High Precision Open-World Website Fingerprinting

Tao Wang
Hong Kong University of Science and Technology
taow@cse.ust.hk

***Abstract*—Traffic analysis attacks to identify which web page a client is browsing, using only her packet metadata — known as website fingerprinting (WF) — has been proven effective in closed-world experiments against privacy technologies like Tor. We want to investigate their usefulness in the real open world. Several WF attacks claim to have high recall and low false positive rate, but they have only been shown to succeed against high base rate pages. We explicitly incorporate the base rate into precision and call it r-precision. Using this metric, we show that the best previous attacks have poor precision when the base rate is realistically low; we study such a scenario (r = 1000), where the maximum r-precision achieved was only 0.14.**

To improve r-precision, we propose three novel classes of *precision optimizers* that can be applied to any classifier to increase precision. For r = 1000, our best optimized classifier can achieve a precision of at least 0.86, representing a precision increase by more than 6 times. For the first time, we show a WF classifier that can scale to any open world set size. We also investigate the use of precise classifiers to tackle realistic objectives in website fingerprinting, including different types of websites, identification of sensitive clients, and defeating website fingerprinting defenses.

***Index Terms*—website fingerprinting; traffic analysis; Tor; privacy**

I. Introduction

The last few years have seen a sharp increase in TLS usage rate (26% in 2014 to 70% in 2018 [14]). Combined with packet destination obfuscation techniques such as proxies and encrypted SNI, the network-level privacy leaks of the Internet are being plugged. Privacy-sensitive individuals may turn to anonymity networks, which encrypt and redirect the user's traffic across proxies, hiding sender, recipient, and packet contents from a passive observer (a network eavesdropper). However, none of these technologies hide significant features of the user's traffic, such as packet frequency, timing, order, and direction. **Website Fingerprinting** (WF) attacks allow passive network eavesdroppers to use these features to identify the client's destination web page, compromising her privacy.

The WF attacker wishes to monitor a set of sensitive web pages and identify when a client visits these pages. In the closed-world scenario, the client is configured to only visit these sensitive web pages, and the attacker needs to identify which one. In the harder open-world scenario, the client can visit any page, and the attacker must also correctly identify which page visits are non-sensitive.

WF attacks have been proven effective against real-world privacy technologies in the closed-world scenario [3], [8], [13], [19], [29]. However, there is a general academic consensus that known WF attacks fail in the open world because they are too imprecise. In 2013, a Tor developer criticized WF techniques for failing in the open world [21]. In 2014, Juarez et al. found that "prior work succumbs to the base rate fallacy in the open-world scenario" [11]. In 2016, Panchenko et al. studied WF attacks in a large open world and concluded that "no existing method scales when applied [in the open world]" [19].

Multiple works have achieved open-world success since then [7], [20], [29], but they can only identify *high base rate* pages (such as the most popular search engines), as our work will confirm. These may not be the sensitive pages the attackers are interested in. Failure in the large open world implies that WF does not pose a threat to privacy-sensitive individuals visiting *low base rate* pages, such as whistleblowing, file sharing, and politically and culturally sensitive pages. With more than one billion pages on the Internet, the vast majority of pages have a base rate much lower than that which can be threatened by known attacks.

To tackle the unsolved problem of **open-world website fingerprinting** (OWF), this work achieves the first scalable OWF attack for the realistic low base rate scenario. Low base rate open-world scenarios have been challenging for other fields that rely on machine learning as well, including forensic analysis [24], intrusion detection systems [22], and medical imaging [27]. For example, open-world failure in forensic analysis has been subject to public controversy [16] as it leads to wrongful convictions.

In Section II, we show that previous work did not correctly include the base rate when calculating precision. We formulate r*-precision* to explicitly include the base rate, allowing us to evaluate classifiers in OWF over a low base rate and identify several new insights for OWF.

Using r-precision, we show that previous classifiers are not precise when the base rate is low. This motivates our contribution of three novel classes of techniques that can improve the open-world precision of any classifier; we call them Precision Optimizers (POs). We demonstrate the effectiveness of our POs by combining them with six of the best previous WF attacks to create effective open-world WF attacks against a Tor user. We focus on attacking Tor due to its popularity and because it is currently the hardest web anonymity technology to attack using WF [28]. We present these results in Section III; our best PO can improve a classifier's precision from 0.024 to 0.86 in a low base rate scenario, giving us the first attack to achieve high precision in a low base rate scenario.

We show that our optimized classifiers are better able to handle low base rate — and thus pose a greater threat to privacy — in Section IV. In particular, we use them to attack several WF defenses in Section IV-B precisely. In Section V,

DOI 10.1109/SP40000.2020.00015

TABLE I: How we count the number of true positives (N_{TP}), wrong positives (N_{WP}), and false positives (N_{FP}). After counting them, we obtain the true positive rate $R_{TP} = N_{TP}/N_P$, wrong positive rate $R_{WP} = N_{WP}/N_P$, and false positive rate $R_{FP} = N_{FP}/N_N$.

		Classified as		
		Correct sensitive class	Wrong sensitive class	Non-sensitive class
True class	Sensitive class (# = N_P)	True Positive (# = N_{TP})	Wrong Positive (# = N_{WP})	False Negative (# = $N_P - N_{TP} - N_{WP}$)
	Non-sensitive class (# = N_N)	Not possible	False Positive (# = N_{FP})	True Negative (# = $N_N - N_{FP}$)

we show that our optimized classifiers can scale to any open-world size across a range of base rates, and they perform well on alternative scenarios including identification of sensitive clients and actively browsing users. We give a survey of related work in Section VI, and conclude in Section VII.

II. Background

A. Terminology and Threat Model

Machine learning terminology. A classifier takes as input a testing element and determines which class it belongs to. In our case, the testing element is a sequence of packets (with timing, size and direction) and each class is a web page.[1] When the classifier claims that the tested packet sequence is sensitive, it is known as a *positive*; it is a *true positive* (TP) if the tested packet sequence came from the same page the classifier identified, and it is a *false positive* (FP) if the tested packet sequence came from a non-sensitive page. We define a *wrong positive* (WP) to be a sensitive page mistaken as another sensitive page. While some previous works have considered a wrong positive to be a false positive [7], [19], [29], we do not, because the tested packet sequence did not come from a non-sensitive page. Later, we will show that equating wrong positives with false positives would lead to a **significant error** when calculating precision. Refer to Table I for an illustration of these terms and how their rates (TPR/WPR/FPR) are defined. We also refer to TPR as recall; some previous WF works used recall as the main metric to compare and optimize WF attacks [28], [29].

WF threat model and the open world. We use the same threat model as all previous WF works involving the open world [7], [11], [19], [29]. Our WF attacker, Oscar, is a *passive* eavesdropper that is *local* to the client, Alice. The attacker watches packet sequences sent by the client, and knows the client's identity. The privacy-sensitive client uses encryption with proxies to hide her packet contents and destination web page from the attacker, for example, by using Tor. Therefore, the attacker cannot figure out what the client is doing by simply reading her packet headers; the destination web page is hidden.

In the open world, the attacker seeks to compromise her privacy by using a classifier that decides if the client is visiting a set of *sensitive pages*. These sensitive pages could be interesting to the attacker for a variety of privacy-compromising reasons, such as profiling, flagging potential threats, or censorship. The classifier is trained on supervised packet sequence data, which Oscar collected by visiting the sensitive pages himself. The set of sensitive pages is, inevitably, a small subset of all web pages, so the attacker must be able to recognize when the client is not visiting sensitive web pages, avoiding the *base rate fallacy*.

Precision and the base rate fallacy. The base rate fallacy describes the following problem in the open world: a classifier may have high recall and low FPR, but it may still be useless in practice. This happens when the base rate of positive events (sensitive web page accesses) is much lower than the FPR, as the attacker would be overwhelmed by incorrect positive classifications. A OWF eavesdropper overwhelmed by false positives would not be able to determine if a given client is actually visiting sensitive pages or not; the OWF attack has failed to achieve its main objective. This is why it is necessary to achieve high precision when the base rate is low in OWF.

The base rate fallacy is the chief challenge of OWF. To avoid the base rate fallacy, the attacker wants to classify a page access as sensitive only if he is *certain* that the classification is correct. We propose and evaluate three classes of precision optimizers (POs), each of which uses a different kind of certainty to reject questionable sensitive classifications and thus improve precision.

In this work, we also consider WF *defenses*. These defenses have been proposed to defend web-browsing clients against WF attacks. WF defenses are applied by the client and her proxies, and they transform the packet sequence to disrupt the attacker's ability to classify them correctly. We will evaluate the effectiveness of these defenses in the open world with our optimized classifiers.

We present our notation in Table II.

B. r-precision

The base rate fallacy shows that an open-world classifier should only be considered effective if its positive classifications are largely correct; otherwise, the attacker cannot act on its positive classifications. By this standard, TPR, WPR, and FPR alone cannot tell us if the classifier is effective. We also need to include a fourth metric: the base rate at which the client accesses sensitive web pages. Without considering the base rate, it is not possible to determine how any attack fares in realistic low base rate scenarios. We illustrate how the base rate can be incorporated into the calculation of precision as follows.

[1] In this work, we use sequences of Tor cells where each cell has the same size; we still call them packet sequences for generality.

When an attacker monitors a client's web page accesses, his positive classifications (accesses believed to be sensitive pages) may be true (N'_{TP}), wrong (N'_{WP}), or false (N'_{FP}). We mark these values with primes to distinguish them from experimental values, which are not primed. The primed variables represent real observed values due to the behavior of a client,

Many fields in this work did not calculate precision while claiming open-world success, so they succumb to the base rate fallacy [7], [20], [29]. Works that did calculate precision used the following formula [11], [18], [19]:

$$\pi = \frac{N_{TP}}{N_{TP} + N_{WP} + N_{FP}}$$

However, the correct formula for precision should be:

$$\pi = \frac{N'_{TP}}{N'_{TP} + N'_{WP} + N'_{FP}}$$

It is significant to note the difference between the two. The non-primed variables, counted during experiment, are **not** unbiased estimators for the primed variables. For example, N_{TP} grows in proportion to how many sensitive pages we include in the experimental testing data set, while N'_{TP} grows in proportion to how many sensitive pages the client would actually visit. These variables are not related. Using the former formula would be a mistake: by doing so, the experimenter implicitly assumes that the real client visits sensitive pages at the same rate as the experimenter. Previous work often set the non-sensitive and sensitive set sizes to be similar [7], [11], [19], [20], [29]. This means that the experimenter implicitly assumes that the client visits sensitive pages around half of the time ($r \approx 1$), which is unrealistic.

To derive the correct formula, we need to use R_{TP} as an unbiased estimator for R'_{TP}, and so on for R_{FP} and R_{WP}. Therefore, we need to convert the above values to rates:

$$\begin{aligned}\pi &= \frac{R'_{TP} \cdot N'_P}{R'_{TP} \cdot N'_P + R'_{WP} \cdot N'_P + R'_{FP} \cdot N'_N} \\ &= \frac{R'_{TP}}{R'_{TP} + R'_{WP} + \frac{N'_N}{N'_P} \cdot R'_{FP}} \\ &\approx \frac{R_{TP}}{R_{TP} + R_{WP} + r \cdot R_{FP}} = \pi_r\end{aligned}$$

Setting $r = N'_N/N'_P$, we arrive at our formulation of *r-precision* (π_r) that incorporates the base rate. r (which we call the base ratio) is the relative likelihood of negative events (client visiting any non-sensitive page) to positive events (client visiting any sensitive page). A higher r reduces r-precision holding all other rates constant, making the classification problem harder. **Mathematically, r-precision is equivalent to precision**, but it explicitly displays the base ratio r to avoid the above implicit base rate error when calculating precision.

Note that N'_P and N'_N are *not* the sizes of the positive and negative data sets in our experimental setup. $r = N'_N/N'_P$ represents the real ratio of non-sensitive to sensitive pages a client would visit. In a realistic setting, clients may have different values of r representing how often each client visits sensitive pages. Alternatively, r can also represent the same ratio for a given set of clients, or the set of all clients. (We will present some values of r extracted from real users in Section V-B.) r is not a fixed value, nor is it estimated or determined by the attacker. A correct analysis of the precision of a WF classifier should include experiments on various explicit values of r.

We can also see that R_{WP} and R_{FP} contribute differently to π_r, as only the latter is magnified by r. This is why we do not consider wrong positives to be false positives. Doing so would cause us to underestimate precision.

TABLE II: Notation used in this paper.

N_P, N_N	# of positives, negatives
N_{TP}, N_{WP}, N_{FP}	# of true, wrong, false positives
R_{TP}, R_{WP}, R_{FP}	True, wrong, false positive rates
Above variables, primed	Real values (not experimental parameters)
r	Base ratio, equal to N_N/N_P
π_r	precision for base ratio r
PO	Method to improve WF precision
Recall	Equal to R_{TP}
P	A packet sequence
C	A class

C. Does precision trump recall?

In the classical formulation of WF, the effectiveness of an attack is measured with its recall (TPR). We present three arguments to convince the reader that it is more important to optimize the precision of WF attacks than their recall.

Base rate fallacy. The base rate fallacy tells us that a WF classifier can be ineffective no matter how high its recall is. "The boy who cried wolf" tells us that a high-recall, low-precision classifier is useless in a low base rate scenario. This is because the attacker cannot use the classifier's information in any way unless the attacker is confident that the classifier's positive classifications are true: that is to say, precision is high. The base rate fallacy highlights the epistemological deficiency of recall in the open world; it does not tell us anything about open-world effectiveness.

Chilling effect. Web-browsing clients using anonymity networks are sensitive to privacy and do not want the attacker to capture any of their browsing behavior. When WF precision is high, even a moderate recall may trouble privacy-sensitive users. For example, a whistleblower who would suffer a 10% recall — a 10% chance of revealing each sensitive page access to any eavesdroppers — may instead choose to self-censor rather than incur significant personal risk.

Repeated visits. Browsing patterns are often consistent and self-repeating, and users visit the same websites frequently. A low-recall, high-precision attacker would eventually be able to determine if the client is interested in particular sensitive pages. A high recall is unnecessary in this scenario. Repeated visits cause a gradual decay in privacy.

There are situations where both recall and precision are important. If the attacker wants to determine the rate at which

clients visit a page on an anonymity network, low recall would distort the count as much as low precision. There are also scenarios where, due to side information or other preconditions, the possible set of pages the client visited is small, better represented by a closed-world scenario. This is the case for hidden services, which have recently been found to consist a relatively small ecosystem [10]. Nevertheless, in the general open-world case, the above arguments explain why a low-recall, high-precision attacker is more threatening than a high-recall, low-precision attacker. For this reason, we focus entirely on optimizing precision rather than recall or a combined metric such as F-measure or G-measure.

D. Experimental setup

Data set. We collected our data set between February and April 2019 with Tor Browser 8.5a7 on Tor 0.4.0.1-alpha, using one machine on a university network. We focus exclusively on Tor because it is both popular and resilient. Furthermore, it is the most difficult for WF out of currently usable anonymity networks [8], and WF attacks designed to succeed in the Tor scenario also tend to succeed against other privacy technologies [28]. The data set includes both HTTP/1.1 and HTTP/2 web pages.

We collected a set of sensitive (monitored) pages and non-sensitive (non-monitored) pages. We chose the top 100 pages on Alexa as the sensitive set, visiting each of them 200 times, and the next 80,000 pages on Alexa as the non-sensitive set, visiting each of them once. We decided that the sensitive pages should be top sites to ensure reproducibility. Some previous work has instead chosen politically sensitive pages [29], but this made the scientific results unreproducible, because most of those web pages have become unavailable in only a few years. We gave each page up to 90 seconds to load, collecting all cells. Some other pages (not counted in the above) had failed to load; we filtered them away.

For each web page, we collected the times, sizes, and directions of all packets, from which Tor cells can be derived [30], representing a local, passive attacker's information. Some attacks called for TCP packets, for which we directly used the above raw traces; others called for Tor cells, for which we processed the TCP packets to extract Tor cells. We did not intentionally include any noise.

For convenience, we may describe a subset of our full data set using the numeric notation 50x100+20, which denotes that the subset has 50 monitored pages, 100 instances of each page, and 20 non-monitored pages (non-monitored pages always have one instance each). It is necessary to conduct certain experiments on a subset of the data due to computation and memory limitations.

Our experimental setup is the same as all works studying open-world WF. We are allowed to perform data collection on one computer specifically because Tor Browser preserves anonymity by refusing to allow clients to customize it, and Tor's random circuit construction ensures that our attacker will not train on the same circuits as the client. To our knowledge, there is no work showing that this setup skews results unfairly.

Precision and recall. Our objective is to maximize r-precision (π_r) for WF attacks. In our experiments, we present results for $r = 20$ and $r = 1000$, representing respectively an easy and hard classification setting. Note that r describes the total base ratio of all sensitive pages (we consider 100 sensitive pages in our work). For example, if the attacker wants to monitor 100 sensitive pages and the client has a 1/100,000 chance of visiting each sensitive page, the attacker's precision would be correctly captured by our π_{1000} scenario.

The objective of this work is to optimize precision, which may sometimes entail sacrificing recall. This is informed by our argument that a low-recall, high-precision attacker is more threatening to privacy than a high-recall, low-precision attacker. So that our POs will not produce completely inaccurate classifiers, we set a minimum recall in this work of 0.2. We chose 0.2 so that the classifier would still be threatening to people who visit sensitive pages only once; they could not be sure that they would escape detection. Our approach to OWF is novel compared to previous work; we maximize r-precision while ensuring the recall is acceptable, while previous works maximize recall [3], [19], [20], [28]–[30].

Presentation of results. In this work, we measure the 95% confidence interval of a statistic $\hat{x}$ by taking:

$$C(\hat{x}) = 1.96\sqrt{\frac{\hat{x}(1-\hat{x})}{n}}$$

This is the confidence interval of the mean for the normal distribution using the Wald method, and we apply it to TPR, WPR, and FPR. We then write $\hat{x} \pm C(\hat{x})$ to show the confidence interval of $\hat{x}$. We are able to use the Wald method as our n is large (usually 80,000). However, the above does not apply to r-precision. Recall that the definition of r-precision is:

$$\pi_r = \frac{R_{TP}}{R_{TP} + R_{WP} + r \cdot R_{FP}}$$

Let us denote $R_{TP}^{max} = R_{TP} + C(R_{TP})$, and $R_{TP}^{min} = R_{TP} - C(R_{TP})$, and correspondingly for R_{WP} and R_{FP}. We take a naïve 95% confidence interval by computing the maximum π_r:

$$\pi_r^{max} = \frac{R_{TP}^{max}}{R_{TP}^{min} + R_{WP}^{min} + r \cdot R_{FP}^{min}}$$

We take $C(\pi_r) = \pi_r^{max} - \pi_r$ and show the confidence interval as $\pi_r \pm C(\pi_r)$.[2] When r is large (as in our experiments), precision is dominated by the $r \cdot R_{FP}^{min}$ term in the denominator.

When π_r is *high*, $C(\pi_r)$, its confidence interval, is unstable. This is because a high r-precision indicates a very small number of false positives, especially in the $r = 1000$ scenario. For example, consider an experiment with $N_P = 20000$ and $N_N = 80000$ where we find that $R_{TP} = 0.45, R_{WP} = 0, R_{FP} = 0.00005$. This gives $\pi_{1000} = 0.9$. But the number of false positive events is very small, with $N_{FP} = R_{FP} \cdot N_N = 4 < 10$. This violates a general rule of thumb: there must be more than ten occurrences of an event to use the Wald

[2]This $C(\pi_r)$ is larger than an alternative $C'(\pi_r) = \pi_r - \pi_r^{min}$, so the estimation is cautious.

method as the confidence interval. In such cases, we calculate the maximum FPR using the Wilson method (which better suits the extremely small rate R_{FP}), with $z = 1.96$ for the 95% confidence interval:

$$R_{FP}^{max} = \frac{R_{FP} + \frac{z^2}{2N_N} + z\sqrt{\frac{R_{FP}(1-R_{FP})}{N_N} + \frac{z^2}{4N_N^2}}}{1 + \frac{z^2}{N_N}}$$

Then, we take the minimum precision as

$$\pi_r^{min} = \frac{R_{TP}^{min}}{R_{TP}^{max} + R_{WP}^{max} + r \cdot R_{FP}^{max}}$$

Finally, we express the precision with its lower bound of $\pi_r \geq \pi_r^{min}$. We present no value for the upper bound of precision as it cannot be accurately measured. Continuing the above example, we find $R_{TP}^{min} = 0.353$, $R_{TP}^{max} = 0.367$, $R_{WP}^{max} = 0$, and $R_{FP}^{max} = 0.00010$, so we would write $\pi_{1000} \geq 0.76$, not $\pi_{1000} = 0.9$. We use the Wilson method for π_{1000} whenever $N_{FP} < 10$.

III. Precision Optimizers

We modify closed-world classifiers to achieve high r-precision for OWF using Precision Optimizers (POs). Our POs teach the underlying classifier to be conservative, such that it would assign a packet sequence to the negative class (non-sensitive web page) if it is not certain about its classification. This reduces FPR, thus increasing r-precision.

To optimize r-precision, we first ask the closed-world classifier to classify the element as usual. If it is a negative classification, we do not apply PO. Otherwise, if the classifier decides the element should be classified as a sensitive page (we refer to that class as the *assumed class*), we ask a PO whether or not we should *reject classification* of the assumed class. The PO may agree with the assumed class, or it may reject the assumed class and instead classify the element as non-sensitive. (Our POs do not change a sensitive classification to another sensitive classification.) This is shown in Figure 1. This strategy is conceptually similar to the Classify-Verify strategy described in Juarez et al.'s previous work [11].

Our POs are designed to be *classifier-agnostic*: they treat the classifier as a black box and they can be applied to all classifiers. Some POs are also parametrically tuneable to allow maximization of r-precision.

We start this section by motivating our work with an experiment on the baseline r-precision of WF classifiers without any POs (Section III-A). Then, we present three types of POs: confidence-based POs (Section III-B), distance-based POs (Section III-C), and ensemble POs (Section III-D). Our techniques are inspired by techniques used in clustering and ensemble learning.

We will describe in detail how each type of PO improves the precision of known attacks in the following sections, but first we front-load our presentation with Figure 2 showing our precision optimization in the $r = 20$ and $r = 1000$ scenarios. In each bar, the lighter area show the original precision without POs, the darker area show how much we increased precision

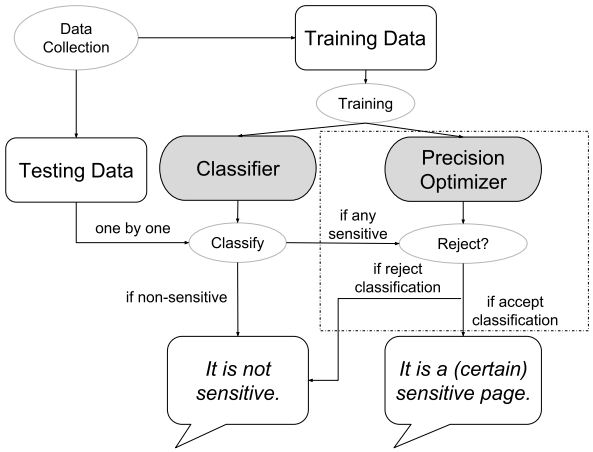


Fig. 1: Flowchart describing how we classify an element. Our Precision Optimizers (POs) modify classification to improve precision as represented by the dotted box.

using our POs, and an arrow (if any) indicates use of the more conservative Wilson method to obtain a lower bound for precision. These graphs clearly show that we can significantly increase precision in both scenarios and all attacks, with `Ha-kFP` and the ensemble method performing especially well in for $r = 1000$. We will describe each PO in detail in the following.

A. Baseline precision

We first present the precision of previous work, non-optimized, as a comparative basis. We experimented on the classifiers in Table III, using the experimental setup and methodology described in Section II-D. We tested two strategies seen in previous work, the non-monitored class strategy and the k-neighbors strategy.

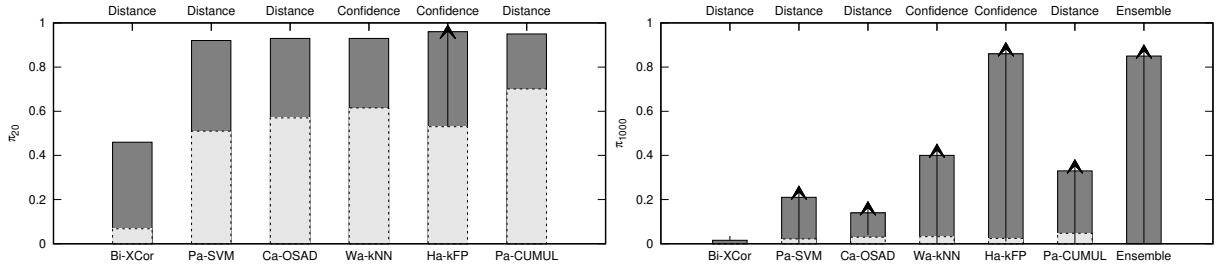
Non-monitored class strategy. This strategy adds an extra "non-monitored class" that includes all non-sensitive pages. In our case, there would be 100 sensitive (positive) classes and 1 non-sensitive (negative) class. The classifier is then asked to determine to which of the 101 classes each testing element should be assigned. We present the best results for 20-precision and 1000-precision in Table III.

Besides the six attacks we will optimize, we also include the Deep Fingerprinting attack (`Si-DF`) by Sirinam et al. [25], the current state of the art in website fingerprinting. We do not optimize `Si-DF` because our POs do not apply to its neural network mechanism. Indeed, non-optimized `Si-DF` has the best performance: with a TPR of 0.94 and a FPR of 0.005, previous works would have presented its precision (which is actually 1-precision) as $\pi_1 = 0.995$. However, this implicitly requires an unrealistic $r = 1$ value; in the more realistic (and much more difficult) $r = 1000$ scenario, its 1000-precision is still low at $\pi_{1000} = 0.143$.

Amongst the attacks to be optimized, the best-performing attacks are `Wa-kNN` and `Pa-CUMUL`, though 30–40% of

TABLE III: Summary of the seven WF attacks, as well as their baseline 20-precision (π_{20}) and 1000-precision (π_{1000}) using the non-monitored class strategy (i.e. no optimization).

Name	Classifier	Classification mechanism	π_{20}	π_{1000}
Bi-XCor [1]	Scoring	Cross correlation on inter-packet timing and lengths	$.068 \pm .001$	$.0007 \pm .0001$
Pa-SVM [20]	SVM	SVM on sequence features	$.510 \pm .013$	$.022 \pm .001$
Ca-OSAD [3]	SVM	Custom SVM kernel using Levenshtein distance	$.57 \pm .07$	$.03 \pm .01$
Wa-kNN [29]	kNN	Custom-weighted kNN on sequence features	$.615 \pm .012$	$.032 \pm .001$
Ha-kFP [7]	Random Forest	1000 decision trees on sequence features	$.53 \pm .01$	$.024 \pm .001$
Pa-CUMUL [19]	SVM	SVM on cumulative packet sizes	$.701 \pm .007$	$.047 \pm .002$
Si-DF [25]	NN	Neural Network on raw data	$.860 \pm .018$	$.143 \pm .04$



(a) 20-precision for each attack using the best PO.

(b) 1000-precision for each attack using the best PO.

Fig. 2: Overall best optimized 20-precision and 1000-precision results across the three types of POs. The best optimizer used is written above the chart for each bar. Lighter areas indicate the original precision of each attack in previous work, and the darker top area indicates the increased new precision achieved with the given PO. An arrow (if shown) jutting out of the bar indicates that the precision was obtained with the conservative Wilson method.

their classifications were wrong in the easier π_{20} scenario (when about 5% of the user's page visits are sensitive to the attacker). In the harder π_{1000} scenario, all attacks are imprecise. The results show that the trivial strategy of adding a non-sensitive class, used in previous work, is unable to achieve high precision with realistically high r.

All 100,000 packet sequences were part of the testing set for each attack except `Ca-OSAD`.[3] To form the training set, we used 10-fold cross validation, so that the training set would have 100x180+72000 elements and the testing set would have 100x20+8000 elements, and we would repeat this ten times with ten disparate testing sets. However, we found that this was impossible for one attack, `Ha-KFP`, which ran out of memory. For this attack only, we reduced the training set to 100x20+8000 elements during 10-fold cross validation, similar in size to their original data set.

k-neighbors strategy. Any classification strategy based on the proximity of the testing element to training elements can be enhanced with the k-neighbors strategy. When classifying a testing element, the classifier finds the k closest training elements to the testing element. The classifier will output a positive class only if all k closest training elements belong to that positive class. Otherwise, the classifier will output the negative class, rejecting classification. This strategy was seen in two previous attacks, `Wa-kNN` and `Ha-kFP` [7], [29]. Other attacks cannot use this strategy because they do not classify based on proximity.

[3]We tested `Ca-OSAD` only on 100x100+10000 elements because of the computational time involved to compute the custom SVM distance kernel, which scales with the square of the number of instances. On the full data set, it would've taken around 300,000 CPU hours.

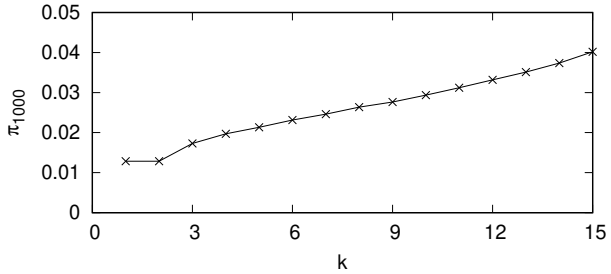


Fig. 3: 1000-precision for `Ha-kFP` under the k-neighbors strategy, varying k from 1 to 15.

We test the effectiveness of this strategy and show π_{1000} in Figure 3 for `Ha-kFP` (the better attack). We varied k from 1 to 15, the same range of values as both previous works. We see that increasing k improves precision within this range, but not sufficiently significantly; its highest value $\pi_{1000} = 0.04$ is still poor. This strategy also allows `Ha-KFP` to reach $\pi_{20} = 0.69$ at $k = 15$. Thus, we find that the non-monitored class strategy and the k-neighbors strategy are both insufficient to achieve high precision.

B. Confidence-based PO

To classify an input element P, some classifiers compute some matching function $match$ between P and all trained

classes C, and classify P to the class that maximizes the value of the function:

$$\underset{C}{\operatorname{argmax}}\, match(P, C)$$

As an example, in the following we describe the $match$ function used by Support Vector Machines (SVMs). SVMs are used by several WF attacks [3], [19], [20], [30]. We specify the $match$ function for other classifiers in the Appendix.

SVMs attempt to find an optimal separator between two classes in training. We denote $f_{C,C'}(P) \in \{C, C'\}$ as the classification output of an SVM trained on two classes C and C' when classifying P. For multi-class classification, SVMs can use the "one-against-one" classification system [4], as follows. To decide whether or not P belongs to C, the system computes a score $S(P, C)$:

$$S(P, C) = \left|\{C' \neq C | f_{C,C'}(P) = C\}\right|$$

In other words, $S(P, C)$ is the number of classes C' such that the SVM prefers C over C' for classifying P. In the end, the element is classified to the class with the highest aggregate score. Therefore, S fits the definition of the matching function for SVMs: $S(P, C) = match(P, C)$.

The matching function of a classifier can be interpreted as its confidence. If $match(P, C)$ is low for all classes, the classifier is reluctant to classify P to any class. Normally, the classifier will nevertheless choose the highest-scoring class. This causes false positives despite the classifier's uncertainty. A confidence-based PO would recognize such uncertainty, and instead classify the element as negative.

Our confidence-based PO works as follows. Suppose that the classes are ordered from highest match to lowest, such that C_1 matches P the most (i.e. C_1 is the assumed class) , followed by C_2, and so on, until C_{N+1}. We first scale all $match$ values linearly so that $match(P, C_1) = 1$ and $match(P, C_{N+1}) = 0$. For parameters K and M_{match}, we reject classification (classify the element as negative) if $\sum_{i=2}^{K+1} match(P, C_i) > K \cdot M_{match}$. In other words, we classify an element as negative if the top K competing classes to the assumed class have a mean $match$ score of M_{match} or above. We vary K and M_{match} and test their effects on r-precision. The output of $match$ is also useful in cases where the attacker may want to rank his classifications, or explicitly output the confidence of classification.

Which attacks apply?

All known attacks in the WF literature can be said to compute a $match(P, C)$ function for all C and choosing the highest-scoring class to classify P. Therefore, the confidence-based PO applies to all of our classifiers.

Results

We present the results of confidence-based PO with regards to how it improves r-precision of our chosen WF attacks in Table IV.

Optimized `Ha-kFP` becomes highly precise even under the difficult 1000-precision scenario. In the optimal case ($K = 3$, $M_{match} = 0.08$), we were able to achieve $\pi_{1000} > 0.86$ as shown in the table, and the attack achieved no false positives in all 80,000 non-monitored traces. In fact, if we were using the Wald method, its mean precision would be much higher (0.999), but we must use the more conservative Wilson method as explained in Section II-B.

TABLE IV: Best 20-precision (π_{20}) and 1000-precision (π_{1000}) with confidence-based PO.

Name	π_{20}	π_{1000}
Bi-XCor [1]	$.16 \pm .01$	$.0040 \pm .0002$
Pa-SVM [20]	$.60 \pm .02$	$.031 \pm .002$
Ca-OSAD [3]	$.77 \pm .10$	$.06 \pm .03$
Wa-kNN [29]	$\geq .93$	$\geq .40$
Ha-kFP [7]	$\geq .96$	$\geq .86$
Pa-CUMUL [19]	$.79 \pm .02$	$.076 \pm .005$

An attack may seem to achieve a low FPR but still be insufficient to obtain high precision under π_{1000}; for instance, optimized `Wa-kNN` achieved a false positive rate of 0.03%, but due to its poor true positive rate and the conservative nature of the Wilson method, it was still unable to beat optimized `Ha-kFP`. This shows the importance of the r-precision metric.

Some of the above attacks became many times more precise under π_{1000}. `Bi-XCor` became 6 times more precise; `Wa-kNN` became at least 13 times more precise; `Ha-kFP` became at least 40 times more precise. The SVM-based classifiers `Pa-SVM` and `Pa-CUMUL` gained a relatively small improvement using confidence-based metrics. This may be because the SVM $match$ function was not sufficiently informative about classifier confidence.

C. Distance-based PO

Several WF attacks use or induce a notion of distance between packet sequences when performing classification.[4] We found that those distances, when used to augment the normal classification algorithm of WF attacks, could serve to remove questionable positive classifications and thus improve precision.

We derived distances between packet sequences based on known attacks. For example, we defined a distance based on `Pa-SVM` by executing its feature extraction algorithm, and then applying the radial basis function on the extracted features. From each distance between packet sequences, we derive a distance between packet sequences and classes. Note that while classifiers always chose the class with the highest $match$ score, they did not choose the class with the shortest distance. For details on how we derived distances from WF attacks, we refer the reader to the Abstract. Then, we tested two different distance-based POs:

1) **Too-far PO**: We trained the PO by computing expected in-class distance (distance between packet sequences of

[4]In our work, we do not use the strict mathematical definition of a "metric" when referring to distances. In particular, many of our distances do not satisfy the triangle inequality in edge cases. Rather, the distance quantifies the difference between packet sequences from the classifier's perspective. We avoid use of the word "metric" for this reason, opting to use the word "distance".

TABLE V: Best π_{20} and π_{1000} with the too-far PO and too-close PO on a 100x100+10000 data set.

Name	Too-close PO			Too-far PO		
	π_{20}	π_{1000}	Best distance	π_{20}	π_{1000}	Best Distance
Bi-XCor [1]	$.42 \pm .02$	$.014 \pm .004$	Wa-kNN	$.46 \pm .02$	$.016 \pm 0.001$	Wa-kNN
Pa-SVM [20]	$.91 \pm .05$	$\geq .18$	Bi-XCor	$.92 \pm .04$	$\geq .21$	Bi-XCor
Ca-OSAD [3]	$.88 \pm .10$	$\geq .08$	Wa-kNN	$.93 \pm .09$	$\geq .14$	Bi-XCor
Wa-kNN [29]	$.93 \pm .04$	$\geq .21$	Bi-XCor	$.94 \pm .05$	$\geq .30$	Bi-XCor
Ha-kFP [7]	$.98 \pm .02$	$\geq .59$	Wa-kNN	$.97 \pm .03$	$\geq .39$	Wa-kNN
Pa-CUMUL [19]	$.95 \pm .04$	$\geq .33$	Bi-XCor	$.94 \pm .05$	$\geq .27$	Bi-XCor

the same class, for each class). If the distance of a testing packet sequence to the assumed class was more than M_{far} times the expected in-class distance, we rejected classification.

2) **Too-close PO**: If there were at least M_{close} classes that were closer to the packet sequence than its assumed class, we rejected classification.

Which attacks apply?

We tested five distances, each one based on a different attack; `Ha-kFP` did not produce a distance. All classifiers can be optimized with both distance-based POs, even if it itself does not produce a distance. This means that we have a total of 60 optimized classifiers (two types of POs, five distances, six classifiers).

Results

We present the results for the two distance-based POs in Table V for the full data set.

In both cases, we can achieve significant increases in both 20-precision and 1000-precision. In particular, `Ha-kFP` with `Wa-kNN` distance reached $\pi_{20} = .98$ and .97 respectively. It reaches $\pi_{1000} \geq .59$ and $\geq .39$ as well, representing a more than 20-fold increase in precision compared to no POs.

For both distance-based POs, the best distance to use was always the `Bi-XCor` distance or the `Wa-kNN` distance. This implies that they contained significant information not sufficiently incorporated in future attacks; for the case of `Bi-XCor`, this is perhaps because it had achieved a comparatively low TPR. Appropriately, the distance for `Bi-XCor` could not save itself from relatively poor precision; no other distance could, either. This would suggest that `Bi-XCor`'s distance is useful, but its classifier is weak.

For the too-close PO, in all cases $M_{close} = 1$ was optimal for precision. $M_{close} = 1$ means that the PO rejected the classification of any element that was closer to a different class than the assumed class. Increasing M_{close} increased both TPR and FPR in a ratio that was not favorable for precision.

For the too-far PO, the optimal value for M_{far} was slightly less than 1 for each of the above classifiers. A larger M_{far} weakened the precision optimizer, but a smaller M_{far} may cause TPR to drop too significantly. We show the effect of M_{far} on the lower bound of π_{1000} for `Ha-kFP` and the `Wa-kNN` distance in Figure 4. Peak precision occurred at $M_{far} = 0.84$. It makes sense that $M_{far} > 1$ would be imprecise: this represents a PO that would not reject classification even if the distance to its assumed class was

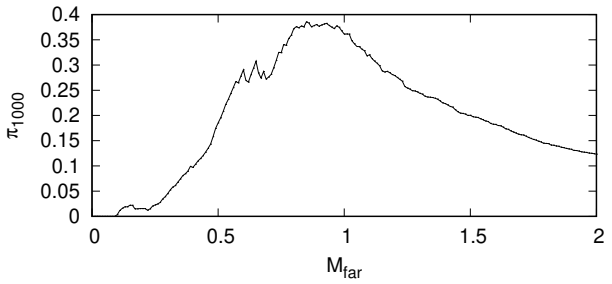


Fig. 4: Lower bound of π_{1000} for `Pa-CUMUL` with too-far PO, using the `Bi-XCor` distance on 100x100+10000 elements while varying M_{far}. The PO rejected any assumed class for which the testing element was at least M_{far} times as distant as the expected distance to that class.

greater than expected. There was almost no change in precision beyond $M_{far} = 2$ (when the PO almost never rejected a classification). It is interesting to see that a wide range of values for M_{far} gives similar results, suggesting the power of the PO is not a consequence of parameter overfitting.

D. Ensemble PO

In ensemble learning, multiple classifiers simultaneously classify the same testing element, and we decide the final class based on an aggregate of each classifier's individual classification. In adapting ensemble learning techniques for OWF, we hypothesized that disagreements between different classifiers could show a lack of confidence. Therefore, we should reject classification when different classifiers output different classifications.

We evaluate a simple bagging scheme for OWF. We trained all classifiers except `Ca-OSAD` (as it could not be trained on the full data set) separately on the same training set using 10-fold classification. Then, we take a subset of the classifiers and, for each testing element, we ask each of them to determine the assumed class. We rejected classification whenever there was no **unanimous** decision among all classifiers in the chosen subset of classifiers. Therefore, the more classifiers there are, the more conservative our classifications become.

We show the results for all 31 possible subsets of 5 classifiers in Table VI, focusing on the minimum π_{1000} (Wilson method). The black marker indicates which algorithms are used. Filled black rows represent that the algorithm forms part of the subset for that result. From top to bottom, they represent `Bi-XCor`, `Pa-SVM`, `Wa-kNN`, `Ha-kFP`, and `Pa-CUMUL`. Row i contains results for the use of i classifiers in ensemble.

TABLE VI: Lower bound for π_{1000} when a subset of the five WF attacks are used in ensemble PO. The marker besides each result indicates which of the five WF attacks are used (black bar = used, white bar = not used). From top to bottom, the bars represent Bi-XCor, Pa-SVM, Wa-kNN, Ha-kFP, and Pa-CUMUL in order.

.001	.021	.031	.006	.005					
.18	.20	.118	.225	.329	.337	.347	.399	.288	.387
.633	.630	.657	.696	.588	.675	.703	.619	.686	.689
.795	.764	.793	.845	.771					
.823									

For example, the second result in the third row represents the use of `Bi-XCor`, `Pa-SVM` and `Ha-kFP`. All results here exceed a recall of 0.2.

From Table VI, we see that the optimal precision is achieved when all except `Pa-SVM` are used for $\pi_{1000} \geq 0.845$. The best single classifier was `Wa-kNN`; the best two-classifier ensemble was to add `Ha-kFP`; the best three-classifier ensemble was to add `Pa-SVM`; but the best four-classifier ensemble does not include `Pa-SVM`, as it was too conservative and rejected too many correct classifications. Using more classifiers was more conservative and generally gave better precision, except in the five-classifier case where `Pa-SVM` rejected many correct positive classifications without helping to reject incorrect ones.

To expand on the above bagging scheme, we tested several other schemes including giving different weights to different classifiers and changing the number of votes required to accept a classification (instead of requiring all votes to accept a classification). Even with an exhaustive search of optimal parameters, none of these schemes outperformed the simple bagging scheme in the optimal case, so we omit these results.

IV. Different scenarios for a precise WF attack

To go beyond the standard open-world scenario, we test the effectiveness of a WF attack in several other scenarios in this section. We introduce each scenario in the following and analyze how high precision helps the classifier tackle the scenario.

A. Identifying a sensitive client

We want to know if the attacker could determine, after some period of observing the client, whether or not the client has a habit of visiting sensitive pages. This scenario simulates an attacker who wants to learn about the client's online behavior. For example, the attacker may want to figure out the client's political affiliation, romantic status, or other demographics by deciding if the client visits certain pages frequently. High recall and precision are both advantageous for this scenario, so we want to know if our preference for high precision helps classifiers.

Let us define a sensitive client as one who visits sensitive pages at a rate of b, and a non-sensitive client as one who does not visit sensitive pages. The attacker faces a binary classification problem to determine if a client is one of the above. To do so, the attacker observes each client for N page accesses, performs WF classification on their accesses, and gets some number x of sensitive page accesses. The attacker decides that the client is a sensitive client if $x \geq M_{identify}$. $M_{identify}$ is a parameter that controls the trade-off between identifying sensitive clients correctly and mistaking non-sensitive clients as sensitive clients.

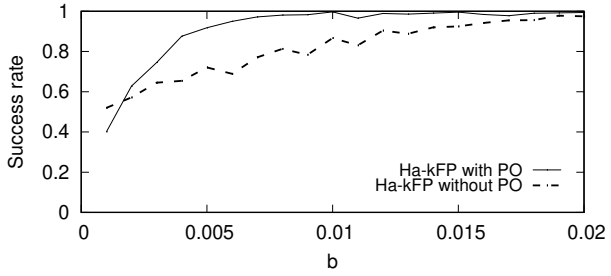


Fig. 5: Success rate of identifying a sensitive client for `Ha-kFP` with and without confidence-based PO, while varying b, the rate of sensitive page visits.

We compare `Ha-kFP` with and without confidence-based PO in Figure 5. We vary the base rate between 0.001 and 0.02 and model an attacker who has observed 1000 page accesses. Therefore, in the toughest $b = 0.001$ case, the client has only visited one sensitive page. To distinguish between sensitive and non-sensitive clients, we set $M_{identify}$ to be equal to $1000 \cdot (b \cdot (R_{TP} + R_{WP}) + (1 - b) \cdot R_{FP})/2$, i.e. the mean of the expected number of positive classifications for the sensitive client and the non-sensitive client. We can see that the optimized attack is much more successful at identifying sensitive clients when b is low, with a 63% chance of correct identification at $b = 0.002$, up to 99% at $b = 0.01$. Not only does the original attack achieve less success at identifying sensitive clients, it also more frequently falsely identifies non-sensitive clients as sensitive ones: the rate increases with lower b, from 25% at $b = 0.01$ to 43% at $b = 0.001$ (not shown in the graph). This rate cannot be reduced without also reducing the true identification rate. The optimized attack did not make false positives.

B. Attacking defenses

A number of defenses against WF have been proposed for anonymity technologies like Tor. Much like WF attacks, these defenses are almost always evaluated with recall: a good defense would be judged by its ability to decrease the recall of all classifiers. We wanted to know the precision of our

TABLE VII: Best 20-precision (π_{20}) with confidence-based PO on Ha-kFP against three defenses: Random Padding, Tamaraw and WTF-PAD. Bandwidth (B/W) and Time overhead are also given.

Name	Original	With PO	Overhead	
	π_{20}	π_{20}	B/W	Time
Random Padding	$.17 \pm .01$	$.86 \pm .08$	50%	50%
Tamaraw [2]	$.0088 \pm .0004$	$.036 \pm .005$	91%	247%
WTF-PAD [12]	$.24 \pm 0.01$	$.96 \pm .02$	32%	0%

optimized attacks on those defenses and whether or not their precision could be significantly deterred by defenses.

We evaluated three WF defenses: Random Padding set to produce 50% bandwidth and time overhead by adding random packets, Tamaraw by Cai et al. [2], and WTF-PAD by Juarez et al. [12]. Bandwidth overhead is equal to the percentage of extra packets added by the defense, and time overhead is equal to the percentage of extra time required to load each web page.

We present the results in Table VII. For these results only, we lower the minimum recall requirement from 0.2 to 0.02, because no attack can achieve a higher recall than 0.06 for Tamaraw. For WTF-PAD, we could only test on 100x100+10000 because it was very slow.

We see that WTF-PAD is slightly less capable of reducing the precision of an attacker compared to Random Padding, and much less capable compared to Tamaraw, though it is the cheapest in overhead. Even without any PO, attacking WTF-PAD was relatively easy compared to Tamaraw, though Tamaraw is far more expensive, especially in time overhead. Recent work by Sirinam et al. has shown high TPR against WTF-PAD [25]. With precision optimization, Ha-kFP can be precise (for $r = 20$) on both WTF-PAD and Random Padding; a five-fold jump in precision is possible on Tamaraw, though the best attack is still highly imprecise. Tamaraw cannot be defeated, though its incredible cost in overhead has hampered its adoption.

We did not evaluate any "targeted defenses": a targeted defense allows the client to choose which page to mimic while accessing specific pages. While targeted defenses give clients the advantage of creating a specific cover story, rather than a series of randomly perturbed packet sequences, currently there is no known mechanism to automatically choose correct targets to mimic. Wrongly chosen targets could significantly impede the ability of the defense to lower precision, so we cannot evaluate them fairly. These defenses include Glove [17], Walkie-Talkie [32], and Decoy pages [20].

C. Attacking different data sets

Like most other works on WF, we perform the main evaluation of our work on a data set that consists of 100 monitored web pages chosen from the top pages. Here, we consider whether variations in the data, corresponding to realistic scenarios a WF attacker would want to tackle, would change the precision of our optimized classifiers.

HTTP/1.1 and HTTP/2. Our data set consists of a mix of HTTP/1.1 (30%) and HTTP/2 (70%). HTTP/2, the newer version, changes how resources are loaded to encourage same-stream parallelism. We separate our data set into two parts for those two versions and compare their recall and precision for Ha-kFP. Before optimization, recall on the two sets was respectively 0.91 and 0.88, while 1000-precision was 0.023 and 0.027. After optimization, recall was 0.44 and 0.37, while 1000-precision was respectively > 0.85 and > 0.84. There is almost no difference between precision on the two data sets, while the HTTP/1.1 data set showed a slightly higher recall. It is possible that the slight difference is due to HTTP/2, but it is also possible that the nature of the web pages was simply different.

Same-domain pages. We want to test if the attacker can precisely separate web pages of the same type and domain. We collect a new Wikipedia data set for this purpose: our 100 monitored classes are all Wikipedia pages concerning sensitive topics, for which we collect 100 instances each, and we also collect 10,000 non-monitored Wikipedia traces corresponding to random walks of Wikipedia's pages following links starting from the main page. Details of data collection can be found in the Appendix. This is a more difficult task since the pages are highly similar in size and structure.

Our results show that original Ha-kFP can achieve $\pi_{20} = 0.09$ and $\pi_{1000} = 0.002$ on this data set with a low TPR of 0.52, while confidence-based optimized Ha-kFP can achieve $\pi_{20} = 0.28$ and $\pi_{1000} = 0.008$ lowering TPR to 0.2, with $K = 3$ and $M_{match} = 0.28$. Although we are not able to achieve high precision in this challenging problem, our methods do bring a three- to four-fold increase in precision. The difficulty of the above task can be seen in the following statistics: the mean difference in the number of cells between a randomly chosen monitored instance and a non-monitored one was 810 (compared to a mean size of 1720 cells), while in the original data set it was 4260 (compared to a mean size of 4120 cells). This reflects the fact that monitored and non-monitored pages were much more similar in the Wikipedia data set. The web pages were also smaller, thus leaking less information to classify.

Different monitored pages. We chose the top 100 Alexa pages as our monitored sensitive set chiefly to ensure reproducibility as lesser ranked pages often had a shorter lifespan. If the site is taken down, the results can no longer be reproduced. To address questions as to whether high-precision attacks on the top 100 Alexa pages would be reproducible on other pages, we collect a different data set here corresponding to 100 randomly chosen pages from the top 100,000 instead, similarly with 200 instances each, making up a 100x200+80000 data set as before. We use Ha-kFP with confidence-based PO, the best optimized attack available, and achieve $\pi_{1000} > 0.81$ on this data set, a slightly diminished but still highly precise result. This shows that our methods are similarly precise on other web pages.

TABLE VIII: TPR of each of the four classes and what portion of the false positives each class contributed to ("FPR contribution"). TPR was calculated for a modified data set with these 4 new classes and 96 original classes.

Name	Original		With PO	
	TPR	FPR contribution	TPR	FPR contribution
AJAX1	0.54	0.9%	0.11	0%
AJAX2	0.60	0.7%	0.15	0%
LINKS1	0.87	0.6%	0.02	0%
LINKS2	0.9	1.4%	0.06	0%

Active client. we want to test the precision of an attacker against an *active* client who is generating random events through page browsing, rather than a static client who visits a page and does not act on it. We chose two sites for this task; for the first site "AJAX", the active client randomly scrolls down the web page, causing more content to be loaded because of its AJAX code; in the other site "LINKS", the active client randomly clicks on links to browse topics. For each site, we generated two classes, corresponding to a fast client (AJAX1, LINKS1) and a slow client (AJAX2, LINKS2). Exact details are in the Appendix.

We replaced 4 of the original 100 classes with these new classes and asked `Ha-kFP` to classify the new problem precisely. (The data set size is the same at 100x200+80000). We present the results in Table VIII. We show TPR for the four chosen classes before and after optimization, as well as what portion of the false positives they each contributed to. Since there are 100 classes, each class on average contributes to 1% of the false positives.

We found that AJAX1 and AJAX2 were especially difficult to classify, as their true positive rate was low even in the original case. This was probably due to the highly random network activity. None of these four classes were especially responsible for false positives in the modified data set. After optimization, the true positive rate of each class was reduced below the overall mean true positive rate (0.38), suggesting that the four classes were somewhat harder to classify correctly. Overall, however, the classifier still achieved $\pi_{20} > 0.99$ and $\pi_{1000} > 0.85$, about as good as the original data set. The four new classes were difficult to classify, but they did not end up hurting precision.

V. Discussion

A. Does Website Fingerprinting scale?

We have shown that optimized classifiers can be highly precise in our large open world. Yet, the reader may comment that our experimental open world is tiny compared to the actual open world (with over a billion pages). Can these results be extended to the actual open world?

Since any experiment's size is invariably limited when compared to the space of all potential inputs, what is crucial is to ensure that the experiment *procedure* correctly simulates a realistic attack scenario. Our open-world experiment is correct because we are pessimistically simulating an attacker who has

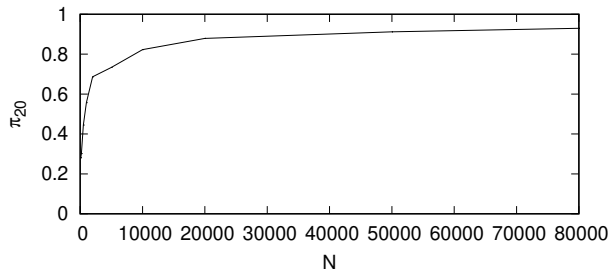


Fig. 6: 20-precision for `Wa-kNN` on a data set of 100x200+N elements using 10-fold cross-validation, where N is the size of the open-world class ranging from 100 to 80,000.

no knowledge whatsoever of the testing open world; at most, he is only allowed to train on his own toy open world, which does not intersect with the client's testing open world. The difference in size between the training and testing worlds does not affect the correctness of this procedure. For this reason, we only need to ensure that **the classifier should never be tested on the same non-sensitive pages it has trained on**.

To achieve this experimentally, we visited each non-sensitive page only once, and split the training and testing set (generally using 10-fold cross validation, though the training set was smaller for some attacks). We also filtered out pages with similar domain names so that they would not both appear in our non-monitored set. Since the classifier has no knowledge whatsoever of the non-sensitive pages in the testing set, its decision that a testing element should be classified as non-sensitive is not based on any specific knowledge about the page it came from. In other words, the actual open world — and its larger size — has no impact on the classifier's success.

Some previous work increased the size of the open world and found that accuracy and precision decreased [19], [20]. We argue that this was only the case because they did not keep r constant and implicitly increased r when increasing the open world size. To show this, we also increased the size of the open world, but we measured π_{20} (20-precision), holding the base rate of visiting non-sensitive pages constant. We evaluated optimized `Wa-kNN` on a data set of 100x200+N elements, where N, the open world size, varied from 100 to 80,000, with 10-fold cross-validation. We show the results in Figure 6, which shows that a larger open world set size will increase π_{20} from 0.28 to 0.93, as may be expected of a competent classifier. Other classifiers show similar results, though we specifically picked `Wa-kNN` as it does not have to wrestle with class size imbalance.

B. What is the value of r?

The value of r parametrically captures an aspect of client behavior: the ratio between the probability a client would visit a non-sensitive page compared to that of a sensitive page. We never assume that the attacker knows or needs to estimate the value of r (his strategy is the same no matter what the client does). For simplicity of presentation we decided to focus on two scenarios, $r = 20$ and $r = 1000$. Here we justify the

choice of these parameters by presenting data from real web browsers.

We wrote a small executable to read browser history files (Firefox and Chrome) and distributed it to volunteering acquaintances. We created two lists of 100 web pages each, respectively the top 100 web pages, and 100 sensitive web pages that are banned in certain countries including the U.K., India, Australia, New Zealand, U.S., and Russia.[5] The latter set mostly consists of file sharing sites and streaming services, and we selected the 100 most popular ones (by Alexa rank) amongst those. All clients were fully informed of the above and how the data would be presented, and approval was obtained from the relevant institutional ethics review boards. We included a hash as a rudimentary way to stop participants from editing their data.

Since Tor Browser keeps no history and no logins between sessions, users generally have to visit the home page of a web site (sometimes to log in) before visiting any of its other web pages. This is not the case in normal browsing, so we took this into consideration to avoid under-counting the number of page visits (thus overestimating r). We counted the number of page visits in two different ways. First, the "Site" method counts all page visits corresponding to that site, not just the specific page. This likely over-counts the number of actual visits to these pages. Second, the "Session" method also counts all page visits to that site, but only once in a given session. We define a session to be any continuous series of page visits with less than 5 minutes of inactivity between two consecutive visits. For example, if a user visits ten profile pages on `facebook.com` in one session without going to the home page (because they are already logged in), this counts as ten visits in the "Site" method and one visit in the "Session" method. The latter more realistically represents private browsing and Tor Browser. Exact page visits are counted in any case. While it can be argued that r extracted from normal browsing would differ from r extracted from Tor Browser users, we have no way of obtaining the latter and the former nevertheless represents real browsing activity.

We show values of r in Table IX. We see that r values for the top sites generally varied from 10 to 30, while the sensitive values varied more significantly from several hundred to several thousand. If we treat the "Session" value as more realistic, then monitoring these particular sensitive pages would probably require the attacker to succeed at $r = 1000$ to $r = 2000$. The value of 54338 for one data set is questionable because it corresponds to a single recorded visit of a sensitive page; a few more visits would significantly decrease the value.

For presentation, we chose to fix r to two specific values earlier, $r = 20$ and $r = 1000$ to represent popular pages and sensitive pages respectively. Here, we want to vary r to examine its effects on r-precision. Since our classifiers are not dependent on r, we take the TPR, WPR, and FPR of the best previous result, `Ha-kFP` using confidence-based PO, and re-calculate the conservative lower bound of r-precision using the Wilson method while varying r. We show these results in Figure 7.

[5]We intentionally did not include any pages banned in China as it does not represent what our participants would consider sensitive.

TABLE IX: Values of r for several participants, based on two data sets and two methods of counting r, "Site" and "Session".

# of pages	Top pages		Sensitive pages	
	Site	Session	Site	Session
127095	6	15	281	1896
54338	10	33	2089	54338
116566	5	14	485	2534
13161	20	20	258	274

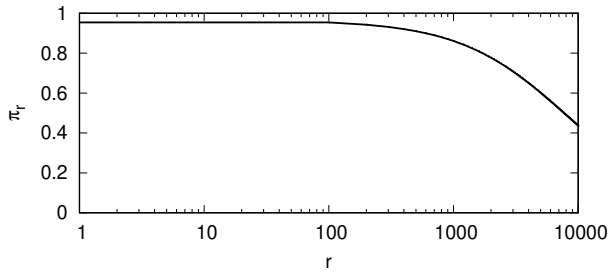


Fig. 7: Lower bound for π_r by the Wilson method when we vary r from 1 to 10000. Note that the x-axis is logarithmic.

First, there is only a very slight decrease of r-precision from $r = 20$ ($\pi_{20} > 0.96$) to $r = 100$ ($\pi_{100} > 0.95$). $r = 100$ represents a moderately difficult scenario, since the base rate we measure is the sum of 100 pages. For example, if 10% of the population regularly visit a page, and if they visit 30 pages per day, the pages they visit once per month would match the rate of the sensitive pages examined by the $r = 100$ scenario. Our success with the $r = 20$ scenario can be extended to the $r = 100$ scenario. Second, there is a more significant drop in r-precision at $r = 10000$. This is in fact an experimental limitation, as the value is obtained by the conservative Wilson method. At $r = 2000$, we still achieve $\pi_{2000} > 0.78$ with our current methods and data set.

C. Is Website Fingerprinting realistic?

Generally, WF works make several assumptions to present results. These assumptions include the use of a cold cache, freshness of the training set and the lack of noise that could impede classification. Multiple works [11], [31] have noted that violating these assumptions would cause recall to drop. We note that assuming the use of a cold cache is reasonable for Tor (as it does not keep cache in disk), and Wang et al. [31] have shown that the training set can be kept sufficiently fresh for classification. Here, we discuss issues related to the last assumption, the lack of noise.

In machine learning, noise could refer to several different aspects of data. First, there may be label noise — mislabeled elements in the training set. This is irrelevant to our WF attack model as the attacker visits specific pages himself to collect training data. Second, perturbations to the data may be introduced by differences between the experimental setting

and the real client setting, such as network conditions and browser settings. Much of these differences are ameliorated when experimenting on Tor, where network conditions are already random and the browser cannot be easily configured differently. The browser also does not keep cookies across sessions, including login cookies; for example, a user must login to a social media page every time she starts the browser again, thus going through the easily-fingerprinted front page. Third, user actions may cause data to load in a different way compared to the experimental client (which is static and does not act beyond loading a page). For example, the user may be listening to music in another tab, which causes network activity. Wang et al. [31] showed that classification is only impeded when the bandwidth rate of noise is very high, possibly from a video or a file download. Our work addresses several types of active users to find that they will not impede precision, though we cannot claim to have a complete investigation of all types of user activity.

VI. Related Work

As the presentation of our results has already included descriptions of much previous work, we offer only a brief overview of related work in this section focusing on WF attacks. Cheng et al. [5], Sun et al. [26], Hintz et al. [9], and Bissias et al. [1] were some of the first to show successful classifiers to determine which page someone is visiting based on traffic patterns. Later works referred to this traffic analysis problem as website fingerprinting.

The original paper on Tor considered traffic analysis to be a serious threat [6], though no attack had been successful on Tor at that time as Tor equalized cell sizes. In particular, Herrmann et al. [8] showed that their attack, as well as Liberatore and Levine's attack [13], did not succeed against Tor, though their attacks were able to beat SSH and selected VPNs. Lu et al. [15], Panchenko et al. [20] and Cai et al. [3] were some of the first to show success against Tor. Attack accuracy was improved and computational time was decreased by Wang et al. [29], Hayes et al. [7], another work by Panchenko et al. [19], and more recently with deep learning by Rimmer et al. [23] and Sirinam et al. [25]. These works also attempted to lower the false positive rate for open-world effectiveness.

Some previous WF works have discussed the base rate fallacy, though they did not include the base rate in their analysis or experiments. Panchenko et al. [19] discussed issues with precision, though their attack was not precise (as seen in our results as well), and they found that WF attacks generally would fail in the large open world if they are not precise. This important point underlines our paper's motivation. Hayes et al. [7] are able to achieve a more precise attack using the k-neighbors strategy, as did an earlier work by Wang et al. [29]. We showed that our POs are more effective and can be applied to any attack.

VII. Conclusion and Future Work

This work tackles the open problem of open-world website fingerprinting (OWF). We found that OWF classifiers were not precise considering the realistically low base rates at which people normally visit sensitive web pages. This implied that WF would only succeed in identifying highly popular web pages. We formulate r-precision (π_r), the percentage of sensitive classifications that are correct if the client visits r times as many non-sensitive pages as sensitive pages. We address a confusion between experimental data set sizes and real world data set sizes sometimes found in previous work, and show the importance of distinguishing between wrong positives and false positives in calculating precision.

As no previous attack was precise under $r = 1000$, we present three classes of POs to improve the precision of WF classifiers. Our confidence-based POs ask classifiers to output the degree of confidence they have in their classifications, and we reject classifications that are not confident enough. Our distance-based POs reject classifications of testing elements that are too far from the assumed class. Our ensemble-based POs reject elements for which a chosen set of classifiers did not unanimously agree on the assigned class. While confidence-based `Ha-KFP` performed the best at $\pi_{1000} > 0.86$, distance-based POs were able to allow almost any attack to achieve high π_{20} precision, and ensemble-based POs were nearly as precise as confidence-based `Ha-kFP` with the added benefit of requiring no parametrization.

Previous authors have shown that a number of problems in realistic scenarios would lower recall, such as noisy packet sequences, poor training sets and multi-tab browsing; solutions have been proposed in previous works. We evaluated one particular scenario involving active users with random behavior to show that we can still achieve high precision, but we do not know if other realistic problems and their solutions would affect precision. One way to definitely demonstrate the practicality of OWF would be to create a private Tor guard that performs website fingerprinting on consenting Tor clients, telling them which sensitive web pages they visited and asking them if it is correct.

On the flip side, defenses should also be designed to minimize the attacker's precision. One approach to designing defenses is to create "anonymity sets" of web pages that look the same to the attacker after padding. This approach provides an upper bound on the maximum recall of any attacker, but not precision. We should consider re-designing such defenses to provide an upper bound on the maximum r-precision instead.

Our data and code can be found at:

https://github.com/OpenWF/openwf.git

REFERENCES

[1] BISSIAS, G. D., LIBERATORE, M., JENSEN, D., AND LEVINE, B. N. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Privacy Enhancing Technologies* (2006), Springer, pp. 1–11.

[2] CAI, X., NITHYANAND, R., WANG, T., GOLDBERG, I., AND JOHNSON, R. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 21st ACM Conference on Computer and Communications Security* (2014).

[3] CAI, X., ZHANG, X. C., JOSHI, B., AND JOHNSON, R. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the 19th ACM Conference on Computer and Communications Security* (2012), pp. 605–616.

[4] CHANG, C.-C., AND LIN, C.-J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST) 2*, 3 (2011), 27.

[5] CHENG, H., AND AVNUR, R. Traffic Analysis of SSL-Encrypted Web Browsing. http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps, 1998.

[6] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium* (2004).

[7] HAYES, J., AND DANEZIS, G. k-Fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *Proceedings of the 25th USENIX Security Symposium* (2016).

[8] HERRMANN, D., WENDOLSKY, R., AND FEDERRATH, H. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security* (2009), pp. 31–42.

[9] HINTZ, A. Fingerprinting Websites Using Traffic Analysis. In *Privacy Enhancing Technologies* (2003), Springer, pp. 171–178.

[10] JANSEN, R., JUAREZ, M., GALVEZ, R., ELAHI, T., AND DIAZ, C. Inside job: Applying traffic analysis to measure Tor from within. In *Proceedings of the 25th Network and Distributed System Security Symposium* (2018).

[11] JUAREZ, M., AFROZ, S., ACAR, G., DIAZ, C., AND GREENSTADT, R. A Critical Evaluation of Website Fingerprinting Attacks. In *Proceedings of the 21st ACM Conference on Computer and Communications Security* (2014).

[12] JUAREZ, M., IMANI, M., PERRY, M., DIAZ, C., AND WRIGHT, M. Toward an Efficient Website Fingerprinting Defense. In *Computer Security–ESORICS 2016*. Springer, 2016, pp. 27–46.

[13] LIBERATORE, M., AND LEVINE, B. N. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (2006), pp. 255–263.

[14] LINUX FOUNDATION. Let's Encrypt Stats — Percentage of Web Pages Loaded by Firefox Using HTTPS. https://letsencrypt.org/stats, 2018. Accessed Dec. 2018.

[15] LU, L., CHANG, E.-C., AND CHAN, M. C. Website Fingerprinting and Identification Using Ordered Feature Sequences. In *Computer Security–ESORICS 2010*. Springer, 2010, pp. 199–214.

[16] NATIONAL RESEARCH COUNCIL. *Strengthening forensic science in the United States: a path forward*. National Academies Press, 2009.

[17] NITHYANAND, R., CAI, X., AND JOHNSON, R. Glove: A Bespoke Website Fingerprinting Defense. In *Proceedings of the 13th ACM Workshop on Privacy in the Electronic Society* (2014).

[18] OVERDORF, R., JUAREZ, M., ACAR, G., GREENSTADT, R., AND DIAZ, C. How Unique is Your .onion?: An Analysis of the Fingerprintability of Tor Onion Services. In *Proceedings of the 24th ACM Conference on Computer and Communications Security* (2017).

[19] PANCHENKO, A., LANZE, F., ZINNEN, A., HENZE, M., PENNEKAMP, J., WEHRLE, K., AND ENGEL, T. Website Fingerprinting at Internet Scale. In *Proceedings of the 23rd Network and Distributed System Security Symposium* (2016).

[20] PANCHENKO, A., NIESSEN, L., ZINNEN, A., AND ENGEL, T. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Proceedings of the 10th ACM Workshop on Privacy in the Electronic Society* (2011), pp. 103–114.

[21] PERRY, M. A Critique of Website Traffic Fingerprinting Attacks. https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks, November 2013. Accessed Feb. 2015.

[22] PIETRASZEK, T. Using adaptive alert classification to reduce false positives in intrusion detection. In *Recent Advances in Intrusion Detection* (2004), pp. 102–124.

[23] RIMMER, V., PREUVENEERS, D., JUAREZ, M., VAN GOETHEM, T., AND JOOSEN, W. Automated website fingerprinting through deep learning.

[24] SAKS, M. J., AND KOEHLER, J. J. The coming paradigm shift in forensic identification science. *Science 309*, 5736 (2005), 892–895.

[25] SIRINAM, P., IMANI, M., JUAREZ, M., AND WRIGHT, M. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 25th ACM Conference on Computer and Communications Security* (2018), ACM, pp. 1928–1943.

[26] SUN, Q., SIMON, D. R., WANG, Y.-M., RUSSELL, W., PADMANABHAN, V. N., AND QIU, L. Statistical Identification of Encrypted Web Browsing Traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy* (2002), IEEE, pp. 19–30.

[27] SWETS, J. A. Roc analysis applied to the evaluation of medical imaging techniques. *Investigative radiology 14*, 2 (1979), 109–121.

[28] WANG, T. *Website Fingerprinting: Attacks and Defenses*. PhD thesis, University of Waterloo, 2016.

[29] WANG, T., CAI, X., NITHYANAND, R., JOHNSON, R., AND GOLDBERG, I. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium* (2014).

[30] WANG, T., AND GOLDBERG, I. Improved Website Fingerprinting on Tor. In *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society* (2013), pp. 201–212.

[31] WANG, T., AND GOLDBERG, I. On Realistically Attacking Tor with Website Fingerprinting. In *Privacy Enhancing Technologies* (2016), Springer.

[32] WANG, T., AND GOLDBERG, I. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *Proceedings of the 26th USENIX Security Symposium* (2017).

APPENDIX

Here we describe each of the six previously published attacks we tested with our POs. The attacks are Bi-XCor [1], Pa-SVM [20], Ca-OSAD [3], Wa-kNN [29], Ha-kFP [7] and Pa-CUMUL [19]. We describe how each classifier represents packet sequences P as $R(P)$, the distance $d(P, P')$ between two packet sequences P and P', the training and the testing procedures. We describe the testing procedure by specifying $match$ (as explained in Section III-B); each classifier assigns the element to the class that scores the highest with $match$.

We denote packet sequences as $P = \langle p_1, p_2, \ldots, p_n \rangle$, where $p_i = (t_i, \ell_i)$, t_i is the interpacket time between p_{i-1} and p_i, and ℓ_i is the byte length of packet p_i, with positive packet lengths representing outgoing packets from the client and negative packet lengths representing incoming packets to the client. With Tor cells, $\ell_i \in \{-1, 1\}$ as all cells have the same size. P represents the WF attacker's information, and he attempts to deduce which web page it came from.

Bi-XCor

Representation. We split $R(P) = (R_t(P), R_\ell(P))$, where:

$$\begin{aligned} R_t(P) &= \langle t_1, t_2, \ldots, t_n \rangle \\ R_\ell(P) &= \langle \ell_1, \ell_2, \ldots, \ell_n \rangle \end{aligned}$$

Distance. Consider two lists a and b with mean $\bar{a}$, $\bar{b}$ and standard deviation σ_a, σ_b respectively. We define the cross-correlation function $X(a, b)$ between them:

$$X(a,b) = \frac{\sum_{i=1}^{\min(|a|,|b|)} (a_i - \bar{a})(b_i - \bar{b})}{\min(|a|, |b|) \cdot \sigma_a \cdot \sigma_b}$$

We have:

$$d(P, P') = 2 - X(R_t(P), R_t(P')) - X(R_\ell(P), R_\ell(P'))$$

Training. We represent each class C as $R(C) = (R_t(C), R_\ell(C))$, where the i-th element of $R_t(C)$ is the mean of all t_i for training packet sequences from class C, and similarly for $R_\ell(C)$.

Testing.

$$match(P, C) = d(R(P), R(C))$$

Pa-SVM

Representation. We extract a number of features from each packet sequence related to packet ordering, directions, and sizes: $R(P) =< f_1, f_2, ..., f_{|F|} >$. To see the list of features, refer to the original work [20] or our code.

Distance. We use the radial basis function with $\gamma = 2^{-25}$ to compute distances between the feature representations of packet sequences. The distance is:

$$d(P, P') = 1 - e^{-\gamma||R(P)-R(P')||^2}$$

Training. We train an SVM on the above pairwise distances by finding support vectors which separate classes.

Testing. The matching function uses one-against-one SVM classification as described in Section III-B.

Ca-OSAD

Representation.

$$R(P) = \{\ell_1, \ell_2, \ldots\}$$

Distance. We compute the pairwise distance between packet sequences P and P' as:

$$d(P, P') = 1 - e^{-2 \cdot OSAD(P,P')^2 / \min(|P|,|P'|)}$$

In the above, $OSAD(P, P')$ is the Optimal String Alignment Distance between $R(P)$ and $R(P')$.

Training. We train an SVM using the custom kernel calculated from the above pairwise distances.

Testing. The matching function uses one-against-one SVM classification as described in Section III-B.

Wa-kNN

Representation. We extract a number of features from each packet sequence related to packet ordering, directions, and sizes: $R(P) =< f_1, f_2, ..., f_{|F|} >$. To see the list of features, refer to the original work [29] or our code.

Distance. We use a weighted L_1-distance between P and P':

$$d(P, P') = \sum_{i=1}^{|F|} w_i |f_i - f'_i|$$

Training. We learn weights w_i that optimize the accuracy of the weighted distance.

Testing.

$$match(P, C) = \min_{P' \in C} d(P, P')$$

Ha-kFP

Representation. We extract features from each packet sequence, similar to Wa-kNN. To see the list of features, refer to the original work [7] or our code.

Distance. Ha-kFP does not produce a distance.

Training. We train a Random Forest classifier with 1000 decision trees, where each tree draws a random sample of the input elements with replacement, resulting in a sample of equal size to the input. Each leaf L of a decision tree records $L(x)$, the number of training samples of each class that fell in that leaf, for class x.

Testing.

If P falls in leaf L for decision tree i, we calculate $match_i(P, C) = L(C)/\sum_x L(x)$. Then

$$match(P, C) = \sum_{i=1}^{1000} match_i(P, C)$$

Pa-CUMUL

Representation. We extract features from each packet sequence, based on total size, time, and 100 linear interpolations of aggregated packet sizes. To see the list of features, refer to the original work [19] or our code.

Distance. We use the radial basis function with $\gamma = 2^{-28}$ to compute distances between the feature representations of packet sequences. The distance is:

$$d(P, P') = 1 - e^{-\gamma||R(P)-R(P')||^2}$$

Training. We train an SVM on the above pairwise distances by finding support vectors which separate classes.

Testing. The matching function uses one-against-one SVM classification as described in Section III-B.

Distances

For our distance-based POs, we derived a distance between packet sequences P, P' based on five previous WF attacks: `Bi-XCor`, `Pa-SVM`, `Ca-OSAD`, `Wa-kNN` and `Pa-CUMUL`. The distance is equivalent to $d(P, P')$ as written above for each WF attack. Then, we derived a distance between packet sequences P and classes C based on the distance between packet sequences as follows. We denote $C[:N]$ to mean the N closest elements to P in C, and $C[N]$ to mean the N-th closest element to P in C.

1) $d(P, C) = \sum_{P' \in C} d(P, P')/|C|$.
2) $d(P, C) = \sum_{P' \in C[:5]} d(P, P')/|C|$.
3) $d(P, C) = \sum_{P' \in C[:25]} d(P, P')/|C|$.
4) $d(P, C) = d(P, C[1])$.
5) $d(P, C) = d(P, C[5])$.
6) $d(P, C) = d(P, C[25])$.

DATA COLLECTION

We collected the Wikipedia data set (100x100+10000) as follows. For the non-monitored pages, we started from one of five pages: the main page, the portal of current events, the "United States" page, the "India" page, and the "World War II" page. Then, we randomly traversed links on the page to other Wikipedia articles, avoiding special pages and pages corresponding to dates. The random walk lasted for a uniform length between 1 to 20 steps, after which we would restart at one of the above five pages. This was meant to simulate a client who surfed Wikipedia pages starting from a topic of interest. For the monitored pages, we manually chose ten politically/culturally sensitive topics, and for each topic we manually chose ten relevant pages. We share the list of all pages in our data set:

https://github.com/OpenWF/openwf.git

We collected the different data set used in Section IV-C as follows. For AJAX1 and AJAX2, we visited `reddit.com` and scrolled down. AJAX1 scrolls down every 1 to 5 seconds for up to 1000 pixels; AJAX2 scrolls down every 1 to 10 seconds for up to 500 pixels. Each stops after 20 to 40 seconds. LINKS1 and LINKS2 are based on `en.wikipedia.org`, and follows a random walk. LINKS1 visits a new link every 1 to 5 seconds; LINKS2 visits a new link every 1 to 10 seconds, and each stops after 20 to 40 seconds. All randomness is uniform.