Jean-Pierre Smith*, Prateek Mittal, and Adrian Perrig

Website Fingerprinting in the Age of QUIC

Abstract: With the meteoric rise of the QUIC protocol, the supremacy of TCP as the de facto transport protocol underlying web traffic will soon cease. HTTP/3, the next version of the HTTP protocol, will not support TCP. Current website-fingerprinting literature has ignored the introduction of this new protocol to all modern browsers. In this work, we investigate whether classifiers trained in the TCP setting generalise to QUIC traces, whether QUIC is inherently more difficult to fingerprint than TCP, how feature importance changes between these protocols, and how to jointly classify QUIC and TCP traces. Experiments using four state-of-the-art website-fingerprinting classifiers and our combined QUIC-TCP dataset of ∼117,000 traces show that while QUIC is not inherently more difficult to fingerprint than TCP, TCP-trained classifiers may fail to detect up to 96% of QUIC visits to monitored URLs. Furthermore, classifiers that take advantage of the common information between QUIC and TCP traces for the same URL may outperform ensembles of protocol-specific classifiers in limited data settings.

Keywords: Traffic Analysis, Website Fingerprinting, QUIC, Wireguard

DOI 10.2478/popets-2021-0017
Received 2020-08-31; revised 2020-12-15; accepted 2020-12-16.

1 Introduction

While the Internet continues to entrench itself in our daily lives, users have begun to grasp the extent of their digital footprints as both industry and governments alike encroach upon privacy in an effort to extract value from the mass of personal data. Privacy-enhancing technologies, such as Tor [1] and VPNs, allow users to browse the web in relative privacy by hiding communication meta-data, such as destination IP addresses, using intermediate proxies. However, the fundamental trade-offs among latency, bandwidth overhead, and privacy imply that their encrypted network communications still leak information about the underlying connection [2]. This information, in the form of encrypted packet timings, sizes, and directions, is the basis of website-fingerprinting attacks.

In a website-fingerprinting attack, an observer attempts to identify a visited website from a network communication trace by comparing it to previously collected samples of known websites. Modern website-fingerprinting attacks can identify web-pages requested through Tor with accuracy and precision in excess of 90% [3–7].

Previous works on website fingerprinting, however, have all been limited to the use of TCP as the underlying network transport protocol. This choice was motivated by TCP being the dominant transport protocol in use, even as other technologies driving the Web at the application layer – browsers, scripting and styling languages, and web servers – rapidly evolved. The dominance of TCP is changing with the swift rise of the QUIC transport protocol, a connection-oriented protocol layered atop UDP, as an alternative transport protocol for delivering web traffic [8]. Driven by corporations such as Google [9], Akamai [10], and Cloudflare [11], QUIC is positioned to expel TCP as the de facto HTTP transport protocol. Indeed, HTTP/3, the next version of the core web protocol, will only support QUIC, and not TCP [12]. Although QUIC – first deployed in 2013 [9] – is a recent protocol, in some regions it already accounts for up to 10% of Internet traffic [13, 14]. This number is expected to further increase as other large content providers such as Facebook [15], Cloudflare [11] and Microsoft [16] continue to deploy support for QUIC.

QUIC's success is in no small part due to a host of new promising features, ranging from 0-RTT session resumption and protocol versioning, to stream-based multiplexing and variable in-band control information. *These new features, as well as the co-existence of QUIC and TCP traffic, warrant a rethinking of the current state of website fingerprinting.* Consequently, with the Internet's transition to QUIC, the following research questions arise: do classifiers trained in the TCP setting generalise to QUIC traces, is QUIC inherently more difficult to fingerprint than TCP, how does the importance

***Corresponding Author: Jean-Pierre Smith:** ETH Zurich, E-mail: jean-pierre.smith@inf.ethz.ch
Prateek Mittal: Princeton University, E-mail: pmittal@princeton.edu
Adrian Perrig: ETH Zurich, E-mail: adrian.perrig@inf.ethz.ch

of features change between the two protocols, and how can we jointly classify QUIC and TCP traces?

This paper makes the following contributions:

- We discuss challenges to website fingerprinting that arise with the shift in the HTTP landscape from TCP to QUIC, such as co-existence and dependence on TCP, embedded and variable-length control information, and protocol versioning (Section 3).
- We collect and evaluate the first publicly available[1] QUIC-TCP website-fingerprinting dataset, with around 117,000 QUIC and TCP web-page traces. Our closed-world dataset consists 20,000 traces across 200 TCP and QUIC visits for 100 URLs. Our open-world dataset adds an additional 97,000 traces across 16,000 URLs and both protocols (Section 4).
- We demonstrate that up to 96% of web-pages of interest that are requested via QUIC instead of TCP go undetected by state-of-the-art TCP-trained classifiers, such as k-FP [5], DF [7], Var-CNN [17], and p-FP(C) [18] (Section 6).
- We show that features manually engineered for TCP can be applied in the QUIC setting, but that care must be taken when transitioning since the features that are important may shift between settings (Section 6).
- Finally, we evaluate two approaches towards jointly classifying QUIC and TCP traces: a direct classification based approach (*Mixed*) and an ensemble-based approach (*Split*). We show that though classifying QUIC is not inherently more difficult than TCP, classifying both together is non-trivial given a constant training-set size (Section 7).

2 Background

In this section we introduce website fingerprinting, our threat model and assumptions, and the classifiers used throughout this paper.

2.1 Website Fingerprinting

In website fingerprinting, the goal of an observer is to identify a visited website over an encrypted channel, based on side-channel information such as those derived from packet sizes and timestamps [19]. The observer accomplishes this using a *classifier* [3, 5–7, 17–23]. The classifier, having been trained on samples of web-pages, receives a vector of features derived from the trace of a loading web-page and labels it as a previously observed web-page.

In the *closed-world setting*, the trace is known to be from one of n *monitored* web-pages, and the classifier must determine which. In the *open-world setting*, the trace belongs to either one of the n monitored web-pages or to any other *unmonitored* web-page. The classifier must therefore determine whether the trace is of a monitored web-page, and if so, which, or whether it is of some unmonitored web-page. This latter setting better emulates the real-world in which a sample may come from a previously unseen or unknown web-page.

2.2 Threat Model and Assumptions

We consider the open-world setting where a passive eavesdropper wishes to identify whether a client is visiting a monitored web-page, and if so which. The client, desiring to conceal this information, obscures trivially identifying information such as the HTTP requests, TLS server name indication extensions, and destination IP addresses by encrypting and proxying his communications. To do so, the client utilises a datagram-oriented encryption layer such as WPA2 on wireless LANs (IEEE 802.11i), or VPN services services such as Wireguard [24], OpenVPN, or IPSec in tunnel mode. The observer can therefore only witness the sizes and timings of the packets. We focus on datagram-oriented tunnelling approaches since QUIC running over TCP would exhibit a behaviour that is neither QUIC nor TCP, whereas our goal is to investigate the differences between these two protocols. Furthermore, Tor [1] does not widely support the QUIC protocol. We select Wireguard [24] as a representative VPN technology.

Current website-fingerprinting attacks implicitly utilise a number of simplifying assumptions related to the collection of the traces used in both the learning of the fingerprints and identification of the websites [6, 17, 18, 23, 25]: (1) an observer can identify the start and end of a web-page load, and can thus extract its trace from the overall traffic; (2) web-page loads are not interleaved, i.e., web-pages load sequentially, and the connection is free of background noise such as media or file transfers; and (3) the page of interest is the index page of the domain.

1 https://github.com/jpcsmith/wf-in-the-age-of-quic

We inherit these assumptions to remain consistent with current evaluation approaches, as we investigate the impact of QUIC on the current website fingerprinting setting. Additionally, previous works have shown these assumptions are not unfounded [26, 27]. Furthermore, we posit that the impact of the QUIC protocol would persist even for more complex fingerprinting settings.

2.3 State-of-the-Art Classifiers

Previous works have proposed numerous classifiers for use in the website fingerprinting setting. Below we introduce and motivate the classifiers used in this work. Section 10 describes other related work in the website-fingerprinting setting.

k-fingerprinting classifier (k-FP). The k-fingerprinting classifier of Hayes and Danezis [5] is composed of a random forest and nearest neighbour algorithm. It uses the random forest to extract an alternate feature set comprised of the leaf index at which the sample arrives in each tree in the forest. Using these alternate feature vectors, the classifier determines the k-nearest neighbours to the test sample as measured by their Hamming distances. If the classes associated with these neighbours from the training set are in agreement, then the test sample is given the same class; otherwise, it is associated with the unmonitored class. We extract the prediction probability of a class as the fraction of nearest neighbours for that class when $k = 6$ [7]. The k-FP classifier utilises features manually engineered from the traces which allows for more explainable classifications.

Deep Fingerprinting classifier (DF). The Deep Fingerprinting classifier proposed by Sirinam et al. [7] is a deep-learning classifier based on convolutional neural networks. It leverages techniques from computer vision to automatically extract features from the sequences of packet directions of each trace.

p-FP(C) classifier. The p-FP(C) classifier of Oh et al. [18] is another deep-learning classifier based on convolutional neural networks and sequences of packet directions. We select p-FP(C) over their multi-layer perceptron classifier, p-FP(M), as it performed better in the TLS setting [18] which is more analogous to our setting.

Var-CNN classifier. The Var-CNN classifier by Bhat et al. [17] is the final classifier utilised in our evaluation. It uses two instances of computer vision's state-of-the-art ResNets, trained on packet directions (Var-CNN$_S$) and packet timings (Var-CNN$_T$) in addition to manually extracted summary statistics. It is one of the only website-fingerprinting classifiers to utilise both automatically learned and manually extracted features [17].

3 A QUIC Disruption

QUIC is a new transport protocol that is increasingly replacing TCP as the main carrier of HTTP traffic, raising new challenges for creating website fingerprints. These challenges include co-existence and dependence on TCP, embedded, variable-length control information, and protocol versioning, and are described in this section.

3.1 QUIC

QUIC is a new connection-oriented transport protocol layered upon UDP. It was originally designed by Google with the intent of speeding up the Web [9], but is now undergoing standardisation in the Internet Engineering Task Force [8]. It will be the sole transport protocol of the next generation of the HTTP protocol, HTTP/3 [12]. QUIC and HTTP/3 – with the support of browser vendors such as Google, Apple, Microsoft, and Mozilla [16] as well as major content-delivery-network players such as Cloudflare [11] and Akamai [10] – are swiftly rising to displace HTTP/2 and TCP. Despite not yet having been fully standardised, QUIC is already being used extensively to deliver web content. Depending on the vantage point [13, 14], more than 2.6%–10% of the Internet's traffic is currently transported via QUIC.

QUIC's goals include establishing connections swiftly, minimising transport latency, providing multiplexing without head-of-line blocking, and providing an always-secure transport by mandating encryption [8].

3.2 New Challenges in Fingerprinting Websites

With the rise of the QUIC protocol in the HTTP landscape, we anticipate greater difficulty in the task of website fingerprinting. *The introduction of the QUIC protocol adds further variations to the traces for a given web-page. To combat this, we expect challenges in collecting and maintaining the associated datasets, as well as in the complexity of the classifier.*

Backwards compatibility with TCP. QUIC is a versioned protocol and endpoints are expected to revert to TCP in the event of version incompatibility. A website supporting QUIC must therefore continue to support TCP for the indefinite future [28]. This, coupled with practices such as "racing" QUIC and TCP connections to determine the better protocol for a given setting, means that web-pages may not be consistently loaded with only one protocol. *An observer must now accommodate the possibility of web-pages being loaded with either QUIC or TCP.*

Furthermore, loading a web-page involves requesting its constituent resources from potentially different servers. While a website and its associated web-pages may support QUIC, their required resources may not. In response, the browser loads the offending resources over TCP. Given the dynamic nature of web-pages and their required resources, this results in varying portions of the web-page at any given time being loaded by TCP instead of QUIC. *Therefore QUIC traces, such as those collected in Section 4, consist of varying mixtures of QUIC and TCP connections.*

Variable-length control packets. Packet payloads in QUIC carry frames: blocks of control information or application data. This allows embedding control information such as flow control or acknowledgements alongside application data as necessary. As the protocol reacts to changes in the network state, it embeds control information into the packets. In contrast to TCP where control messages occupy a fixed and consistent size in the header of each packet, control information in QUIC is embedded in frames in the packet when required. Some control frames, such as acknowledgement ranges, additionally consume a variable amount of bandwidth.

Differentiating between an encrypted packet carrying data, control information or a mixture of both, and thus the states of the connection, is therefore no longer as trivial as inspecting sequence and acknowledgement numbers.

Versioned protocol. QUIC is a versioned protocol, where each connection endpoint is expected to support multiple recent versions. The version that is used in the connection is agreed upon during the initial handshake. The use of versions, encryption of control and packet data, and a user-space implementation allow new QUIC versions to be rapidly developed and deployed.

Given the incremental changes between Google's QUIC versions [29], we presume that the behavioural differences that are present in the traces among subsequent versions will be minor. Versioning may, however, change the protocol's behaviour completely [28].

3.3 Other Notable Changes to the Transport Layer

The following changes add further variation to the collected traces, but may not substantially impact the traces for a web-page.

Multiplexed streams. Data in a QUIC connection is sent over streams. Streams provide a lightweight, ordered byte-stream abstraction to the application. Data for a stream is transmitted in stream frames within the encrypted payload. Each stream has its own stream identifier, offset and flow control, thereby allowing data to be concurrently sent on multiple streams.

The use of multiplexed streams in the QUIC protocol prevents losses in transmission of a single resource from delaying the entire transmission. Instead, such occurrences may result in different traces associated with the resources being requested, since the order of server responses is more reliant on network conditions.

Always-encrypted transport. Similar to TCP, QUIC leverages TLS to encrypt its communication. In contrast to TCP however, this encryption is mandatory and encompasses the entire QUIC packet, with the exception of a few flags and the connection identifier, which are visible for most data packets.

Connection identifiers. QUIC connections are associated with a pair of connection identifiers. Each identifier represents an endpoint of the connection and is chosen by that endpoint's peer. Additionally, new identifiers are available to an endpoint upon request. These identifiers allow a QUIC connection to survive changes to an endpoint's IP address or port.

0-RTT session resumption. QUIC endpoints that previously communicated are able to send encrypted data within their first packet. QUIC's handshake and TLS 1.3's session resumption [30] eliminate TCP's minimum of 1 round trip necessary to establish the connection, thereby hastening data transfers and web-page loads.

4 Combined QUIC-TCP Dataset

We utilise a combined QUIC-TCP dataset in investigating the impact of the transition from TCP to QUIC on

Table 1. Final distribution of traces within the dataset across the monitored and unmonitored sets. The gateway locations are New York (NY), Frankfurt (FRA) and Bengaluru (BLR).

	Monitored	Unmonitored
URLs / Classes	100	16,182
Traces per URL	200	6
... per protocol (QUIC, TCP)	100	3
... per gateway (NY, FRA, BLR)	∼66	2
... per protocol per gateway	∼33	1
Total traces	20,000	97,092

website fingerprinting. We therefore began by collecting such a dataset, as existing datasets only contain traces collected with TLS, Tor, or VPNs over TCP [4–7, 26].

The labelled QUIC-TCP dataset (summarised in Table 1) consists of 20,000 monitored traces divided into 100 web-pages with 100 traces per transport protocol; and a set of around 97,000 unmonitored traces with approximately 16,000 unique domains and 3 samples per protocol. Our inclusion of both QUIC and TCP traces in a single dataset enables direct comparison across both protocols for the same web-page.

We collected the dataset in two phases: identification of QUIC-enabled web-pages and the recording of their traces.

4.1 Identifying QUIC-Enabled Web-Pages

We identified QUIC-enabled web-pages by filtering them from a large set of popular domains using the HTTP alternative service header described below [31].

We first combined the Alexa Top 1M domain list used in previous works [6, 7] with the Cisco Umbrella top 1 million list of frequently queried DNS domains [32], and the Majestic Million list of the 1 million most referred domains [33]. We then filtered the list of duplicates and of invalid domains (e.g., reverse DNS domains) by removing domain names that do not end in a top-level domain [34]. The resulting list consisted of around 2.2 million domain names.

Next, we converted each domain name to a URL by pre-pending it with "https://"[2], and fetched a single instance of the web-page over HTTPS/TCP, using the Python aiohttp library [35]. For each fetch, we recorded the final redirected URL, HTTP status, and alternative service (`alt-svc`) response header where present. A web-page may provide an `alt-svc` header in its response to indicate alternative network locations where the web-page or resource can be retrieved [31]. This header informs the client that the web-page is available via HTTP/3 or HTTP/2 over QUIC [12], as well as which versions of QUIC are supported. We timed out each request after 30 seconds, and rate-limited them to avoid overwhelming the server.

Of the 2.2 million domains, around 0.99 million domains returned a successful HTTP response code in the 200-range, indicating a valid, HTTPS-enabled domain. We then filtered these domains by selecting only domains with alternative services containing either the string "h3" (HTTP/3) or "quic". Finally, we ensured a diverse set of domains by keeping only a single result where multiple pages redirected to the same final URL (ignoring any trailing slash or path in the URL); and by selecting a single URL among domains with the same private domain name but different public suffixes [36] (e.g. "google.com" and "google.de"). In this way, we identified a total of approximately 98,300 unique URLs which supported some version of the QUIC protocol. Of these URLs we selected 30,000 URLs that supported recent QUIC versions.

4.2 Collecting the Traces

Having identified a set of domains supporting the QUIC protocol, we then split our dataset into monitored and unmonitored web-pages and collected their traces.

We split the 30,000 URLs into a monitored set consisting of 300 URLs, and an unmonitored set consisting of the remaining URLs. Though interested in collecting traces from 100 monitored URL, we conservatively crawled 300 URLs to account for failures such varying QUIC support across our VPN gateway regions, timeouts, and IP blacklisting. We then sampled the 300 URLs uniformly at random from the remaining URLs. When sampling the monitored domains, we ensured that no private domain occurred more than 50 times in the dataset, for example, '*.blogspot.com' or '*.myshopify.com'. This latter step controls for biases where such domains, by virtue of their shared infrastructure, account for a disproportionate fraction of the QUIC-enabled domains. For each monitored URL we collected 120 TCP traces and 120 QUIC traces, to ensure our quota of 100 traces for each URL in the final

2 QUIC mandates the use of encryption, and so we assumed that web-pages not supporting HTTPS will also not support QUIC.

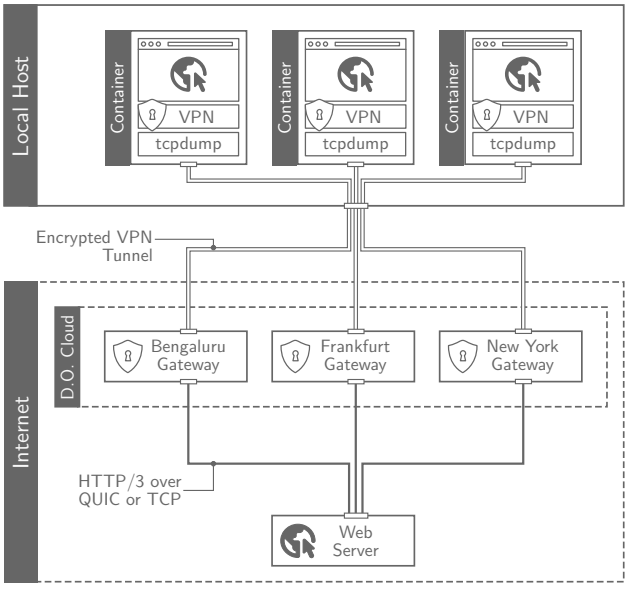


Fig. 1. The collection infrastructure spanning a host with multiple docker containers connected to three VPN gateways in the cloud and collecting a single web-page from a web-server.

dataset. For each unmonitored URL we collected 3 TCP traces and 3 QUIC traces.

We collected the dataset through three Wireguard VPN gateways deployed globally – in New York, U.S.A; Frankfurt, Germany; and Bengaluru, India – to ensure robustness of the dataset to the client's choice of VPN location (Figure 1). For each web-page in the monitored set, 40 TCP traces and 40 QUIC traces were collected via each VPN server. For each web-page in the unmonitored domain, 1 TCP and 1 QUIC trace were collected via each VPN server.

Each VPN server was deployed on a VM in the Digital Ocean cloud service and was provisioned with 1 GB of memory and 2 CPUs. We further utilised 15 simultaneous instances of the Chromium browser on a RedHat Linux virtual machine with thirty-two 2.7GHz processors and 20GB of RAM. We deployed each instance in a docker container to isolate the state, network, memory, and CPU consumption of each browser instance from any other. Five browser instances were assigned to each of the VPNs, and each browser instance performed a single web-page request then was closed and reopened to clear the session cache. Additionally, requests for each web-page collected over a single VPN server were spaced by at least 30 seconds to prevent the IP of the VPN server being blacklisted by the web-server. Further details on collection and trace sanitisation can be found in Appendix A.

5 Features and Methods

To investigate the potential impact of the QUIC protocol on the website fingerprinting landscape, we utilised a combination of machine learning classification and feature analysis. We describe the utilised features, our experimental methods, and the reported metrics in this section.

5.1 Features

We adapted the k-fingerprinting (k-FP) [5], Deep Fingerprinting (DF) [7], p-FP(C) [18], and Var-CNN [17] classifiers to the VPN and QUIC settings by addressing the inclusion of packet sizes, the presence of encrypted control packets, and the shift in feature importances.

We trained all classifiers on packet-size information, in addition to packet directions used in previous works [5, 7, 17, 18]. In contrast to the Tor-setting in which all packets are padded to fixed length, packet sizes are available in the VPN setting. We therefore provided the deep-learning classifiers, DF, p-FP(C), and Var-CNN, with the signed packet sizes instead of signed directions. Additionally, we trained the Var-CNN classifier with 5 additional metadata features based on packet sizes: incoming, outgoing, and total bytes, as well as the fraction of incoming and outgoing bytes. These features are analogous to those utilised by Var-CNN on the packet counts [17]. While only the p-FP(C) classifier had been evaluated with packet sizes, we anticipate that the additional information will not disadvantage the classifiers, and they may simply train to ignore them. Furthermore, we trained the k-FP classifier with the full set of timing, direction, and size features identified by Hayes and Danezis [5].

Moreover, we removed acknowledgement and control packets by filtering small packets since this increases classifier performance and is employed in the Tor and TLS fingerprinting settings [37, 38]. By contrast, removing control packets from a QUIC connection is non-trivial, as acknowledgements are variable in length and, like other control information, are encrypted. Furthermore, in the tunnelled VPN setting considered in this paper, TCP acknowledgements are also not clearly indicated. To remove such small packets we instead determined an appropriate packet-size threshold and removed all packets that are below this threshold, as explained in Section 8.

Finally, we utilised a QUIC-specific feature set for the k-FP classifier, as the shift in classification domain from TCP to QUIC may necessitate a change of the underlying features. While deep learning classifiers learn their own features and so should adapt to the new setting, the features engineered for k-FP were specific to the TCP setting. We therefore evaluated the applicability of these features to the QUIC setting in Section 6.2.

5.2 Open-World Experiments

We evaluated the classifiers in the more realistic open-world scenario. We describe the general procedure below, and identify in subsequent sections whenever we diverge from this procedure.

We trained and tested each classifier on the traces from our monitored and unmonitored datasets. The monitored set consisted of 100 domains with 100 (potentially mixed-protocol) samples as is common in the literature [3, 5, 26]. By contrast, the unmonitored set contains approximately 97,000 traces with around 16,000 URLs collected through the various VPN locations via both protocols.

For each experiment we performed at least 10 repetitions using 10-fold cross-validation. When splitting the monitored dataset, we ensured that each class (or class-protocol pair) had equal representation in the training and test sets, thereby reducing the variance of our measurements across the multiple splits. When splitting the unmonitored dataset, we ensured that no URL appears in both the training and validation set, as well as that no URL is common to the two protocols. This ensures that the unmonitored test set represents previously unseen URLs, and that recurring unmonitored URLs do not influence comparisons between QUIC and TCP. This latter constraint results in an unmonitored set of effectively 48,000 traces.

Where a classifier was trained on both protocol versions, we used a 1:1 ratio of QUIC and TCP traces. The transition to QUIC and HTTP/3 is ongoing and it is not currently possible to determine the resulting usage distribution of HTTP/1 and HTTP/2 over TCP, and HTTP/3 over QUIC. We therefore assumed a uniform distribution between these two protocol.

5.3 Metrics

Below we provide the definitions of the metrics used in our evaluations.

True-, wrong-, and false-positives. When a trace is labelled as a monitored web-page it is called a *positive*. We use the definitions of positives as presented by Wang [25], since they have shown them to be more representative of classifier performance [25]. A *true-positive* is a monitored trace that is correctly labelled. A *wrong-positive* is a monitored trace incorrectly labelled as a different monitored web-page. A *false-positive* occurs when a trace belonging to an unmonitored web-page is incorrectly labelled as a monitored web-page [25]. Their respective rates (TPR, WPR, and FPR) are calculated by dividing by the number of positive (TPR and WPR) and negative (FPR) labels respectively.

Recall. Recall is equivalent to the true-positive rate (TPR). It is the rate of labelling a monitored web-page as the correct monitored web-page.

r-precision (π_r). The *precision* or *sensitivity* of a classifier on a given dataset is the ratio of true-positives to other positives. This metric is inherently coupled to the number of positive and negative samples in the dataset, and requires knowledge of the dataset's distribution to adequately judge how the classifier would generalise in the real world. Wang proposed r-precision in order to make this distribution explicit [25]. r-precision is defined as:

$$\pi_r = \frac{\text{TPR}}{\text{TPR} + \text{WPR} + r \cdot \text{FPR}}$$

Here, r is the expected ratio of negative to positive visits in the real world. Throughout this paper, we use a ratio of $r = 20$ corresponding to a visit to a single monitored web-page for every 20 unmonitored web-pages, consistent with that of Wang [25].

6 From TCP to QUIC

In this section, we investigate whether classifiers trained in the TCP setting generalise to QUIC traces, how the importance of features changes between the two protocols, and whether QUIC is inherently more difficult to fingerprint than TCP.

6.1 Generalisation from TCP to QUIC

Generalisation, a core concept in machine learning, refers to the ability of a classifier to correctly classify previously unobserved samples. In website fingerprinting, this traditionally applies to the classification of traces observed in deployment, after having trained

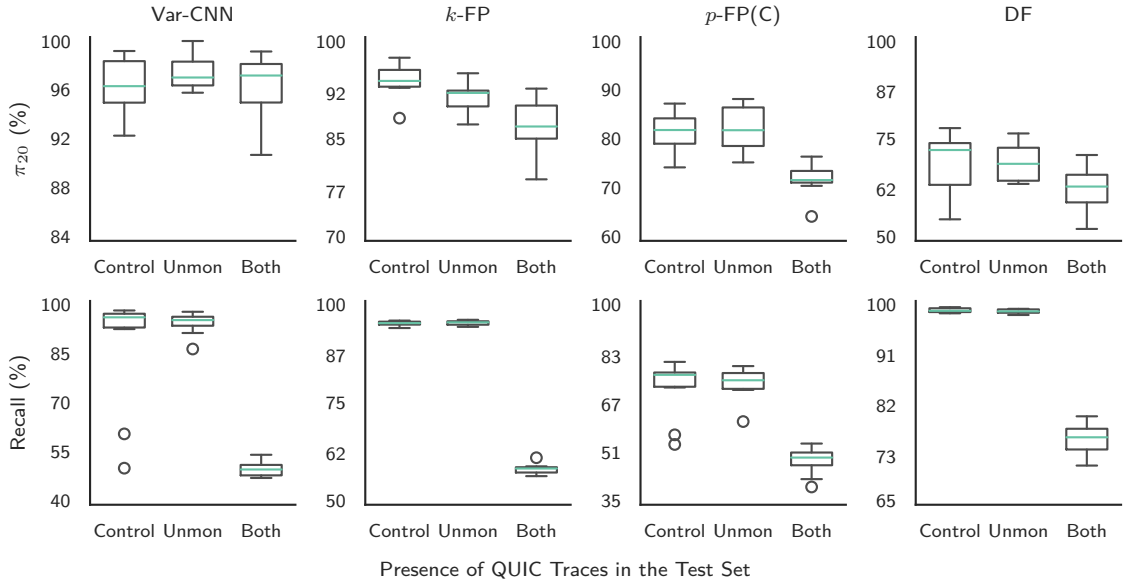


Fig. 2. Median r_{20}-precision and recall scores of the various classifiers when trained on TCP samples but tested in the settings where the client does not visit websites using QUIC and there exists QUIC traces in the open-world (Unmon.); and where the client may additionally visit web-pages using either QUIC or TCP (Both). The performance in the TCP-only setting is shown for comparison (Control).

on traces collected in the lab. In this setting, given the current approach of training classifiers for VPNs/TLS proxies in a manner agnostic of the tunnelled protocol, we investigate whether a classifier trained on tunnelled TCP traces is generalisable to settings involving the QUIC protocol.

TCP-trained classifiers are robust to unmonitored QUIC traces. First, we evaluated whether the classifier is able to maintain its predictive power in the presence of unmonitored QUIC traffic. We investigated this by evaluating the classifiers in two open-world settings, each with a dataset of 100×100 monitored traces and 48,500 unmonitored traces. In the control setting, we trained and tested the classifier on TCP traces only. In the experimental setting, we replaced half of the *unmonitored test* samples with unmonitored QUIC traces. Classifier hyper-parameters were set according to their respective papers.

Figure 2 shows the median r_{20}-precision and recall for these two settings as "Control" and "Unmon" respectively. Here, the largest changes in precision are a decrease of 3.5% for the DF classifier, and decrease of 1.8% for the k-FP classifier. The largest change in recall is a 1.8% decrease for the p-FP(C) classifier. These changes, however, fall generally within the interquartile-ranges of Control's results. We therefore conclude that *despite there being a change in protocol, the unmonitored QUIC traces are sufficiently different from the monitored set that they are still classified as unmonitored traces, with only a slight loss of precision.*

Up to 96% of monitored QUIC traces evade TCP-trained classifiers. Next, we investigated whether a client can evade detection of visits to a monitored page by requesting pages using QUIC.

We utilised the above setting, but with a 1:1 ratio of QUIC and TCP in both the *monitored and unmonitored* test sets. These results are also shown in Figure 2 as "Both". Here, we find larger decreases in the median r_{20}-precision than in the setting above – 10% decrease for p-FP(C) and 9.3% for DF. Furthermore, the changes in recall are significantly more pronounced.

We observe a steep reduction in the recall of the classifiers in comparison to the control setting, with a difference of 47%, 37%, 27%, and 23% for the Var-CNN, k-FP, p-FP(C), and DF classifiers respectively. This behaviour is confirmed for additional QUIC-TCP ratios in Appendix B.

The confusion matrix shown in Table 2 of the Var-CNN classifier provides an explanation for the severe reduction in recall. Of the 5,000 tested QUIC samples, 4,801 (96%) of them are incorrectly classified as unmonitored traces by the classifier. *Despite the traces being of the same page, they are distinct enough to result in client visits to the monitored URLs going undetected.*

Table 2. Confusion matrix for the Var-CNN classifier's predictions across all 10 repetitions, when trained on TCP and tested on monitored and unmonitored TCP and QUIC samples. Wrong-positives are treated as true-positives for simplicity of presentation.

Actual	Predicted		Misclassify (%)
	Monitored	Unmonitored	
Monitored			
QUIC	199	4,801	96.02
TCP	4,776	224	4.48
Unmonitored			
QUIC	7	24,293	0.03
TCP	40	24,230	0.16

6.2 Feature Comparison Between QUIC and TCP

In the previous section, we saw that a classifier trained on TCP is inaccurate at identifying QUIC traces. In this section, we determined whether TCP's manually engineered features are adequate for use with dedicated QUIC classifiers.

We measured the importance of each of the 3,043 manually engineered features collated by Li et al. [39], using the mean decrease in impurity (MDI) as determined by the random-forest underpinning the k-FP classifier. The mean-decrease in impurity identifies the degree to which a random-forest classifier has learned to use that feature to perform predictions, and has been used to identify important features for TCP [5, 37]. We trained the k-FP classifier on TCP- and QUIC-only datasets, using 3 repetitions of 10-fold cross-validation (30 repetitions total), and extracted the mean decrease in impurity across the trees in the forest. We then ranked the resulting MDIs, averaging ties, and computed the mean rank over the 30 repetitions.

Figure 3 illustrates the rank of each feature in the QUIC dataset plotted against the rank in the TCP dataset, for various groups of the features. We observe three sets of trends across the various semantic feature groups: groups that remain consistent across the protocols, that shift in importance towards one protocol or the other, and that shift the importance across protocols for features within the feature group.

Packet count importances generally remain consistent. The groups Packet Count and First-30 packet counts show little bias toward one protocol or the other. A classifier trained with these features should therefore have no issue utilising them regardless of whether the protocol is QUIC or TCP. The Last-30 packet counts however show more importance to TCP than QUIC.

Many features lose or gain in importance across protocols. The Time Statistics, Burst, First20, Last30 Packet Count, and Packets Per Second feature groups were ranked more important for the TCP classifier than the QUIC classifier. By contrast, the Transposition feature group shows a trend towards being more important for the QUIC classifier.

Features from incoming packets affected more than from outgoing packets. We further divided the Transposition and the Intervals I–III feature groups by their incoming and outgoing components. The Transposition group encodes the number of packets before the first 300 incoming and outgoing packets; whereas the Intervals feature groups are variations of the number of packets between each sequential pair of incoming or outgoing packets. Intervals III is created from Intervals II by aggregating a few of its features and exhibits a similar behaviour across the protocols. In these as well as the Transposition feature group, the incoming interval features show no bias towards either protocol, whereas the outgoing show a bias towards being important for QUIC. Contrarily, the incoming Intervals I group shows a slight bias towards TCP.

Importances shift within distribution features. The Packet Distribution and CUMUL feature groups belong to our final group are spread around the $x = y$ line, especially at the higher (less important) ranks. There are two main reasons for this. Firstly, the higher ranks are relatively unstable. Given the other features, the ranks greater than 1000 account for only around 5% of the importance (mean-decrease in impurity). Secondly, these features are computed over windows of each trace and shift with the change in protocol. For example, for the Burst feature group, which is delineated by incoming packets, the size of the third burst of outgoing packets was the number 1 ranked QUIC feature, whereas it was ranked 225 in TCP. By contrast, the fourth burst (one later) was ranked 88th in TCP but 301st in QUIC. QUIC's handshake establishes connections with 1-RTT less. Therefore a feature which would have appeared in the fourth burst in TCP occurs in the previous BURST in QUIC. Therefore, while features exist within these groups that are beneficial to each protocol, the exact feature may differ between the protocols.

Feature selection must consider both protocols. These results highlight that while the QUIC and HTTP/3 protocols are evolutions of the TCP- HTTP/2 stack, feature engineering for these protocols may take

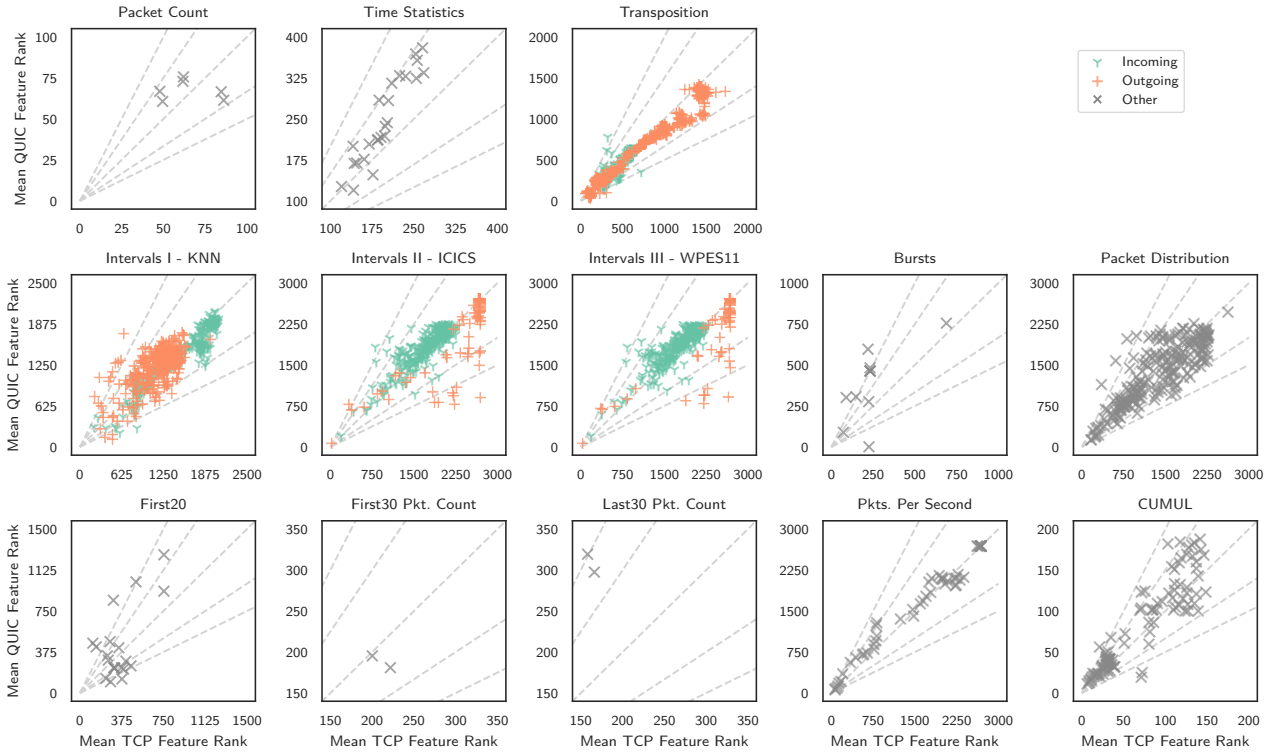


Fig. 3. Mean rank of features on the QUIC dataset versus the TCP dataset for the various semantic feature groups. A rank of 1 indicates the most important feature for that dataset. The dotted lines indicate regions where a rank is equal to, 1.5, or twice the other; that is, $\text{rank}(f_{i,\text{QUIC}}) = r \cdot \text{rank}(f_{i,\text{TCP}})$, for $r \in \{0.5, 0.667, 1, 1.5, 2\}$. See Appendix D for a description of the features.

different shapes. *Feature selection for QUIC and TCP must therefore consider both protocols simultaneously.* This is especially true when selecting features derived from windows of the trace, since the importance of those features may drastically change when going from one protocol to the next.

6.3 Is QUIC Harder to Fingerprint than TCP?

Next, we investigated whether the additional complexity in the QUIC protocol makes QUIC more difficult to fingerprinting than TCP. We select a subset of the features above for use with k-FP for classifying QUIC traces, and compare the classifiers' performances to the TCP-only setting.

6.3.1 Dedicated QUIC Features

Figure 4 shows the r_{20}-precision and recall of the k-FP classifier in the open-world QUIC setting, when trained and evaluated on the most highly ranked features. We

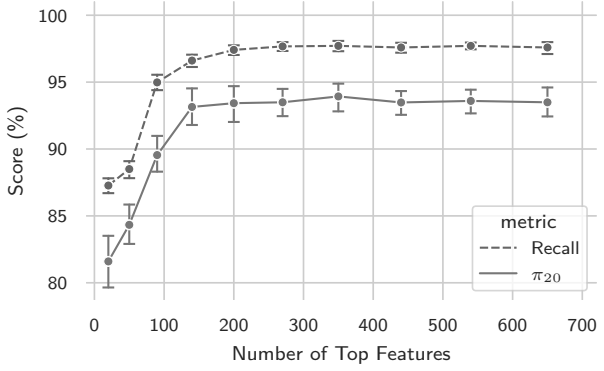


Fig. 4. Mean and 95% bootstrapped confidence intervals of the r_{20}-precision and recall scores for the k-FP classifier trained on varying numbers of the most highly ranked QUIC features in the open-world setting.

can see that similar to the TCP setting and the original k-FP feature-set, the top 200 features are sufficient to reach nearly peak precision and recall. The r_{20}-precision reaches its maximum at around 93% after around 150 features, whereas the recall achieves its maximum at around 97.5% after around 200 features. We therefore

conservatively use the top 300 features for further classification of the QUIC protocol when using k-FP.

6.3.2 Classifying QUIC Traces: No Harder than TCP

Figure 5 shows the median r_{20}-precision and recall of the classifiers in the open-world setting when trained and tested on either QUIC or TCP traces. We find negligible changes, both increases and decreases, in the r_{20}-precision for all the classifiers.

The largest absolute changes in precision, a 2.6% increase for the p-FP(C) classifier and a 1.2% decrease for the k-FP classifier when moving from the TCP to QUIC settings, are still well within the spread of the results. Classifier recall however, seems to increase in the QUIC setting. The k-FP classifier sees a 2.6% increase in recall. This, however, is likely due to the inclusion of features in the QUIC setting that were not originally considered for k-FP. The p-FP(C) classifier exhibits a 4.7% increase in median recall, and the Var-CNN classifier a 1.6% increase, with significantly less variability than the TCP setting. The DF classifier exhibits only a 0.15% increase in median recall.

The increases in recall hint towards more consistent features for the QUIC traces than TCP. Indeed given these results it is clear that *QUIC not fundamentally more difficult to fingerprint than TCP.*

7 Joint Classification of QUIC and TCP

Specialised classifiers are capable of adeptly classifying either QUIC or TCP traces. We now explore two approaches for *jointly* classifying QUIC and TCP traces.

7.1 The Mixed Classifier: Combining the Protocols

Since some web resources are not retrievable using QUIC, QUIC traces contain a mixture of QUIC and TCP connections. These TCP connections are therefore likely also present in the TCP-only traces. Modern classifiers are highly expressive and should be able to leverage these commonalities for an effective classification.

The classifier *Mixed* in Figure 5 shows the evaluation results for classifiers trained and tested on samples from both QUIC and TCP. In this setting, an observer would collect a training set with an equivalent number of QUIC and TCP traces for each URL. This accounts for the observer not knowing whether the targeted webpage will be downloaded via QUIC or TCP. Specifically, we trained the classifier on half-QUIC and half-TCP traces in the monitored set and half of the URLs in the unmonitored set contributed QUIC traces while the other half contributed TCP traces. Here, the k-FP classifier uses the union of the original k-FP features and those selected for QUIC in Section 6.2.

We find median decreases in r_{20}-precision of around 13.4% (DF), 4.1% (p-FP(C)), 3.5% (k-FP), and 0.91% (Var-CNN) when compared to the TCP setting for an equivalent training-set size. The Var-CNN suffered the least, but its precision is likely due to the utilisation of two-classifiers on the inter-arrival times and packet sizes of the trace, as well as its variable number of training epochs. Most classifiers showed only minor decreases in median recall ($\leq 0.60\%$), with the exception of the p-FP(C) classifier, which suffered around an 11% decrease in median recall.

Such a mixture of QUIC and TCP traces negatively affects precision more than recall, as more unmonitored pages are confused as monitored. At the same time, however, for an observer primarily interested in recall, this approach provides comparable recall to that of the TCP setting for most classifiers.

7.2 The Split Ensemble: Distinguishing the Protocol

The high precision and recall scores for the protocol-specific classifiers lend themselves to an ensemble-based approach for classification. Were an observer to determine to which protocol a trace belongs, the trace could be submitted to a protocol-specific classifier for identification. Indeed, while collecting the training set the observer is able to record the protocol used to fetch each instance of each web-page. Using this information, a classifier could be trained to distinguish between traces corresponding to each protocol, and the traces passed to their respective classifier.

7.2.1 Identifying QUIC Traces

In a QUIC trace, the main web-page is loaded via QUIC and subsequent resources are loaded via the same or additional QUIC or TCP connections. Therefore, classifying a trace as QUIC corresponds to determining whether

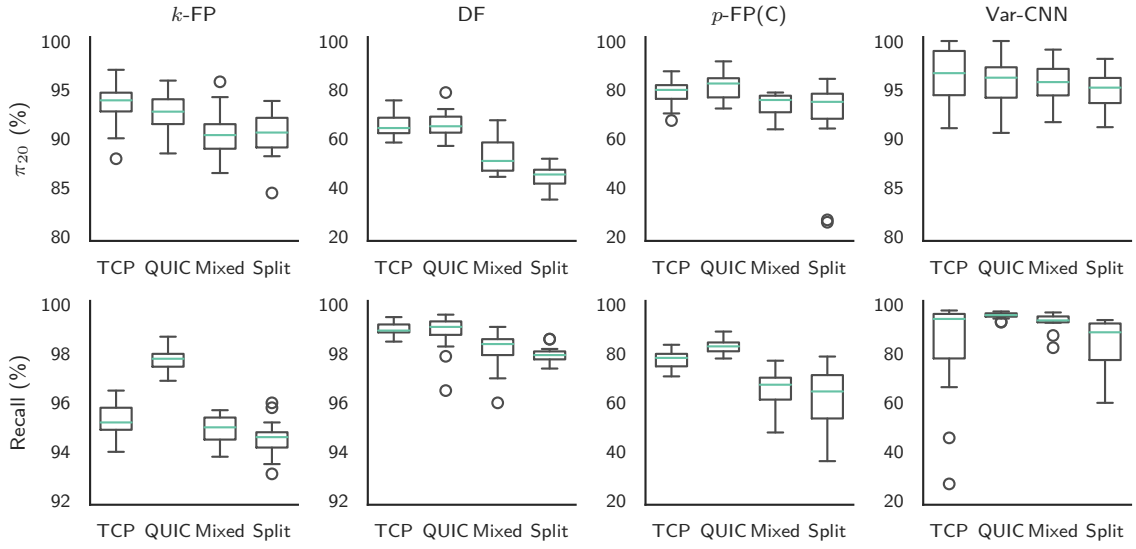


Fig. 5. Median r_{20}-precision and recall scores of the various classifiers when trained and tested on TCP samples (TCP), QUIC samples (QUIC), TCP and QUIC samples (Mixed), and TCP and QUIC samples along with the distinguisher (Split).

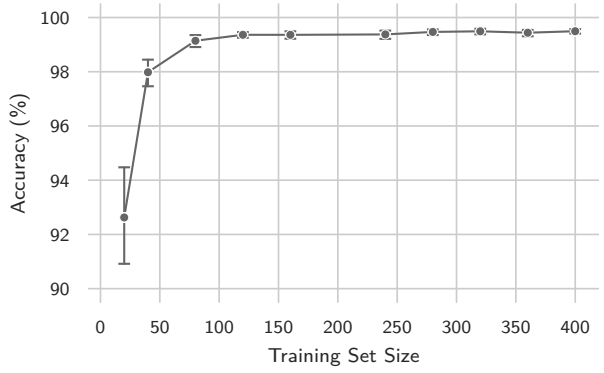
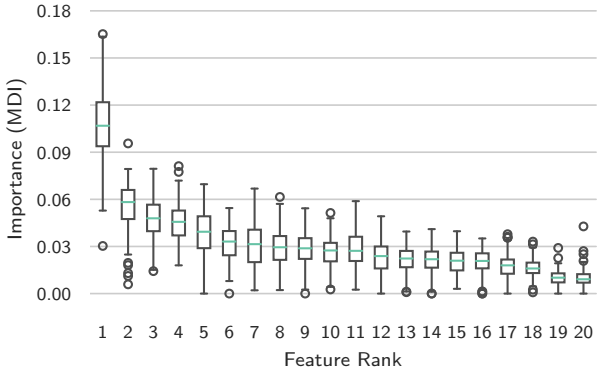


Fig. 6. Mean accuracy and bootstrapped 95% confidence intervals of the distinguisher in identifying traces that containing QUIC.

the trace contains the QUIC protocol. For this binary classification task, we utilised the random-forest classifier with the same features used for website classification. We anticipated that features useful for website classification may also be effective in distinguishing between the traces. We selected the random-forest classifier as it is compatible with the feature set and a highly effective classifier. We term the classifier tasked with distinguishing TCP from QUIC traces the *distinguisher*.

QUIC and TCP traces are easily distinguishable. We trained the distinguisher on two feature vectors per URL, one corresponding to a QUIC trace and the other to a TCP. These feature vectors were taken from both the monitored and unmonitored datasets. These samples were allocated to either the training set or the test set, in varying proportions, but no single URL was common to both the training and test sets. Figure 6 shows the mean accuracy of the distinguisher for a training set whose size increases from 20 to 400 samples. With only 100 samples (50 URLs), the distinguisher is able to differentiate between QUIC and TCP traces with an average accuracy over 99%. With 400 samples (200 URLs), it is able to distinguish QUIC and TCP traces with 99.5% accuracy, rising to 99.8% accuracy with 2560 samples (1280 URLs). *The distinguisher is therefore able to determine with high accuracy whether a trace contains QUIC or TCP connections, with only a small number of training samples.*

Handshakes identify QUIC and TCP traces. Figure 7 illustrates the feature importance scores (MDI) reported by the distinguisher, and gives insight into this performance. The most significant features were related to the outgoing packet sizes, with the most significant being the size of the 1st outgoing burst of packets. Clients using the QUIC protocol must pad their handshake packets to a minimum of 1,200 bytes [8]. A browser may also send multiple of these initial handshake packets to ensure connection establishment in the face of loss. This initial burst therefore starkly contrasts with the small SYN packets of TCP. With the exception of the maximum, variance, and standard deviations of the outgoing packet sizes, the majority of these top-ranked features focus on the initial round of packets and bursts, namely the first 20 packets and the first 3 bursts.



Rank	Feature Description
1	Size of the 1st outgoing burst of packets (Burst)
2	Size of the 10th packet (First20)
3–4	Var. and std. of the outgoing packet sizes
5–8	Sizes of the 7th, 9th, 17th, and 19th packets (First20)
9	# pkts. before the third outgoing packet (Transposition)
10–11	Sizes of the 14th and 18th packets (First20)
12	Maximum outgoing packet size
13–15	Sizes of the 20th, 11th, and 12th packets (First20)
16	Size of the third outgoing burst of packets (Burst)
17	Size of the 13th packet (First20)
18	# pkts. before the 5th outgoing packet (Transposition)
19	The 56th cumulative feature (CUMUL)
20	# pkts. before the 6th outgoing packet (Transposition)

Fig. 7. Relative importance of the features used to distinguish between QUIC and TCP using the mean decrease in impurity.

This indicates that the distinguisher is primarily utilising the difference in the handshakes at the beginning of the trace to distinguish between traces containing the two different protocols. Given the excellent accuracy, we retain this feature-set and classifier for distinguishing between QUIC and TCP.

7.2.2 Splitting the Difference

We can now attempt to split the set of test samples based on their protocol. The ensemble of classifiers works as follows. Given a training set of size n in equal proportions QUIC and TCP, we trained the QUIC classifier to identify the label associated with the $n/2$ QUIC samples, and the TCP classifier to identify the remaining $n/2$ TCP samples. The distinguisher is trained to identify the protocol with all n samples. To classify a trace, we first pass its features to the distinguisher and get a likelihood of the trace being QUIC or TCP. We then perform a weighted (by the confidence of the distinguisher) mean with the predictions of the QUIC and TCP classifiers' probabilities.

We evaluated the Split ensemble in the open-world setting and allowed it the same data-budget as the Mixed classifier. Maintaining a consistent data budget ensures a fair comparison to the Mixed classifier, as both classification approaches utilise the same number of training samples. Generally, we see a reduction in r_{20}-precision and recall across all classifiers when compared to the Mixed classification approach (Figure 5).

The scores achieved by the Split ensemble are not the averages of the classification performances of the QUIC and TCP classifiers, despite near 100% accuracy in distinguishing between traces containing the two protocols. *The reason for this is the reduction in the training set size.* While in total the number of traces collected and used to train the classifiers in the Split ensemble is the same as the Mixed classifier, each protocol-specific classifier in the ensemble only sees half of the total samples. Despite containing a different protocol, QUIC traces contain information that may be beneficial for classification of the TCP traces for the same URL and vice-versa. *The Split ensemble, by training distinct classifiers, is unable to leverage this shared information. Given a fixed budget for the provisioning of traces to train the classifier, the Mixed classifier therefore appears to be superior as it can make use of the commonalities between the protocols' traces.*

8 Removing Control Packets

Prior work has shown that the removal of acknowledgement and control packets improves classifier performance [37, 38]. We show that filtering based on packet sizes in the encrypted tunnel setting, where control information is also encrypted, is sufficient to improve classifier performance.

Over 73% of outgoing packets are smaller than 130-bytes. Figure 8 shows the distribution of packet sizes across the samples in the TCP dataset. Around 73% of all outgoing packets have a size less than 130 bytes, compared to 8.1% of incoming packets. This is consistent with the encapsulation added by Wireguard, where a TCP acknowledgement is embedded in an additional IP header (min. 20 bytes), UDP header (8 bytes), and its TCP packet header header (16 bytes). We found, therefore, that the minimum packet size hovers between 120–130 bytes for TCP traces, with few, smaller Wireguard control packets.

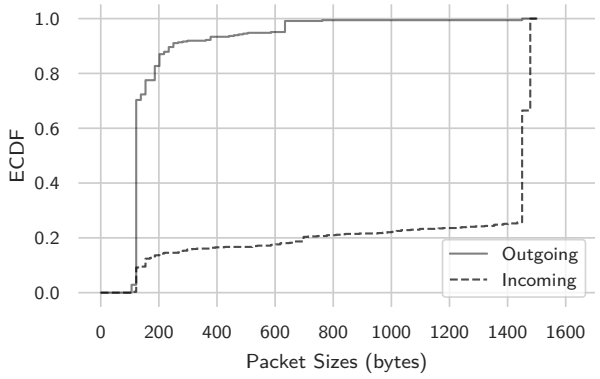


Fig. 8. Empirical cumulative distribution function for incoming and outgoing TCP-over-Wireguard packet sizes across all the traces.

Given the magnitude of these small packets, we anticipate that removing them should reduce the noise inherent in their presence. We evaluated each classifier after removing packets below 175 bytes and 130 bytes, and compared their performance to the use of the full trace. After removing the packets, we adjusted all timestamps to be relative to the new initial packet. We evaluated each classifier in the open-world setting with around 9,000 monitored TCP traces across 100 classes and 17,000 unmonitored TCP traces, using 10-fold cross-validation.

Filtering small packets generally improves precision. Table 3 shows the median r_{20}-precision and recall for the classifiers evaluated on the entire trace and with packets below 130 or 175 bytes removed. We see a consistent increase in the median precision for all of the classifiers with small packets removed, regardless of the threshold used. The DF classifier showed the highest increase in r_{20}-precision, with around a 12% and 11% increase in median precision, followed by k-FP (7.8%, 8.7%) and Var-CNN$_S$ (5.6%, 2.7%).

Filtering small packets improves recall for time-based classifiers. For recall however, we find more variegated results. While the size component Var-CNN (Var-CNN$_S$) shows almost no change in median recall, k-FP and the time component of Var-CNN (Var-CNN$_T$) show large increases in median recall. The k-FP classifier experienced a 2.3%–2.7% increase in median recall, whereas Var-CNN$_T$ enjoyed an 18%–31% increase in median recall, albeit with high variability. By comparison, the DF classifier suffers a minor decrease in median recall, around 0.5%, and the p-FP(C) classifier 3.1%–4.8% decrease in median recall. The larger benefit in recall for both the k-FP and Var-CNN$_T$ classifiers is likely due to their reliance on inter-arrival times, which is heavily distorted by the presence of control packets.

Given these results, and the importance of precision over recall in the website-fingerprinting setting [25], we selected 175 bytes as the filtering threshold for the traces in the earlier experiments.

9 Discussion

Traces requested via either protocol are similarly classifiable. However, differences in the protocols warrant attention when utilising them in conjunction with privacy-sensitive web-pages or when incorporating them into privacy-enhancing technologies.

Privacy-conscious websites take heed. Privacy-conscious websites should take heed in deploying support for the QUIC protocol. The hindrance in classifying QUIC traces with a TCP-only classifier (§6.1) and the potentially reduced performance in accounting for the additional protocol (§7) suggest that it would be beneficial to support QUIC. However, consider the setting where an observer is aware of QUIC and web-pages are accessed over a VPN. An observer that determines whether the web-page was requested with QUIC (§7.2) can use this information to reduce the set of possible web-pages to those that support the QUIC protocol, or to eliminate possible confusions with TCP-only domains within their monitored set.

Clearer connection boundaries. Initial QUIC packets sent by the client have a minimum size of 1200 bytes and may be sent in duplicate to mitigate loss. This is in contrast to TCP's small, acknowledgement-like SYN packets. Such a distinct pattern not only allows differentiating QUIC traces from TCP traces (§7.2), but may also aid in separating sequential traces or enumerating QUIC connections within a trace.

This problem as well as that of the identification of browsers based on the chosen initial packet sizes, may be mitigated by employing constant, MTU-sized handshake packets. Unfortunately, MTU probing occurs after the handshake and so committing to a smaller MTU would reduce the protocol's performance. Additionally, this large, minimum initial packet size implies that traffic analysis defences that resize packets may need to fragment and reassemble these packets if they are to be accepted by the endpoints.

QUIC and Tor. Tor does not support QUIC connections, but doing so would allow the next generation of

Table 3. Median r_{20}-precision and recall of the various classifiers when evaluated on the TCP dataset where no packets, packets below 130 bytes, and packets below 175 bytes have been removed. Var-CNN$_S$ and Var-CNN$_T$ correspond to the size and time components of the Var-CNN ensemble classifier.

	DF	k-FP	Var-CNN$_S$	Var-CNN$_T$	p-FP(C)
metric = π_{20} (%)					
None	37.1 (34.0–44.4)	69.8 (68.3–71.5)	75.8 (72.5–78.8)	59.6 (56.5–64.1)	55.7 (54.3–58.3)
130	49.2 (47.7–52.8)	77.6 (76.8–82.6)	81.5 (78.7–83.0)	62.8 (50.2–75.1)	58.4 (49.9–68.3)
175	48.6 (48.0–55.0)	78.5 (75.7–82.5)	78.5 (77.0–80.5)	61.8 (58.7–73.0)	55.8 (51.7–65.6)
metric = Recall (%)					
None	99.4 (99.3–99.4)	94.6 (93.5–94.9)	98.5 (98.2–98.8)	53.3 (43.0–68.3)	89.4 (87.0–91.6)
130	99.0 (98.8–99.3)	96.9 (96.7–97.0)	98.4 (98.2–98.8)	84.1 (59.9–86.5)	84.6 (73.7–89.4)
175	98.9 (98.7–99.1)	97.2 (97.0–97.7)	98.4 (98.2–98.8)	71.0 (63.4–77.9)	86.3 (78.8–89.9)

Internet applications to access its privacy benefits. Perhaps the most direct way to achieve this would be to support UDP proxying. Unfortunately, though a desire of the Tor Project, supporting UDP or general IP traffic brings a host of challenges [40] and diverges from Tor's established TCP security model.

In Tor, TCP connections are established between neighbouring nodes on an overlay path (Tor circuit) in the Internet. Only the data from the user's TCP connection traverses the circuit – control messages do not. Supporting QUIC via UDP proxying would result in QUIC's handshakes, acknowledgements, and other control messages traversing the circuit. This may allow an observer to identify the underlying protocol (§7.2) or fingerprint the user's QUIC implementation, and would need to be mitigated. Furthermore, it is unclear if QUIC's performance benefits can be realised atop Tor's reliable TCP-based design.

Another avenue for Tor would be to utilise QUIC connections within the network itself [41–43]. Here, QUIC's features such as stream multiplexing and authenticated control messages can help to reduce congestion [41] and interference among circuits [42], and prevent attacks that leverage TCP's unencrypted control packets, such as TCP reset attacks [8, 44]. These changes, whether replacing the TCP connections between nodes with QUIC connections [42, 43] or establishing Tor-client-to-exit QUIC connections [41], constitute a shift in the security model of Tor. Mathewson and Perry [44] have outlined how this may impact privacy in Tor but further security analyses are necessary.

9.1 Limitations

We acknowledge the following limitations of our work. First, we collected web-pages that supported both QUIC and TCP. This enabled us to evaluate settings in which the classifier is trained on samples from both protocols. Any biases in the distribution of web-pages that support QUIC and TLS/TCP, such as the exclusion of all HTTP-only websites, are therefore also present in our dataset. HTTPS promotion efforts have led to over 75% of Chrome web-page loads being performed by HTTPS [45, 46], but there may be websites, such as private or hobby sites, that have not undertaken the transition. We note however that similar biases towards popularity are present in the Alexa Top 1m domain list employed in previous works [6, 7].

Furthermore, despite our dataset consisting of around 117,000 traces, similar to recent works collecting 98,000 [18] and 135,000 [7] traces, the inclusion of a second transport protocol and multiple VPN vantage points in our dataset reduced our effective training set size. Researchers, however, still rely upon datasets of 4,000–20,000 [18, 22, 25, 27, 47] traces to train and evaluate their classifiers, which are smaller than a protocol-based partition of our dataset; and we were able to achieve r_{20}-precision and recall scores of over 90% on these undefended traces (§5). Nevertheless, advanced artificial neural networks and traffic analysis defence evaluations may benefit from a larger dataset.

10 Related Work

Cheng and Avnur [48], Sun et al. [49] and Hintz [50] were among the first to show that it was possible to identify a visited website using only the inferred sizes of the HTML page and resources. Bissias et al. [19] introduced the more challenging setting of classifying traces based on only packet sizes, directions, and timings as are available in the VPN and Tor settings. Since then, website

fingerprinting attacks have grown in both complexity and power. They utilise algorithms such as the nearest neighbour search [3], random forest classifiers [5], hidden Markov models [20], stream algorithms [21, 22], and neural networks [6, 7, 17, 18, 23] to achieve identification rates of over 95% on encrypted, padded network traffic; whereas others have sought to boost classifier performance with detailed feature analyses [5, 37, 39, 51].

Numerous website fingerprinting defences have been proposed [3, 47, 52–60]. Among these, fixed-rate and padded defences such as BuFLO [56], CS-BuFLO [55], and Tamaraw [61] offer among the highest levels of privacy albeit with high overheads. Other defences, such as Glove [54], Supersequence [3], Walkie-Talkie [53], FRONT, and GLUE [47] selectively apply cover traffic, padding, and delays to reduce the uniqueness of fingerprints while avoiding high overheads.

Another branch of the website fingerprinting literature has sought to verify the assumptions underpinning the attacks. These works have developed algorithms for splitting or otherwise classifying sequential and overlapping traces [21, 26, 27, 47, 62]. They have also addressed the scalability of training and maintaining classifiers, with models that may be incrementally updated [22] or can transfer their learning to the classification of previously unseen URLs using only a few samples [23].

More generally, Sy et al. [63] evaluated the QUIC protocol header for potential privacy leaks; and Govil et al. [64] utilised QUIC's connection migration feature to improve user privacy by continuously changing the client's IP address mid-connection. However, despite the varied literature, none of these works have investigated the impact of the introduction of the QUIC protocol to the HTTP landscape on website fingerprinting.

11 Conclusions

The introduction of QUIC is complicating the task of fingerprinting tunnelled web traffic.

Website fingerprinting in the age of QUIC. While classification of QUIC traces is not inherently more difficult than that of TCP traces, website fingerprinting classifiers trained on tunnelled TCP traces do not generalise well when monitored URLs may be visited using the QUIC protocol. Indeed, such failure to account for the possibility of the web-page being requested via the QUIC protocol can result in up to 96% of the QUIC traces for monitored URLs evading such a classifier. However, if the observer knows that the monitored URLs do not support the QUIC protocol, then using such a classifier has little to no penalty in the presence of unmonitored QUIC traces.

Classification of both QUIC and TCP traces together is possible, but comes at a cost. We explored two approaches to performing such classification and found that neither performs equivalent to TCP-only classifiers with the same data-budget. Training the classifier on samples from both QUIC and TCP leverages patterns common to both the QUIC and TCP traces for a given URL. The ensemble approach, by contrast, utilises the high performance of the protocol-specialised classifiers by first determining to which protocol a trace belongs. While we observed high accuracy in determining the protocol to be associated with a trace, the reduction in the size of the training set for each classifier results in an overall reduction of performance.

Changes for the "New Age". We encourage future works in website fingerprinting to consider and account for the impact of the QUIC protocol on their attack, defence, or feasibility analysis. We provide access to our combined QUIC-TCP dataset for this.

While manually engineered features for TCP are also usable in the QUIC setting, it is important to consider both protocols while performing feature selection. Alternatively, the added complexity of performing feature engineering and selection for both protocols can be avoided by using artificial-neural-network based classifiers. These classifiers have already shown great performance in the website-fingerprinting setting, and their automatic feature creation avoids the feature engineering overhead of the additional protocol.

Furthermore, the addition of a new protocol to the ossified HTTP stack presents an opportunity to enhance user-privacy. The presence of such a critical protocol in user-space facilitates privacy-conscious protocol implementations. Future work may explore how such implementations could distort the fingerprint of network traffic while remaining compatible with the QUIC standard.

Finally, in this work we investigated the addition of a new protocol to the transport layer. Reliable transport protocols, such as QUIC and TCP, are primarily composed of their congestion control and loss recovery components. Therefore other changes in the transport layer may drastically impact the resulting fingerprints. For example, the BBR congestion-control algorithm [65] is a potential congestion-control game-changer that departs from the traditional loss based approach of previous congestion-control algorithms. Like QUIC there may be benefits to accounting for its presence.

Acknowledgements

We gratefully acknowledge support from ETH Zurich, and from the Zurich Information Security and Privacy Centre. This work was supported in part by the National Science Foundation under grants CNS-1553437 and CNS-1704105, and by the United States Air Force and DARPA under Contract No. FA8750-19-C-0079. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force, DARPA, or any other sponsoring agency.

References

[1] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, page 21, USA, 2004. USENIX Association.

[2] D. Das, S. Meiser, E. Mohammadi, and A. Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 108–126, May 2018. 10.1109/SP.2018.00011.

[3] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, San Diego, CA, August 2014. USENIX Association. ISBN 978-1-931971-15-7. URL https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao.

[4] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016. URL https://nymity.ch/tor-dns/pdf/Panchenko2016a.pdf.

[5] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1187–1203, Austin, TX, August 2016. USENIX Association. ISBN 978-1-931971-32-4. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/hayes.

[6] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, 2018. 10.14722/ndss.2018.23105.

[7] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep Fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1928–1943, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356930. 10.1145/3243734.3243768.

[8] Jana Iyengar and Martin Thomson. QUIC: A UDP-based multiplexed and secure transport. Internet-Draft draft-ietf-quic-transport-27, Internet Engineering Task Force, February 2020. URL https://tools.ietf.org/html/draft-ietf-quic-transport-27.

[9] Jim Roskind. Experimenting with QUIC, 2013. URL https://blog.chromium.org/2013/06/experimenting-with-quic.html.

[10] Medhat Yakan and Akhil Jayaprakash. Introducing QUIC for web content, October 2018.

[11] Alessandro Ghedini and Rustam Lalkaka. HTTP/3: the past, the present, and the future, September 2019. URL https://blog.cloudflare.com/http3-the-past-present-and-future.

[12] Mike Bishop. Hypertext transfer protocol version 3 (HTTP/3). Internet-Draft draft-ietf-quic-http-27, Internet Engineering Task Force, February 2020. URL https://tools.ietf.org/html/draft-ietf-quic-http-27. http://www.ietf.org/internet-drafts/draft-ietf-quic-http-27.txt.

[13] Jan Rüth, Ingmar Poese, Christoph Dietzel, and Oliver Hohlfeld. A first look at QUIC in the wild. In Robert Beverly, Georgios Smaragdakis, and Anja Feldmann, editors, *Passive and Active Measurement*, pages 255–268, Cham, 2018. Springer International Publishing. ISBN 978-3-319-76481-8. 10.1007/978-3-319-76481-8_19.

[14] Mirja Kühlewind. Some updates on QUIC deployment numbers. In *IETF 106 Proceedings*, November 2019. URL https://trac.ietf.org/trac/irtf/wiki/map.

[15] Facebook. mvfst, 2019. URL https://github.com/facebookincubator/mvfst.

[16] IETF QUIC Working Group. [QUIC] implementations, 2019. URL https://github.com/quicwg/base-drafts/wiki/Implementations.

[17] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. Var-CNN: A data-efficient website fingerprinting attack based on deep learning. *Proceedings on Privacy Enhancing Technologies*, 2019(4):292–310, 2019. 10.2478/popets-2019-0070.

[18] Se Eun Oh, Saikrishna Sunkam, and Nicholas Hopper. p1-FP: Extraction, classification, and prediction of website fingerprints with deep learning. *Proceedings on Privacy Enhancing Technologies*, 2019(3):191–209, July 2019. 10.2478/popets-2019-0043.

[19] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted HTTP streams. In George Danezis and David Martin, editors, *Privacy Enhancing Technologies*, pages 1–11, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-34746-0. 10.1007/11767831_1.

[20] Z. Zhuo, Y. Zhang, Z. Zhang, X. Zhang, and J. Zhang. Website fingerprinting attack on anonymity networks based on profile hidden Markov model. *IEEE Transactions on Information Forensics and Security*, 13(5):1081–1095, May 2018. ISSN 1556-6021. 10.1109/TIFS.2017.2762825.

[21] Abdullah Qasem, Sami Zhioua, and Karima Makhlouf. Finding a needle in a haystack: The traffic analysis version. *Proceedings on Privacy Enhancing Technologies*, 2019(2):270–290, 2019. URL https://content.sciendo.com/view/journals/popets/2019/2/article-p270.xml.

[22] Reyhane Attarian, Lida Abdi, and Sattar Hashemi. AdaWFPA: Adaptive online website fingerprinting attack for tor anonymous network: A stream-wise paradigm. *Computer Communications*, 148:74–85, December 2019. ISSN 0140-3664. 10.1016/j.comcom.2019.09.008.

[23] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. Triplet Fingerprinting: More practical and portable website fingerprinting with N-shot learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pages 1131–1148, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. 10.1145/3319535.3354217.

[24] Jason A. Donenfeld. Wireguard, July 2020. URL https://www.wireguard.com.

[25] T. Wang. High precision open-world website fingerprinting. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 231–246, Los Alamitos, CA, USA, May 2020. IEEE Computer Society. 10.1109/SP.2020.00015.

[26] Tao Wang and Ian Goldberg. On realistically attacking Tor with website fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016(4):21–36, October 2016. 10.1515/popets-2016-0027.

[27] Weiqi Cui, Tao Chen, Christian Fields, Julianna Chen, Anthony Sierra, and Eric Chan-Tin. Revisiting assumptions for website fingerprinting attacks. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, pages 328–339, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367523. 10.1145/3321705.3329802.

[28] Mirja Kühlewind and Brian Trammell. Applicability of the QUIC transport protocol. Internet-Draft draft-ietf-quic-applicability-07, Internet Engineering Task Force, 2020. URL https://datatracker.ietf.org/doc/html/draft-ietf-quic-applicability-07.

[29] The QUICHE Team. Chrome QUIC protocol versions, 2019.

[30] E. Rescorla. The transport layer security (TLS) protocol version 1.3. RFC 8446, Internet Engineering Task Force, August 2018.

[31] M. Nottingham, P. McManus, and J. Reschke. HTTP alternative services. RFC 7838, Internet Engineering Task Force, April 2016.

[32] Cisco Umbrella. Umbrella popularity list - top 1 million, June 2020. URL http://s3-us-west-1.amazonaws.com/umbrella-static/index.html.

[33] Majestic. The majestic million, June 2020. URL https://majestic.com/reports/majestic-million.

[34] Internet Corporation for Assigned Names and Numbers (ICANN). List of top-level domains, June 2020. URL https://www.icann.org/resources/pages/tlds-2012-02-25-en.

[35] Nikolay Kim. aiohttp: Async http client/server framework (v3.6.2), October 2019. URL https://github.com/aio-libs/aiohttp.

[36] Mozilla Foundation. Public suffix list, June 2020. URL https://publicsuffix.org.

[37] Junhua Yan and Jasleen Kaur. Feature selection for website fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2018(4):200–219, 2018. URL https://content.sciendo.com/view/journals/popets/2018/4/article-p200.xml.

[38] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 263–274, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329576. 10.1145/2660267.2660368.

[39] Shuai Li, Huajun Guo, and Nicholas Hopper. Measuring information leakage in website fingerprinting attacks and defenses. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1977–1992, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356930. 10.1145/3243734.3243832.

[40] The Tor Project. Tor FAQ. Online, 2019. URL https://2019.www.torproject.org/docs/faq.html.en#TransportIPnotTCP.

[41] Mike Perry. The case for Tor-over-QUIC. Tor developer mailing list, March 2018. URL https://lists.torproject.org/pipermail/tor-dev/2018-March/013026.html.

[42] W. F. Sabée. Adding QUIC support to the Tor network. Master's thesis, Delft University of Technology, 2019.

[43] L. Basyoni, A. Erbad, M. Alsabah, N. Fetais, and M. Guizani. Empirical performance evaluation of quic protocol for tor anonymity network. In *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 635–642, June 2019. 10.1109/IWCMC.2019.8766609.

[44] Nick Mathewson and Mike Perry. Towards side channel analysis of datagram Tor vs current Tor. Technical Report 2018-11-002, The Tor Project, November 2018. URL https://research.torproject.org/techreports/side-channel-analysis-2018-11-27.pdf.

[45] Google. HTTPS encryption on the web, December 2020. URL https://transparencyreport.google.com/https/overview.

[46] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. Measuring HTTPS adoption on the web. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1323–1338, Vancouver, BC, August 2017. USENIX Association. ISBN 978-1-931971-40-9. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/felt.

[47] Jiajun Gong and Tao Wang. Zero-delay lightweight defenses against website fingerprinting. In *29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, August 2020. USENIX Association. URL https://www.usenix.org/conference/usenixsecurity20/presentation/gong.

[48] Heyning Cheng and Ron Avnur. Traffic analysis of SSL encrypted web browsing. Technical report, University of California, Berkeley, 1998.

[49] Qixiang Sun, D. R. Simon, Yi-Min Wang, W. Russell, V. N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings 2002 IEEE Symposium on Security and Privacy*. IEEE Comput. Soc, 2002. 10.1109/secpri.2002.1004359.

[50] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies*, pages 171–178. Springer Berlin Heidelberg, 2003. 10.1007/3-540-36467-6_13.

[51] Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. Tik-Tok: The utility of packet timing in website fingerprinting attacks.

Proceedings on Privacy Enhancing Technologies, 2020(3): 5–24, 2020. 10.2478/popets-2020-0043.

[52] W. Cui, J. Yu, Y. Gong, and E. Chan-Tin. Realistic cover traffic to mitigate website fingerprinting attacks. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1579–1584, July 2018. 10.1109/ICDCS.2018.00175.

[53] Tao Wang and Ian Goldberg. Walkie-Talkie: An efficient defense against passive website fingerprinting attacks. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1375–1390, Vancouver, BC, August 2017. USENIX Association. ISBN 978-1-931971-40-9. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao.

[54] Rishab Nithyanand, Xiang Cai, and Rob Johnson. Glove: A bespoke website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 131–134, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331487. 10.1145/2665943.2665950.

[55] Xiang Cai, Rishab Nithyanand, and Rob Johnson. CS-BuFLO: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 121–130, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331487. 10.1145/2665943.2665949.

[56] 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 *2012 IEEE Symposium on Security and Privacy*, pages 332–346, May 2012. 10.1109/SP.2012.28.

[57] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In Ioannis Askoxylakis, Sotiris Ioannidis, Sokratis Katsikas, and Catherine Meadows, editors, *Computer Security – ESORICS 2016*, pages 27–46, Cham, 2016. Springer International Publishing. ISBN 978-3-319-45744-4. 10.1007/978-3-319-45744-4_2.

[58] Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. HTTPOS: sealing information leaks with browser-side obfuscation of encrypted flows. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society, 2011. URL https://www.ndss-symposium.org/ndss2011/httpos-sealing-information-leaks-with-browser-side-obfuscation-of-encrypted-flows.

[59] Charles V. Wright, Scott E. Coull, and Fabian Monrose. Traffic Morphing: An efficient defense against statistical traffic analysis. In *NDSS*, volume 9. Citeseer, 2009.

[60] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society - WPES '11*. ACM Press, 2011. 10.1145/2046556.2046570.

[61] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian 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. Association for Computing Machinery. ISBN 9781450329576. 10.1145/2660267.2660362.

[62] Yixiao Xu, Tao Wang, Qi Li, Qingyuan Gong, Yang Chen, and Yong Jiang. A multi-tab website fingerprinting attack. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 327–341, New York, New York, USA, December 2018. ACM. ISBN 9781450365697. 10.1145/3274694.3274697.

[63] Erik Sy, Christian Burkert, Hannes Federrath, and Mathias Fischer. A QUIC look at web tracking. *Proceedings on Privacy Enhancing Technologies*, 2019(3):255–266, 2019. 10.2478/popets-2019-0046.

[64] Yashodhar Govil, Liang Wang, and Jennifer Rexford. MIMIQ: Masking IPs with migration in QUIC. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association, August 2020. URL https://www.usenix.org/conference/foci20/presentation/govil.

[65] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: Congestion-based congestion control. *ACM Queue*, 14, September-October:20–53, 2016. URL http://queue.acm.org/detail.cfm?id=3022184.

[66] Google. Chromium (commit 756066), April 2020. URL https://commondatastorage.googleapis.com/chromium-browser-snapshots/index.html?prefix=Linux_x64/756066.

[67] Yi Shi and Kanta Matsuura. Fingerprinting attack on the Tor anonymity system. In Sihan Qing, Chris J. Mitchell, and Guilin Wang, editors, *Information and Communications Security*, pages 425–438. Springer, Berlin, Heidelberg, Berlin, Heidelberg, December 2009. ISBN 978-3-642-11145-7. 10.1007/978-3-642-11145-7_33.

A Trace Collection, Sanitisation, and Composition

The collection of the traces involved numerous technologies and procedures for ensuring a clean dataset. We describe these methods, the resulting composition of traces, and invalid samples that evaded our measures.

Trace collection. We fetched each web-page using the Chromium web browser, controlled via the Selenium browser automation framework. The Chromium browser underpins both the Google Chrome and Microsoft Edge browser, and was the first browser to support QUIC. We utilised the Chromium browser based on revision 756066 from 2020-04-02 [66], which roughly corresponds to Google Chrome version 83. This allowed us to maintain a stable and reproducible experiment platform, as outdated versions of Chrome are not provided by Google. Each fetch, by an instance of the browser, was performed in a Docker container and therefore iso-

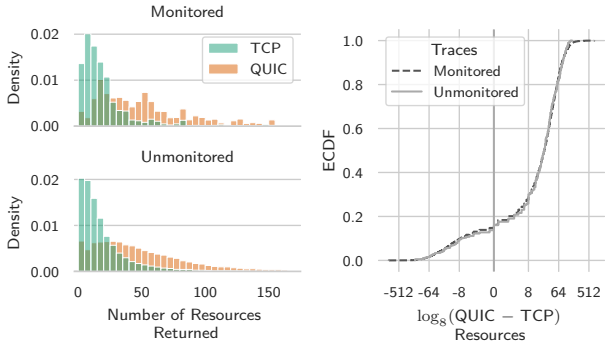


Fig. 9. Distribution of the number of resources returned over TCP or QUIC for QUIC traces in the dataset (left). The ECDF plot (right) shows the fraction of traces that had x or less QUIC resources more than TCP resources.

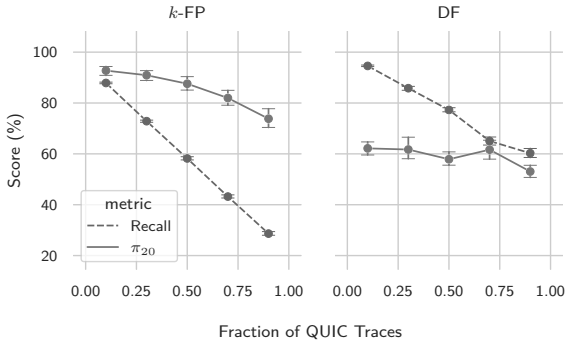


Fig. 10. Mean r_{20}-precision and recall scores and their 95% confidence intervals for classifiers trained on TCP samples but tested in the settings where the client may visit the web-page using QUIC or TCP.

lated from the network traffic of the host machine and other browser instances. Within the container, we captured all network traffic to and from the isolated network stack using the tcpdump utility and filtered it to packets belonging to the Wireguard protocol.

Dataset sanitisation. Additionally, we performed a number of procedures to ensure a clean dataset. First, for each TCP request we disabled the QUIC protocol, and for each QUIC request we forced the browser to only open the initial connection using QUIC. This prevented the browser from attempting to optimise the request by potentially switching to QUIC from TCP or vice-versa. Second, failed requests were detected using a combination of time-outs, connection errors, empty HTML pages, and performance logs provided by the browser. Any HTTP response codes present in the performance logs with a value greater than or equal to 400 for the main domain represented a failure by the server or the client and was removed. Third, any trace with only incoming or outgoing packets were removed. Finally, we filtered the QUIC and TCP domains to those common across both protocols and all versions.

Unfortunately, despite our best efforts, we subsequently noticed that the QUIC traces belonging to 1 URL in the monitored set, and 63 (of 17,000) URLs in the unmonitored set erroneously contained solely TCP connections. We note however that the presence of these traces would bias our results towards finding less of a difference between QUIC and TCP (such as found in Section 6), and therefore strengthen these results.

Trace composition. As mentioned in Section 3, requesting a web-page using QUIC may involve loading resources from additional servers using TCP. We therefore investigated the composition of the collected traces.

Each monitored trace loaded a median of around 43 resources with QUIC, but only around 14 resources with TCP, and each unmonitored trace 40 and 12 resources respectively (Figure 9). Furthermore, the right-hand of Figure 9 shows that 84% of the traces in the monitored and unmonitored datasets loaded more resources via QUIC than TCP, and around 17% loaded 64 or more resources via QUIC. In total, over twice as many resources were loaded with QUIC than with TCP. Hence, despite QUIC traces containing TCP connections, the majority of the resources requested during the loading of web-pages are returned via QUIC.

B Varying the QUIC Ratio in the Generalisability Experiment

We evaluated whether the observed reduction in classifier performance, in the setting where a client attempts to evade the detection of monitored web-page visit by requesting the web-page using QUIC (Section 6.1), is consistent for various ratios of QUIC to TCP traces. We repeated the experiment for one deep-learning and one non-deep-learning classifier, DF and k-FP respectively, while varying the fraction of traces that support QUIC for each URL.

Figure 10 shows the results of this experiment. As the presence of QUIC traces increase in the monitored and unmonitored sets, the k-FP classifier sees an overall decrease in r_{20}-precision of around 18.9%, whereas the DF classifier sees an overall decrease of around 9.1%. Additionally, both classifiers show a consistent decrease in recall. The recall of the k-FP classifier drops steeply

from 87.9% to 28.6%, a decrease of 59.3%. The DF classifier also shows a consistent, albeit less steep decrease from 94.6% to 60.3%, a decrease of 34.3%.

We therefore conclude that the reduction in classifier recall is present for the spectrum of potential QUIC trace presence in the dataset. Additionally, the features learned by the DF classifiers appear to be more robust to the presence of QUIC traces.

C Another Perspective: Precision-Recall Curves

In attempting to identify monitored web-pages, some observers may prefer precision to recall or vice-versa. Classification can be made more or less precise by adjusting a classifier's decision threshold.

For a given sample the classifier reported a confidence value, p_c for each *monitored* class c, where $0 \leq p_c \leq 1$. If the maximum confidence over the different classes, $\max_c(p_c)$, is below the decision threshold we classified the sample as an unmonitored sample. Otherwise, we classified the sample as the associated monitored class. Finally, this procedure was repeated for each sample across 10 splits of the dataset, and r_{20}-precision and recall scores were calculated over all samples.

The r_{20}-precision-recall curves shown in Figure 11 for the generalisation (Section 6.1), comparison (Section 6.3), joint classification (Section 7), and control packet (Section 8) experiments echo our previously stated findings. In addition, the following findings are further illustrated by Figure 11.

- Incorporating unmonitored QUIC samples into training may improve precision of the p-FP(C) and Var-CNN classifiers (Figure 11a).
- The Var-CNN classifier is fairly robust across the QUIC, TCP, Mixed, and Split settings (Figure 11b).
- Classification seems to be slightly easier when classifying QUIC as opposed to TCP (Figure 11b).
- The k-FP classifier and the time component of the Var-CNN classifier (Var-CNN$_T$) benefited the most from the removal of small packets (Figure 11c).

D Descriptions of Features

Below we describe the features utilised in Section 6.2. For a more detailed description of the computation of the features see Appendix E in the work of Li et al. [39], or the respective papers.

Packet count. Features based on packet counts: the total packet count, the count of outgoing packets, the count of incoming packets, the ratio between the incoming and total packet counts, and the ratio between the outgoing and total packet counts.

Time Statistics. The maximum, mean, standard deviation, and third quartile of the inter-arrival times for the total, incoming, and outgoing packet sequences. Additionally, the first, second (median), and third quartiles of the packet timestamps, as well as the total transmission time.

Transposition. The total number of packets before each of the first 300 incoming packets and before each of the first 300 outgoing packets.

Intervals I–III. An interval is the window of packets between a packet and the previous packet with the same direction. Interval I [3] records the number of packets in each incoming and in each outgoing interval. Interval II [67] records the frequency of intervals sizes for sizes from 0 to 300. Intervals with more than 300 packets are counted as containing 300 packets. Interval III [39] is Interval II with the 3rd to 5th, 6th to 8th, and 9th to 13th entries each summed to a single value.

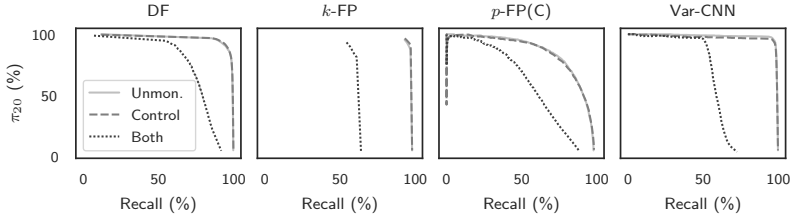
Bursts [3]. The number of packets in each sequence of outgoing packets with no two adjacent incoming packets. Additionally, the maximum and average number of packets in each burst, as well as the total number of bursts and the number of bursts with more than 5, 10, and 20 packets.

Packet Distribution. The number of outgoing packets in the first 200 overlapping chunks of 30 packets, the standard deviation, mean, median, and maximum of these 200 features, and the sums of each non-overlapping subsequence of length 10 of these features.

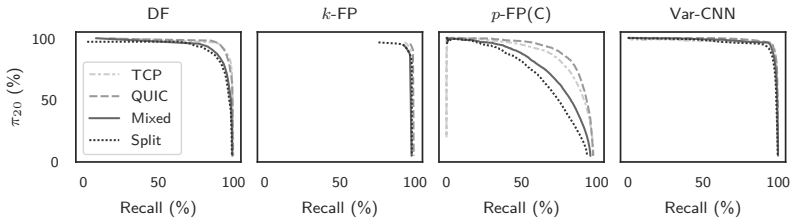
First20, First30 Pkt. Count, and Last30 Pkt. Count. The packet sizes of the first 20 packets, as well as the incoming and outgoing packet counts of the first and last 30 packets of the trace.

Pkts per Second. The number of packets transmitted in each second for the first 100 seconds, as well as the standard deviation, mean, median, minimum, and maximum of these features. Additionally, the totals for each of the twenty 10-second intervals.

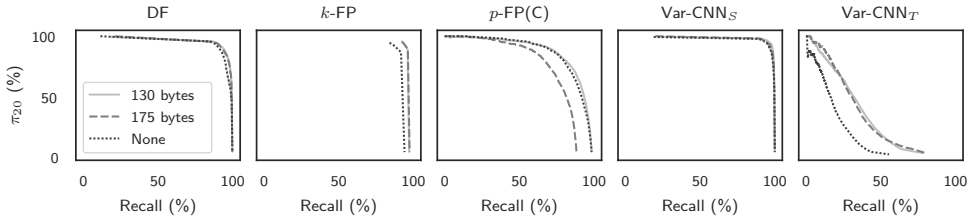
CUMUL [60]. A sample of 100 points from the piecewise linear interpolation of the cumulative sum of the packet sizes.



(a) r_{20}-precision-recall curves for the various classifiers when trained on TCP samples but tested in the settings where the client does not visit websites using QUIC and there exists QUIC traces in the open-world (Unmon.); and where the client may additionally visit web-pages using either QUIC or TCP (Both). The performance in the TCP-only setting is shown for comparison (Control).



(b) r_{20}-precision-recall curves for the various classifiers when trained and tested on TCP samples (TCP), QUIC samples (QUIC), TCP and QUIC samples (Mixed), and TCP and QUIC samples along with the distinguisher (Split).



(c) r_{20}-precision-recall curves for the various classifiers when evaluated on the TCP dataset where no packets, packets below 130 bytes, and packets below 175 bytes have been removed. Var-CNN$_S$ and Var-CNN$_T$ correspond to the size and time components of the Var-CNN ensemble classifier

Fig. 11. Precision-recall curves for the various experiments, when varying the decision threshold to predict a positive class from 0 to 1.