0.012	5.24±0.7	5.17±0.88	5.45±0.62
0.02	4.53±0.09	4.66±1.11	5.85±0.89
0.05	6.09±1.2	6.74±0.93	7.39±0.37

We simulated 80 query traces for each of 100 monitored keywords under each defense. However, since both defenses interleave padded outgoing packets with the response portion, the RcumulRespTLS and RcumulRespTorCell feature sequences extracted from these traces were often shorter than the full 120 records used in previous sections: for Tamaraw with padding length 100, only 2,300 traces had at least 40 records, and for BuFLO with 10 seconds of minimum time for padding, only 7800 traces had the full 120 records. For both defenses, if we use larger bandwidth parameters (e.g, 200–3,000 for Tamaraw and 30–100 seconds for BuFLO), fewer than 10 records remain in the response portion of most traces.

For Tamaraw, first, we need to find the effective incoming and outgoing padding intervals, that give the least accuracy. We realized that as shown in Table 13, KF works poorly with incoming padding intervals less than 0.02 seconds and chose 3 incoming and outgoing interval pairs, which yielded the worst accuracy, for further investigation of KF performance against Tamaraw.

As shown in Figure 6, larger bandwidth parameters (minimum padding time in BuFLO and padding length in Tamaraw), led to lower accuracy as well as higher bandwidth overhead. BuFLO completely defeated KF at the expense of 660% bandwidth overhead (with 0.03 seconds padding interval and 100 seconds for padding time) while the "standard parameters" used by other WF work applied to KF with 100 monitored keyword traces led to accuracy of 10.1% at the expense of 146% overhead. Under Tamaraw, KF became no better than random guessing with 458% bandwidth overhead (0.02 seconds padding interval for both directions and 3,000 for padding length) while performing adequately (6.4% accuracy) with 191% overhead against the parameters used in previous work. This experimental result shows that Tamaraw ensures lower bandwidth overhead compared to BuFLO for perfect defense and existing padding-based defense mechanisms are able to frustrate KF attacks at the cost of bandwidth overhead.



(a) Accuracy(%) under BuFLO **(b)** Bandwidth overhead under BuFLO **(c)** Accuracy(%) under Tamaraw **(d)** Bandwidth overhead under Tamaraw

Fig. 6. Closed-world accuracy for 8,000 WF defense applied Google traces when considering different feature dimensions (Note that with no defense, accuracy is 64.03%)

Apart from padding-based defenses, there are other defenses intended to prevent the search engine from building accurate user search profiles, as described in Section 8.3. We leave the evaluation of KF against those defenses as future work.

6.4 Summary

To answer the research questions discussed before Section 6.1,

- **A1**: CUMUL classifiers nearly perfectly identify Google search engine traces with TPR 98.6% and FPR 0% when adding 100,000 background webpage traces because search query traces consistently have much less traffic, which makes them clearly distinguishable from other webpage traces.
- **A2**: The feature sets based on the response portion are more informative than existing WF features to fingerprint keywords. This is because search results returned by the same keyword share a common traffic pattern in the response portion even though the entire trace does not have enough power to distinguish different keyword traces.
- **A3**: In binary classification, binary-label learning is recommended if KF focuses on reducing the number of users evading censorship while multi-label learning is suggested to guarantee fewer innocent users being misclassified as querying censored keywords.
- **A4**: As the Tor download page recommends, disabling JavaScript is also helpful to mitigate KF. This setting reduces identifying traffic patterns in keyword traces.
- **A5**: Duckduckgo and Bing provide similar levels of protection against KF, while Google's incremental search increases WM accuracy of KF.
- **A6**: Against both BuFLO and Tamaraw, KF was still more effective than random guessing (around 10% vs 1%) at 200% bandwidth overhead, but had significantly reduced performance compared to no defense; with sufficient padding overhead, both BuFLO and Tamaraw reduced the performance of KF to that of random guessing.

6.5 Fingerprintability Analysis

We observed that even in settings where KF had high within-monitored accuracy, some keywords were correctly classified with high probability while others were never correctly classified. To investigate factors that might contribute to fingerprint resistance, we trained a classifier with 30 instances each of 300 monitored keywords and 47000 background keywords from the Google one-shot data set. We selected the 4 "most fingerprintable" keywords (Top-FP, with TPR from 35%–87%) and 4 "non fingerprintable" keywords (Non-FP, TPR of 0%). Table 18 in Appendix G gives summary statistics for these groups, not showing significant differences outside of the Resp features. We plot them in Appendix G as Figure 10, showing that Top-FP keywords are all tightly clustered whereas the Non-FP keywords are more widely spread; however, we found that several Non-FPs not plotted in Figure 10 constructed tighter clusters, discussed later.

Therefore, for a more precise comparison, we chose the 33 most fingerprintable keywords (with TPRs from 17% to 87%) and 52 Non-FP keywords to evaluate how several factors contribute to fingerprintability:

Screenshot equivalence. First, for each of the 85 keyword groups, we checked if screenshots of the instances recorded by the crawler after loading were the same. Second, for groups whose screenshots of instances were the same, we further investigated the contents carried by the HTML responses to identify what specific types of contents vary between the FP and non-FP groups.

We found that 58% of FP keywords and 44% of Non-FP keywords have the same screenshots in all instances. We further analyzed their embedded contents in HTML. Table 14 shows that Non-FP keywords gen-

Table 14. Analysis of Dynamic and Static contents embedded in HTML

DS	Contents	Non-FP(%)	FP(%)
Dynamic	RHS-DIV	59	45
	News	57	42
	Twitter	17	15
	Stock	11	9
	Reviews	9	5
	People also ask[37]	22	5
	See result about[28]	22	0
	People also search for[36]	22	32
Static	Images	10.8	20
	Video	2	17
	Dictionary	4	32
	UI(e.g.,inner searchbox)	17	26

Table 15. Unmonitored keyword set analysis

(a) Euclidean distance in PCA plot. (Note that scale is e+5)

Dist	same	other	back
FP	4	7.7	7.8
NFP	3.5	6.3	5.7

(b) Within-monitored accuracy(%) with different background keywords

Ratio	.2	.3	.4
back-c	21	29	34
back-v	47.7	47.9	48

erally delivered more dynamic contents[2] than FP keywords. For example, 59% of Non-FP traces include a large right-hand side DIV block — containing various dynamic contents such as stock price and user reviews — versus 45% of FP traces. Dynamic contents make traces more inconsistent leading to worse accuracy.

Unmonitored keyword set. We discovered that while the feature vectors of Top-FP groups were generally tightly clustered in PCA space, the feature vectors of Non-FP groups could either be tightly clustered or more widely spread. Thus, we further computed the average PCA Euclidean distance between instances in the same group and to instances in different groups for each Non-FP and FP keyword and compared them to examine the difference according to fingerprintability.

Table 15a shows that average distance between each FP instance and other keyword instances was larger than for Non-FP instances, however, the gap was not high. After investigating confusions, interestingly, most Non-FP keywords were misclassified as "background" labels. Therefore, we re-calculated their average distances to background keyword instances and the distinction became more pronounced. In addition, we trained classifiers with 200 monitored keywords and two different 24,000 background keyword sets, "back-c" and "back-v", and tested each classifoer with the same background set used in Figure 4b. Traces in "back-v" are more widely-spread than in "back-c," with an average PCA Euclidean distance between instances in the background group of 1.02e+6 versus 3.12e+5, and an average distance to instances in the monitored group of 9.40e+5 versus 3.07e+6. Table 15b shows that the WM accuracy was better with "back-v."

Appendix H discusses further analyses of other factors that might impact fingerprintability. In summary, the existence of dynamic contents in search results hinders fingerprintability while the network condition during trace collection does not seem to have any impact. There was not a significant difference in the range of exit nodes used for FP and non-FP query traces. Moreover, fingerprintability is affected by the choice of background set used in training, a factor that is under the control of the adversary.

7 KF Deployment and Mitigation

In the real world, users are likely to visit other webpages and could be involved in other tasks such as listening to music at the same time as using search engines. Furthermore, users do not announce when they are querying a search engine to allow the adversary to begin recording a trace. In addition, search traces and results can be sensitive to when and where users send queries. In this section, we discuss how these issues complicate application of KF to the real world and present strategies that an adversary might deploy to deal with them.

Single query trace identification. First, we need to split a full Tor connection sequence into sessions. T. Wang and Goldberg [45] proposed "split" strategies based on timing and machine learning to perform this separation under more realistic conditions. Similarly, we can use a pre-defined duration t, for which users pause before moving to another webpage or performing some other action such as clicking a link, to determine the splitting point. Such t can be determined experimentally or empirically.

To ensure a better result, we might additionally use the `*Resp*` feature sets to characterize a single query session and use existing feature sets to represent other webpage sessions. Even though the degree of difficulty as well as feasibility is not in the scope of this paper, this machine-learning based approach to identify a query session is viable based on the work of T. Wang and Goldberg [45]. To handle noise, they further proposed adding similar high-bandwidth noise to testing

2 The definition of dynamic contents is that the content is variable enough to be changed within 0.2–54.7 hours based on Table 19 in Appendix H.

data rather than removing the noise to get better accuracy. A similar technique could clearly be applied to KF attacks.

Caching, location and time effect on search results. In our experiment, we did not consider the user customization effect since Tor Browser by default disables caching and cookie storage between browser sessions. Therefore, we assumed that the same content always is returned to different users if they type the same keyword. For the location effect, based on our Tor crawler logs, the exit nodes were selected among 43 different countries, which should capture a significant representation of the diversity by exit location seen by typical Tor users. In addition, Google has not publicly reported how often they update search results. As McDonald's blog [27] mentioned, it is presumed to be 4–5 times a year. Juarez *et al.* [22] reported that WF accuracy goes down to around 0% if the gap between collection of training data and testing data is more than 90 days. Therefore, to ensure good results, training classifiers every 1 to 3 months seems like a fair choice.

Mitigation. As we explored in Section 6.3.7, padding-based defenses deteriorate the performance of KF since insertion of outgoing dummy packets makes identifying the response portion more difficult in addition to concealing traditional features. More sophisticated outgoing padding that focuses on the largest incoming burst to make it less distinguishable among different keywords in terms of the length of the burst, in a method similar to the sequence padding used in Walkie-Talkie [46], and the size of TLS in the response portion seem likely to reduce the bandwidth overhead while further diminishing the performance of KF attacks.

Based on the results in Section 6, Tor alone does not have enough power to protect privacy in search queries against targeted traffic analysis. To avoid bandwidth overhead from padding approaches, cryptography and obfuscation based techniques (as discussed in section 8.3) could be adopted to help conceal the link between users and their search queries, but more work is needed to evaluate the value of these techniques in the context of KF.

Reproducibility. We provide the software and data sets needed to reproduce all of our results on github.[3]

8 Related Work

8.1 Side Channel Attacks and Defenses on Encrypted Network Traffic

The idea of inferring meaningful information based on analyzing encrypted SSL packets was introduced in 1996 [41]. Many studies have exploited side channel leaks in web applications through traffic analysis and investigated their countermeasures. Chen *et al.* [11] showed that financial information, health profiles, and search queries were leaked over HTTPS and WPA by packet inspection. Schaub *et al.* [34] and Sharma *et al.* [35] specifically focused on side channel leaks in Google. Sharma *et al.* discovered the relationship between typed characters and their exchanged encrypted packet lengths, although this attack has not worked since 2012,[4] and Schaub *et al.* presented enhanced side channel attacks with stochastic algorithms. For defenses, Zhang *et al.* [48] developed Sidebuster, based on program analysis to quantify side channel leaks via traffic analysis. Chapman *et al.* [10] also presented a black-box tool for side channel weakness quantification using the Fisher criterion. Backes *et al.* [3] adopted a formal approach, enabling information flow analysis to detect side channel attacks.

In contrast, KF targets Tor where all traffic is encrypted, the size of packets is padded to multiples of 512 bytes, and multiple sessions are sent over the same circuit, which makes the traffic patterns less distinct than other encrypted channels. In particular, KF considers broader feature sets beyond traditional feature sets for better classifier performance.

8.2 Fingerprinting Attacks and Defenses on Tor

Attacks. After Herrmann *et al.* [20] first described WF attacks on Tor using Naive Bayes classifiers, more advanced WF attacks on Tor were suggested by Panchenko *et al.* [31], introducing sophisticated machine learning techniques using diverse feature sets to ensure higher accuracy. Later on, many researchers [7, 44] have introduced more powerful and realistic attacks, improving accuracy up to around 90% in the closed-world setting. Their research aimed to extract new features and improve the quality of classification. In particular, a novel

3 https://github.com/KeywordFingerprinting/KF

4 Since 2012, Google has supported many possible sequences of packet lengths for a given search query to make such prediction more challenging. KF is still effective despite this change, because we did not require a deterministic relationship.

WF attack by Cai *et al.* [7] successfully defeated existing well-known defense mechanisms such as HTTPOS [26]. After that, Juarez *et al.* [22] discussed how previous work [7, 20, 31, 44] overestimated the adversary's capabilities due to unrealistic assumptions such as the closed-world setting. In comparison, most of these criticisms do not apply to our work. Our data was collected using a recent version of Tor Browser and the crawling framework developed by Juarez *et al.* [22]. The multistage nature of the KF attack means that although some query traces might be missed as false negatives due to multitab browsing, with high probability only search query traces will pass on to the keyword classification stage. By necessity, KF attacks do not focus on the index page of a search engine. Finally, we consider both the impact of the training set size and the number of classes in our analysis for evaluating KF attacks.

Following this criticism, L. Wang *et al.* [42] introduced semantics-based, entropy-based, and machine learning-based attacks to detect Tor pluggable transports. More recently, Panchenko *et al.* [30] proposed more scalable attacks to vary the size of background web page sets and further tried to fingerprint web pages (rather than web sites), additionally considering other subpages on the website. Hayes and Danezis [19] proposed Random Forest-based WF attacks on both Tor and standard web browsers, and analyzed the influence of various features using their classifiers.

Defenses. WF Defense mechanisms change the pattern of traffic at the transport level. Wright *et al.* [47] proposed traffic morphing, which changes the traffic pattern to match other traffic. However, since their approach still leaks other information such as packet order, attacks focusing on other features can readily evade their mechanism. Luo *et al.* [26] proposed novel HTTP/TCP level defense mechanisms, which changed window sizes and the order of packets in the TCP stream and injected extra data into HTTP GET headers. The Tor community also introduced "randomized pipelining" [32] to defend against WF, by having the browser load web content in a random order. More recent defenses such as BuFLO [5] and Tamaraw [6] require very high bandwidth overhead and have not been deployed.

8.3 Privacy Protection in Search Engines

Apart from Tor and traffic analysis against defenses, there have been other lines of research on improving privacy guarantee in user search queries based on Private Information Retrieval (PIR) and obfuscating user profiles. Howe *et al.* [21] and Domingo-Ferrer *et al.* [12] used bogus queries to mask actual queries and prevent servers from tracing identifiable user information in query logs. Balsa *et al.* [4] performed an in-depth analysis of privacy properties in web searches and systematically evaluated existing obfuscation-based methods; they found that these methods did not adequately mask a user's actual queries from the search engine. Juarez and Torra [23] proposed a proxy-based approach to dissociate user queries with acceptable overhead in browsing. RePriv [15] proposed in-browser data mining to ensure individual privacy and improving the quality of search by requiring user consent before transferring sensitive information.

9 Conclusion

We have described a novel attack, keyword fingerprinting, to identify search engine queries over Tor, using new feature sets focusing on incoming packets in the response portion of a search query trace. We performed feature analysis to select appropriate new features for this classification task, and analyzed the effect of several variations on the attack, including the choice of classifier, size and contents of the monitored set, the size and contents of the background training set, and the search engine and query method. Across these variations, the results show acceptable performance and suggest that new work is needed to understand how to defend against keyword fingerprinting attacks, given the importance of protecting the contents of search engine queries.

Acknowledgments

This work was supported by the NSF under grant 1314637. We thank Marc Juarez for his help with the Tor Crawler, Tao Wang for sharing the k-NN classifier code, Jamie Hayes for sharing the k-FP classifier code, and our shepherd Ian Goldberg for helpful comments and suggestions regarding WF defense evaluation.

References

[1] Internet live stats. http://www.internetlivestats.com/one-second/#google-band.
[2] Tor homepage. https://www.torproject.org/.
[3] M. Backes, G. Doychev, and B. Köpf. Preventing Side-Channel Leaks in Web Traffic: A Formal Approach. *NDSS*, 2013.
[4] E. Balsa, C. Troncoso, and C. Diaz. OB-PWS: Obfuscation-based private web search. In *Proceedings - IEEE Symposium*

on Security and Privacy, pages 491–505, 2012.

[5] X. Cai, R. Nithyanand, and R. Johnson. Cs-buflo: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 121–130. ACM, 2014.

[6] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 227–238, New York, NY, USA, 2014. ACM.

[7] X. Cai, X. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. *Proceeding of the 2012 ACM conference on Computer and Communications Security*, pages 605–616, 2012.

[8] F. F. Chamasemani and Y. P. Singh. Multi-class Support Vector Machine (SVM) Classifiers – An Application in Hypothyroid Detection and Classification. In *2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications*, pages 351–356. IEEE, Sep 2011.

[9] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[10] P. Chapman and D. Evans. Automated Black-Box Detection of Side-Channel Vulnerabilities in Web Applications. *Proceedings of the 18th ACM conference on Computer and communications security - CCS '11*, (October):263, 2011.

[11] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Proceedings - IEEE Symposium on Security and Privacy*, pages 191–206, 2010.

[12] J. Domingo - Ferrer, A. Solanas, and J. Castella - Roca. $h(k)-$private information retrieval from privacy-uncooperative queryable databases. *Online Information Review*, 33(4):720–744, Aug 2009.

[13] G. Dudek. Aol-user-ct-collection. http://www.cim.mcgill.ca/~dudek/206/Logs/AOL-user-ct-collection//.

[14] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 332–346, Washington, DC, USA, 2012. IEEE Computer Society.

[15] M. Fredrikson and B. Livshits. RePriv: Re-imagining Content Personalization and In-browser Privacy. In *2011 IEEE Symposium on Security and Privacy*, pages 131–146. IEEE, May 2011.

[16] Google-Instance-Disliked-Blacklist-Words. https://www.2600.com/googleblacklist/.

[17] G. Greenwald and E. MacAskill. NSA Prism Program Taps in to User Data of Apple, Google and Others. *The Guardian*, June 2013.

[18] S. Hansell. AOL Removes Search Data On Vast Group Of Web Users, 2006.

[19] J. Hayes and G. Danezis. k-fingerprinting: a Robust Scalable Website Fingerprinting Technique.

[20] D. Herrmann, R. Wendolsky, and H. Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. *CCSW*, 2009.

[21] D. C. Howe and H. Nissenbaum. TrackMeNot: Resisting Surveillance in Web Search. *Lessons from the Identity Trail: Anonymity, Privacy and Identity in a Networked Society*, pages 417–436, 2009.

[22] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt. A Critical Evaluation of Website Fingerprinting Attacks. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, pages 263–274, 2014.

[23] M. Juarez and V. Torra. DisPA: An Intelligent Agent for Private Web Search. pages 389–405. Springer International Publishing, 2015.

[24] Keyword-Tool. http://keywordtool.io.

[25] W. H. Kruskal and W. A. Wallis. Use of Ranks in One-Criterion Variance Analysis. *Source Journal of the American Statistical Association*, 4710087:583–621, 1952.

[26] X. Luo, P. Zhou, E. W. Chan, W. Lee, R. K. Chang, and R. Perdisci. Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows. In *NDSS*, 2011.

[27] B. McDonald. How often does google update its search results? https://hdwebpros.com/blog/how-often-does-google-update-its-search-results.html, 2013.

[28] M. Miller. Google launches knowledge graph, 'first step in next generation search'. https://searchenginewatch.com/sew/news/2175783/google-launches-knowledge-graph-step-generation-search, 2012.

[29] J. Ng. *Blocked on Weibo: What Gets Suppressed on China's Version of Twitter (And Why)*. New Press, The, 2013.

[30] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website Fingerprinting at Internet Scale. *16th NDSS (NDSS 16)*, pages 143–157, 2016.

[31] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. *WPES*, 2011.

[32] M. Perry. Experimental defense for website traffic fingerprinting. https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting, 2011. Accessed: 2015-11-23.

[33] M. Perry. A critique of website traffic fingerprinting attacks. https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks, 2013. Accessed: 2015-11-1.

[34] A. Schaub, E. Schneider, A. Hollender, V. Calasans, L. Jolie, R. Touillon, A. Heuser, S. Guilley, and O. Rioul. Attacking Suggest Boxes in Web Applications Over HTTPS Using Side-Channel Stochastic Algorithms. *Risks and Security of Internet and Systems*, 2015.

[35] S. A. Sharma and B. L. Menezes. Implementing side-channel attacks on suggest boxes in web applications. In *Proceedings of the First International Conference on Security of Internet of Things - SecurIT '12*, pages 57–62, New York, New York, USA, 2012. ACM Press.

[36] J. Slegg. Google adds "people also search for" thumbnails to search results. http://www.thesempost.com/google-adds-people-also-search-for-thumbnails-to-search-results/, 2016.

[37] A. Smarty. Google's "people also ask" (related questions): What are they, and why you should care. http://www.internetmarketingninjas.com/blog/search-engine-optimization/googles-people-also-ask-related-questions/, 2016.

[38] J. Titanium. AOL Search Log Special,Part1. http://www.somethingawful.com/weekend-web/aol-search-log/, 2006.

[39] Tor-Browser-Crawler. https://github.com/webfp/tor-browser-crawler.

[40] tshark. https://www.wireshark.org/docs/man-pages/tshark.html.

[41] D. Wagner, B. Schneier, et al. Analysis of the ssl 3.0 protocol. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, pages 29–40, 1996.

[42] L. Wang, K. P. Dyer, A. Akella, T. Ristenpart, and T. Shrimpton. Seeing through Network-Protocol Obfuscation. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 57–69, 2015.

[43] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, 2014.

[44] T. Wang and I. Goldberg. Improved website fingerprinting on Tor. *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society - WPES '13*, pages 201–212, 2013.

[45] T. Wang and I. Goldberg. On Realistically Attacking Tor with Website Fingerprinting. *Proceedings on Privacy Enhancing Technologies*, (4):21–36, 2016.

[46] T. Wang and I. Goldberg. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks . *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[47] C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, 2009.

[48] K. Zhang, Z. Li, R. Wang, X. F. Wang, and S. Chen. Sidebuster: Automated detection and quantification of side-channel leaks in web application development. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 595–606, 2010.

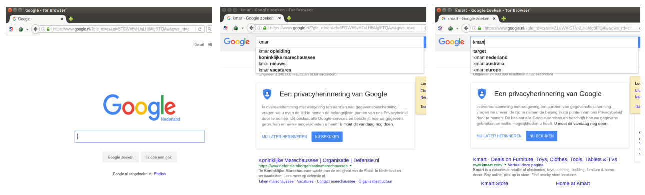
Appendix

A Google Instant

Google returns incremental search results and the suggestion list each time a character is typed, as shown in Figure 7. We implemented these additional functionalities (Section 3.1) in the crawler.

B Traditional Feature Sets

Packet and Cell totals (Total). This is a general feature set widely used in existing work [14, 20, 31, 44]. We computed the total number of packets, total number of incoming packets, total number of outgoing packets, total number of incoming Tor cells, and total number of outgoing Tor cells.



(a) Go to www.google.nl (b) Auto-complete shows its searche suggestions and Google Instant shows the result for kmar (c) Google Instant shows the result for kmart

Fig. 7. Dynamic Google search result page during typing kmart

Table 16. Sum of squares, mean of squares, and χ^2 of feature sets, returned by Chi-square test statistic, and accuracy of feature sets, evaluated using the SVM classifier over 100 parent keywords. Each feature is described in Sections 3, 4, and Appendix B.

Feature	SS	MS	H	Acc
roundedTCP	4.5e+10	4.55e+8	1353	12.73
roundedTLS	6.35e+10	6.42e+8	1905	15.16
cumulTLS	7.08e+10	7.15e+8	2123	18.67
Total	2.15e+11	2.17e+9	6461	35.48
burstIncoming	2.8e+11	2.83e+9	8402	26.7
RcumulRespTLS	2.22e+11	2.24e+9	6667	53.79
RcumulRespTorCell	2.17e+11	2.19e+9	6528	53.43

Tor Cell Trace (torCell). We created a sequence of the number of Tor cells sent in each direction based on the sequence of TLS record sizes. For example, if the sequence of TLS records between the client and guard have sizes 1000, -1500, 700, 500, (where negative numbers indicating incoming packets) the corresponding sequence of Tor Cells is 1, -2, 2 based on the fact that the size of single Tor cell is 512 bytes. We used + to indicate outgoing packets and - to indicate incoming packets.

Rounded TCP (roundedTCP) **and TLS** (roundedTLS). We rounded the packet size by increments of 600, as Cai *et al.* [7] and T. Wang and Goldberg [44] suggested in their work. These are packet sequences for the rounded size of TCP packets and the rounded size of TLS records.

Unique packets (Unique). In the Tor network, certain packet sizes frequently appear. We compiled a list of such common packet sizes; in our experiment, we used the range of [-1050,1050], and marked each packet with 1 if it was on the list and with 0 if it was not. This is similar to the work by T. Wang and Goldberg [44].

Burst of outgoing packets. T. Wang *et al.* [43] introduced bursts of outgoing packets as an identifying

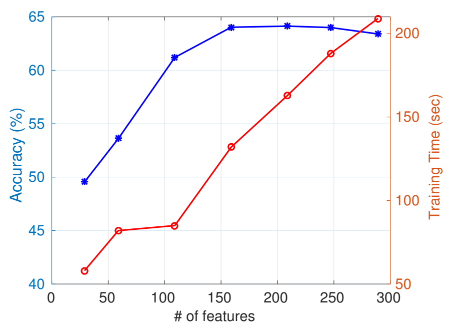


Fig. 8. Accuracy and Training Time when varying the number of features

feature, where a burst is defined as "a sequence of outgoing packets, in which there are no two adjacent incoming packets." They used statistics of bursts as features; e.g., maximum burst length, number of bursts, etc.
Cumulated TLS records (cumulTLS). This feature is used by Panchenko *et al.* [30]. First, we extracted a sequence of TLS records' size. If the sequence T=$(p_1, ..., p_N)$ where p_i is the size of TLS record, we calculated the cumulative sizes, which constitutes C=$(c_1, ..., c_N)$, where $c_1{=}p_1$ and $c_i{=}c_{i-1}{+}p_i$. In this project, we only consider the size of TLS records in cumulTLS feature set while they considered size of TCP and Tor cells in their **CUMUL** feature set.

C Kruskal-Wallis Results and Details

To generate the numbers in Table 16 and Figure 3, first, we applied PCA to feature vectors in each feature set and selected the first two PCA scores to balance different dimensions in each feature set. Then, we replaced numbers with their corresponding ranks. For example, if we have two instances for each of two keywords, a and b, assume that those features vectors are a1=(7.2,10.3), a2=(2.7,4.1), b1=(15.0,3.9), and b2=(6.5,9.1). For ranked measures, we used a1=(5,7), a2=(1,3), b1=(8,2), and b2=(4,6). After that, we computed H for each feature set in Table 16 and for Figure 3, we calculated TR_g^2/N_g for each keyword group.

D The best number of features

We ran SVM classifiers with Aggr4 in closed-world setting by considering different number of features in RcumulRespTLS and RcumulRespTorCell before aggregation. Based on both accuracy and training time, based on Figure 8, we determined 247 as the best feature dimensions and use it throughout Section 6.

E Different label learning

We used different C and γ pairs to show that multi-label learning ensures better precision while binary-label learning returns better recall as given in Figure 9.

F Average PCA distance for background sets

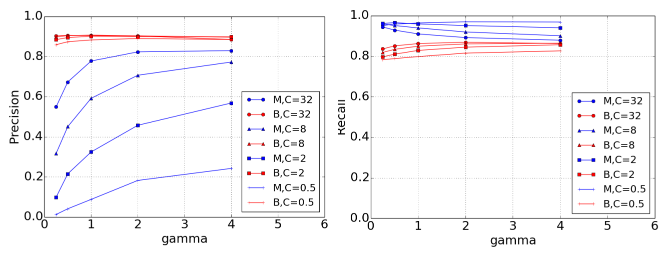
Figure 4b and Figure 4c show that all metrics decreased more significantly with 40,000 background set and since all background sets were constructed based on random partitioning, we additionally computed the average PCA distance between monitored and background traces for each of 10,000–80,000 background sets to show this feature.

G FP vs. Non-FP

We drew the PCA plot in Figure 10 and Table 18 to show difference in feature vectors and packet total statistics according to the fingerprintability.

H Further Fingerprintability Analysis

Tor restart/out of order or dropped packets. We had often received CAPTCHAs during Google trace captures based on Table 19. We examined how often the Tor process had been restarted for each keyword group and further how it affected query results. We found no instances showing a major difference in search results due to the location difference led by a different



(a) Precision (b) Recall

Fig. 9. Precision and Recall when varying C and γ to train 100 traces of 100 parent keywords and 1 trace of 10,000 background keywords (M:Multi-label learning, B:Binary-label learning)

Table 17. We calculated average PCA euclidean distances between montiored and background traces for each of 10k–80k background sets. Note that column indicates the size of monitored sets, row indicates the size of background sets, dist means the average euclidean distance between monitored and background traces, and gap is the difference between distances of two subsequent background sets. For instance, for 20k, the gap is the difference between dist of 20k and dist of 10k.

Back	100		200		300	
	dist	gap	dist	gap	dist	gap
10k	1.40e+6	N/A	2.79e+5	N/A	2.89e+5	N/A
20k	1.41e+6	1.0e+4	2.97e+5	3.02e+4	3.02e+5	1.3e+4
30k	1.42e+6	1.7e+4	3.14e+5	1.69e+4	3.17e+5	1.54e+4
40k	1.52e+6	9.32e+4	1.03e+6	7.18e+5	3.35e+5	1.83e+4
50k	1.55e+6	3.52e+4	1.43e+6	4.01e+5	3.52e+5	1.7e+4
60k	1.59e+6	4.33e+4	1.49e+6	5.71e+4	3.54e+5	1.65e+3
70k	1.60e+6	8.85e+3	1.50e+6	1.33e+4	3.54e+5	2.69e+2
80k	1.63e+6	5.0e+3	1.55e+6	4.93e+4	3.59e+5	4.6e+3

Table 18. Statistics of traces from best fingerprintable keywords. Note that Avg means traces from all 300 parent keywords (30 instances for each) regardless of fingerprintability.

Metric	Top-FP	Non-FP	Avg
# of outgoing packets	84	71	77
# of tor cells	964	883	873
# of packets in Resp	247	193	180
Cumul payload(KB)	462	420	414

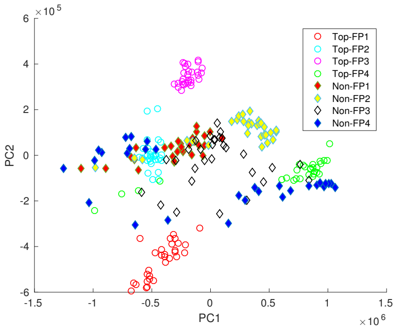


Fig. 10. PCA plot of best fingerprintable keyword traces and non-fingerprintable keyword traces

exit chosen. (For example, we get different search results for a keyword 'man' when we use Google Italia and Google US.) In addition, 54% of FP keywords received a CAPTCHA during collection and the Tor process had been restarted 60 times on average (35% and 23 times for Non-FP keywords). Furthermore, unstable network conditions can lead to dropped, duplicated, or out of order packets. We got the result that 38% of FP traces had such events, whereas 17% of Non-FP contained those.

These results indicate that both CAPTCHA and unstable network condition did not affect the fingerprintability in our experiment.

Table 19. Statistics of capture time and CAPTCHA appearance per keyword for 400 parent keywords collection and 110 instances for each. (Note that restarting the Tor process when encountering CAPTCHAs contributed to the some very large capture times.)

Metric	Capture Time(h)	CAPTCHA(times)
Avg	3.5	37
Min	0.2	0
Max	54.7	220

Page loading time. We expected that the diversity and number of contents in the resulting HTML affected the page loading time. However, the page load time did not show significant difference between FP keywords and Non-FP keywords. (0.08 sec vs. 0.06 sec on average) In fact, page loading time is highly affected by the network condition at that time.