Privacy of DNS-over-HTTPS: Requiem for a Dream?

Levente Csikor
National University of Singapore

Himanshu Singh†
IIIT, Naya Raipur

Min Suk Kang
KAIST

Dinil Mon Divakaran
Trustwave

***Abstract*—The recently proposed DNS-over-HTTPS (DoH) protocol is becoming increasingly popular in addressing the privacy concerns of exchanging plain-text DNS messages over potentially malicious transit networks (e.g., mass surveillance at ISPs). By employing HTTPS to encrypt DNS communications, DoH traffic inherently becomes *indistinguishable* from regular encrypted Web traffic, rendering active disruption (e.g., downgrading to the plain-text DNS) by transit networks extremely hard. In this work, we investigate whether DoH traffic is indeed indistinguishable from encrypted Web traffic. To this end, we collect several DoH traffic traces corresponding to 25 resolvers (including major ones, e.g., Google and Cloudflare) by visiting thousands of domains in Alexa's list of top-ranked websites at different geographical locations and environments. Based on the collected traffic, we train a machine learning model to classify HTTPS traffic as either Web or DoH. With our DoH identification model in place, we show that an authoritarian ISP can identify $\sim$97.4% ($\sim$90%) of the DoH packets correctly in a closed-world (open-world) setting while only misclassifying 1 in $10,000$ Web packets. To counter this DoH identification model, we propose an effective mitigation technique, making the identification model impractical for ISPs to filter and consequently downgrade DoH to plain-text DNS communications.**

***Index Terms*—DoH, DNS, privacy, machine learning**

1. Introduction

Since its inception, the Domain Name System (DNS [52]) has been a plain-text protocol for all practical purposes. As almost every communication is preceded by a domain name resolution, adversaries (e.g., ISPs, censors, authoritarian governments [6]) having access to plain-text DNS (Do53) traffic can easily track user activities [10], [27], [35], [46], tamper with the data [6], [14], or even block connection attempts as witnessed recently [7], [34].

Of the several privacy and confidentiality enhancing proposals in the past (§2.1), DoT (DNS-over-TLS) [39] and DoH (DNS-over-HTTPS) [37] have been receiving particular attention. Both encrypt DNS communications between the stub resolver and the recursive DNS server, thereby mitigating any eavesdropping attack on DNS, while also providing integrity against data tampering. DoT uses a unique TCP port 853 (not used by any other well-known application), to send DNS data over TLS. On the other hand, DoH uses port 443 and, particularly, secure HTTPS to encrypt DNS communication; in other words, it uses the same protocol as encrypted Web traffic. While it is easy to block or apply traffic analysis [38], [65] on DoT communication using port-based filtering, it is impractical to do the same for DoH traffic by filtering on the port number, as it would also block normal Web communications of users. Successful blocking of these new protocols would force the users to fall back to plain-text DNS (Do53), thereby letting adversaries to carry out the traditional filtering mechanisms. DoH has an advantage over DoT, as it cannot be blocked using simple rules. Beyond port-based filtering, one could imagine constructing rules using information such as IP addresses of end-points and plain-text domain information in TLS handshakes, i.e., Service Name Indication (SNI); however, as we explain in §2, they are not only fundamentally limited for this purpose, but there are also ongoing efforts to defeat the use of such information for identification purposes, e.g., [25].

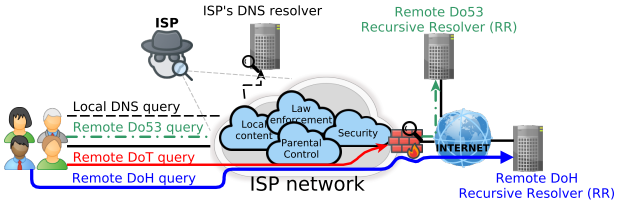


Figure 1: Plain-text Do53 can always be monitored. Due to encryption, DoT cannot be eavesdropped, but can be filtered via port 853. DoH blends into the HTTPS traffic; therefore, it is non-trivial to identify and filter it.

By using HTTPS, DoH packets inherently blend with encrypted Web packets on the wire. Since Web traffic is the most commonly used protocol on the Internet today [53], not being able to distinguish DoH traffic from encrypted Web traffic — we define this as the *indistinguishabilty* property — will make it harder for an authoritarian regime to block DoH traffic. We therefore envision that DoH traffic becomes indistinguishable from encrypted Web traffic, thereby defeating any attempts to thwart the adoption of DoH by users.

In this work, we investigate whether DoH traffic is indeed indistinguishable from encrypted Web traffic. While in [69], there has been an attempt to distinguish *DoH flows from Web flows*, to the best of our knowledge, our work is not only the first comprehensive study on this important indistinguishability property, but is also the one to classify on a *per-packet basis*. Our approach enables to react, i.e., block or drop corresponding flows or packets, in near real time as well as facilitates different traffic analysis attacks such as domain fingerprinting [65]. We take the first steps in developing a machine learning model for identifying and differentiating DoH traffic from

†*Himanshu Singh was an intern at the National University of Singapore during this work.*

encrypted Web traffic. To this end, we make our first main contribution: we develop an automated Docker container to collect several DoH traffic traces that capture the DNS communications made with 25 DoH servers, including the most prominent ones, i.e., Google, Cloudflare, Quad9, and CleanBrowsing (§4.1). As clients, we employ Mozilla Firefox browser that supports DoH since the beginning [64]. We automate the process of visiting the first 20,000 domains of Alexa's top one million websites at different times of the day and at different geographical locations with multiple environments (e.g., `x86, arm64`).

Based on these traces, we make our second main contribution: we train (§4.3) and build a supervised machine learning model (§4.4) to classify HTTPS traffic into (encrypted) Web and DoH traffic. To achieve this, we use only a small number of visible properties of the encrypted traffic as *features*, and exclude any prior knowledge such as IP addresses of the DoH servers (e.g., `104.16.248.249` for Cloudflare), SNI (Server Name Indication) in the TLS handshake (e.g., `mozilla.cloudflare-dns.com`), etc. Our model correctly identifies 97.4% (90%) of the DoH traffic in the closed-world (open-world) settings while only misclassifying 1 in 10,000 Web packets (§5). Essentially this means, network adversaries can deploy our identification model to detect and block DoH traffic with negligible effect on Web traffic due to the low false-positive rate (FPR).

We go further, and as our third main contribution, we develop and study multiple techniques to counter the proposed identification model (§6). Particularly, we show that the suggested padding technique (e.g., RFC 8467 [51]) does not help in blending DoH traffic more with the Web traffic. Thus, we study a new set of padding techniques on DoH packets (§6.2) that effectively and significantly brings down the classification accuracy of the identification model impacting its practical use for adversarial filtering. In particular, even at an unacceptable high FPR of 10^{-3}, the model can correctly classify only 83% and 0% of the DoH packets in a closed- and open-world setting, respectively. At lower FPR of 10^{-4} (required for practical deployment), the model's detection capability further deteriorates in the closed-world setting to 53% (§6.3). Our anti-identification technique is easily deployable at a user's end-system to beat a network adversary's DoH identification model, rendering it impractical for filtering (§7).

2. Background and Related Works

In this section, we give a brief overview on the evolution of privacy and security measures for DNS communications. We discuss whether encrypted DNS can indeed provide full privacy one might desire. Subsequently, we give an overview of the related works.

2.1. Encrypted DNS and its Penetration

Since the inception of DNS, the lack of integrity of Do53 has allowed manipulation attempts, e.g., altering the DNS resolution for services such as online banking, shopping, and social media, or redirect visitors to phishing websites. DNSSEC [71] enables an authoritative name server being responsible for a particular domain to cryptographically sign DNS responses, thereby providing integrity to DNS; but it does not provide privacy guarantees. DNSCrypt [23] aimed at encrypting and authenticating the DNS channel using elliptic-curve cryptography. However, as its specification has not been submitted to the IETF yet [24], it has neither become standardized nor ubiquitous.

In 2015, the IETF defined DNS-over-TLS (DoT, [39]) to encrypt DNS communications in a standardized way. However, since it uses a designated port number (`853`), ISPs can easily identify DoT traffic (to a remote resolver), and subsequently filter it out. Via such disruptions, ISPs can force the requester to downgrade to Do53, and this *may* happen seamlessly according to the opportunistic privacy profiles [22] that most stub resolvers adapt [40].

In 2017, DNS-over-HTTPS (DoH, [37]) was defined by the IETF, which uses the same level of encryption but uses the well-known destination port number `443` used for encrypted Web communications. Thus, besides being uninterpretable by an eavesdropper, DNS queries (and responses) over HTTPS can easily bypass firewalls (cf. Fig. 1). Put it differently, a firewall cannot *differentiate* between DoH and Web packets using the same service port. We explain in §2.2, why other fields (e.g., IP address) and existing techniques fall short in achieving this.

Note that these privacy-enhancing encrypted DNS techniques are only used between stub resolvers and recursive resolvers. While this potentially protects the client's DNS queries and responses from an adversary, DNS information *may* be exposed in the latter stages of the DNS resolution, when the recursive resolver does not have the answers cached. The exact source of the query already becomes obfuscated, though, even in case of EDNS Client Subnet (EDNS CS, [16]). That part of the communication is completely out of scope of DoH (and DoT) in general, and thus also of our work. Interested readers can refer to the IETF DNS PRIVate Exchange (DPRIVE) working group's Draft [48] for more details.

Penetration. DoH, being based on HTTPS, was first deployed (as a client) as part of Mozilla's Firefox browser [12], thereby having no requirement from the underlying operating system for encrypted DNS communications. The first big leap since then has been made by Mozilla (in 2018) by introducing a fully functional support for Trusted Recursive Resolvers (TRR) in its browser, Firefox [50]. Google made similar changes to its Chrome browser in 2019 [11]. Moreover, in 2020, Mozilla announced that it will use DoH, particularly, Cloudflare's TRR, as default for US users (though users can change these settings). Similar attempts raised concerns for ISPs (especially in the UK [43]) as almost all of their services will eventually break if DoH is in use and it relies on a third-party resolver. As a consequence of nominating Mozilla as the "Internet villain of the year", Mozilla did not enable DoH by default for UK citizens [43].

This, however, did not impose a barrier on the adoption of DoH, which, though still in its infancy, continues to gradually become popular. For instance, Microsoft recently announced addition of OS-level DoH support for Windows 10 Insiders [28], [54]. Besides, DoH is not only supported natively in iOS and macOS platforms [58], but they also provide APIs to customize DoH (and DoT) usage in any application, e.g., to connect to a DoH service if

using a WiFi router [57]. Similar applications are also available for Android [33], [47]. Furthermore, recently, Comcast joined Cloudflare and NextDNS in the TRR program of Mozilla as the first ISP providing DoH services through the browser [15].

2.2. ISP's Surveillance Techniques beyond DNS

ISPs and authoritarian governments often deploy monitoring and filtering services based on passive Do53 data. For instance, DNS-based blocking is applied in Switzerland to ban online foreign gambling, in China to put a barrier on political speech [60], and also occasionally in several countries during political turmoils [6]. Even though IP addresses of end-points (§2.2.3), pure (unencrypted) HTTP traffic (§2.2.1), deeper analysis on HTTPS traffic [61], [72], or plain-text domain information of TLS (§2.2.2) of a packet may also reveal sensitive information, still, the main reason behind the success of DNS-based monitoring, is its inherent simplicity and easy-to-track nature. First, even if the services and their contents can be geographically distributed (i.e., can be accessed at multiple IP addresses), can be completely relocated (i.e., assigned new IP address), or can share the resources (i.e., multiple services behind the same IP address), their domain names change only infrequently. Second, even if we become aware of the IP address of a service, accessing its particular content (e.g., the web site itself) for censoring purposes is *not* straightforward. For instance, consider `example.com`, which, when typed into the browser's URL field, gives the usual content, however, trying to use its IP address as a URL instead, i.e., `http://93.184.216.34` (at the time of writing), results in `Error 404`. The reason is that the URL itself conveys necessary information for the service to indicate how and which content should be served. Moreover, when it comes to HTTPS services (which dominate the Internet, cf. §2.2.1), without a proper URL, we cannot indicate the servers for which service they have to prepare the certificate first (see more details in §2.2.2).

Consequently, if an ISP loses the sight of the DNS data, all monitoring and filtering services become affected. Next, we briefly overview what other (meta-)data one may consider for trying to filter a specific service when the DNS communication is encrypted.

2.2.1. Non-encrypted Subsequent Flow. After encrypted DNS communication, if the subsequent flow (sent from the user towards the requested service) is using a non-encrypted channel, i.e., connects to a HTTP website, then by observing the HTTP header, the corresponding traffic can be easily filtered. But, web traffic is increasingly becoming encrypted, with recent reports estimating 94% of web traffic as already encrypted [30].

2.2.2. Server Name Indication. Every website relying on HTTPS requires TLS certificate that facilitates encrypted communication via PKI (Public Key Infrastructure) as well as provides identity assurance of the certificate's holder. When establishing a secure connection to a service, first the corresponding certificate has to be obtained. This is expressed by the Server Name Indication (SNI) field in the first TLS packet, in particular, the `Client Hello` message. Since the TLS header is not encrypted, the SNI information is visible to any eavesdropper and it contains a unique ID of the service (e.g., `www.ieee-security.org`) making it easy to filter. However, in TLS 1.3, for this specific reason, Encrypted Client Hello (ECH [25], formerly known as encrypted SNI, eSNI [26]) has been introduced by the IETF, which can make pure SNI-based filtering infeasible. However, since ECH requires public keys to be distributed through DNS `TXT` records (`_esni.`*domain_name*), using ECH for connecting to a DoH service in the first place would create a pragmatic loop. In this case, therefore, relying on ECH is either neglected (e.g., even if it is enabled in Firefox, it is not used to connect to Cloudflare's DoH resolver, however Cloudflare supports ECH), or done via other channels (e.g., Do53), which can eventually be monitored. Note, by design, if no keys could be obtained for ECH, browsers fall back to the plain SNI transparently. Nevertheless, mere SNI-based filtering of DoH services is yet impractical as any (rogue) DoH service can be created and put behind an obfuscated SNI that does not reveal the existence of a DoH resolver, e.g., `www.exampleconf.com/submit`.

2.2.3. IP address. Recent studies [36], [56] have shown that if an adversary has a plausible set of sites and the corresponding IP addresses a user might visit, then the privacy of encrypted DNS/SNI (i.e., DoH and eSNI) is limited. However, not only the set has to be huge, but making it up-to-date is a daunting task. Besides, as mentioned in §2.2, due to resource sharing, services can have the same public IP address. For instance, consider the DoH resolvers `ContainerPI`, `LibreDNS`, `hostux`, `Tiarap`, etc., they all run their Web and DoH services behind the same IP addresses. More importantly, we found that DoH queries sent to most of the Cloudflare IP addresses within the range `104.16.0.0/16` are successfully resolved (cf. A.12 in Appendix). Clearly, blocking the whole IP range would make many of today's web services inaccessible as Cloudflare is one of the market dominators in cloud-based CDN (Content-Delivery Networks), hosting, load-balancing, DNS services, and cyber-security frameworks (e.g., DDoS protection). Therefore, an ISP cannot simply make critical decisions merely by looking at the destination IP addresses of the packets.

2.2.4. JA3(S) TLS Fingerprinting. Developers at Salesforce Engineering open-sourced JA3 [5], a TLS client/server fingerprinting method, originally designed for malware detection. JA3 hash represents the fingerprint of a TLS application and it is shown that many clients/servers can be correctly identified via this method, e.g., the `ToR` client and server, `trickbot` malware [4].

While JA3 is useful to maintain a fingerprint database of clients in a controlled environment, for identifying individual Web or DoH services in the wild, yet it is incomplete. First, multiple physical and/or virtual instances of the same service may run behind the same public IP address (cf. §2.2.3). However, for the different configurations of the underlying systems, a Web/DoH service can produce multiple JA3S hashes [45] depending on which physical/virtual server our request was directed to by a load-balancer. In particular, since Firefox, when

configured to use Cloudflare's DoH resolver (by explicitly defining its IP address), establishes multiple parallel connections in the background, we observed multiple different hashes for different `Server hellos` right away[1].

On the other hand, due to the same reason, there is a high chance that different services can have the same JA3S hash. Particularly, we find that the DoH resolver at Cloudflare (more precisely, one instance located behind the IP address `104.16.248.249`) has the same JA3S fingerprint as two completely different services within Google's network (a service behind IP addresses `172.217.194.103` and `74.125.200.95`).

We conclude that while other (meta-)data might give some insight into what services a customer wants to visit, unsurprisingly none of them would be able to easily substitute the DNS-based monitoring and content filtering.

2.3. Related Works

Website fingerprinting techniques have been shown to be successful in revealing information from encrypted HTTPS web communication [32], [55], [66], [70]. On the other hand, adversarial learning techniques against fingerprinting methods have been studied recently. In particular, it has been shown that ML models are vulnerable to adversarial examples [2], which are carefully crafted samples mixed into the dataset leading to misclassification. For instance, in [1] dummy packets are injected into Tor user traces to break patterns.

A recent work [38] analyzed DoT traffic for website fingerprinting. In particular, a DoT fingerprinting method was proposed to understand how much information can be deduced through traffic analysis on DoT packets. The authors used machine learning techniques to identify and classify individual websites into three main groups, namely dating, gambling and health insurance. When DNS messages are not padded, they show that DoT traffic can be categorized with 0.07 FNR and 0.05 FPR. Moreover, individual websites can be identified with a false-negative (false-positive) rate of 0.17 (0.005).

In [65], authors studied the effectiveness of traffic analysis attacks on DoH traffic. Authors show that features traditionally used for website fingerprinting are not suitable for DoH traffic analysis. Therefore, they engineer a new set of features and show that the proposed DoH traffic analysis is effective in identifying web pages. However, the study also shows that training a model in an environment other than the one where the model would be actually deployed, has a negative impact on the performance.

Most of the recent studies focus on how to reveal the content of the encrypted DNS messages and apply filtering/censoring accordingly, or even countering these mechanisms; but none of them comprehensively treats the specific problem of identifying DoH traffic itself. While identification is straightforward in case of DoT (by well-known port number 853), the latest work on DoH [65] assumes the relevant DoH packet stream to be readily identified.

A very recent work [69] attempted to differentiate DoH from Web traffic by different ML techniques. The proposed model detects DoH traffic and also identifies DoH clients with high accuracy. However, it differs from our work in multiple aspects. First, all features are defined at a flow-level. This allows the flow-based model to better capture the semantics of a DoH (TCP) connection (in comparison to a packet-based approach), as a flow contains more information. But this also means, feature computation would have to wait until the flow is completed. This also has deployment implications as explained in §3. Our model uses packet-level features; extraction of a feature corresponding to a packet depends on utmost one (immediately) preceding packet. Second, the features in the flow-based model also requires bidirectional flows; whereas our model features are extracted from unidirectional traffic (from client to server). Note that, not all ISPs/ASes would see bidirectional traffic of users; thus the bidirectional features are not available for all adversarial ISPs. Third, the flow-based analysis is carried out on one long-lasting connection (having all the DNS queries resolved), whereas in our case the connection to the DoH resolver is re-established for every domain visited. Finally, the evaluation is limited to a closed-world setting, without information about the DoH resolver used and other relevant aspects (e.g., geographical location). In contrast, we present results on both closed-world and open-world settings (§4.4), and on a diverse dataset (§4.1).

From the perspective of a censoring ISP, in [40], a DoH downgrade attack is studied, wherein different techniques (e.g., DNS traffic interception, cache poisoning, TCP RST injection) at different stages of the connection establishment (e.g., Do53 to obtain the IP of the DoH resolver, when connecting to a DoH resolver) are investigated to hinder the use of any DoH service. According to the opportunistic profile (defined in RFC 8031 [22]) all major web browsers implement, a default action whereby the victims seamlessly fall back to Do53 without explicit notification. However, the evaluation has been done in a controlled environment where the attacker is aware of which connection belongs to DoH (and which do not). In contrast, here, we argue that the very first step of differentiating DoH traffic from Web traffic is a critical part of any censoring efforts, and therefore we address this specific problem here.

An empirical study by Gillmor [29] has revealed that merely analyzing the packet size of a single encrypted DNS transaction can narrow down the queried domain. In a subsequent study [8], authors show that even if the reasonable padding strategy is used, a side-channel analysis on the encrypted and padded DNS traffic (DoT or DoH) can still correctly identify all connections for 32% of the websites considered (Alexa top-10k). In our analysis of countering the proposed model, we confirm that the advised padding technique (also recommended in RFC 8467 [51]) does not help in blending DoH traffic more with the Web traffic either. We term this padding technique as **PT(1)** in §6.2.

3. Threat Model

The primary goal of the attack we present is to *distinguish* encrypted DoH packets from HTTPS Web packets. We consider censorship adversaries (such as, ISPs in repressive regimes, local administrators with draconian

1. Note, unlike IP addresses, when a given service scales up and down, the number of JA3 hashes can easily rise and fall in a very short time.

Internet policies) as the main perpetrator. Our DoH identification makes a necessary intermediate attack step for achieving low-cost censorship against users using DoH. Without distinguishing DoH from Web, a censorship adversary learns absolutely no DNS information and, thus, is left with only high-cost and ineffective censorship options (see the available vectors in §2.2). After identifying DoH packets, a censorship adversary can *downgrade* the identified DNS sessions to Do53 by blocking the corresponding packets, rendering existing DNS-based low-cost censorship effective again.

The secondary, yet important, goal of the attack is to achieve *extremely low false positive rates* to prevent any noticeable disruption of non-DoH Web sessions. Note that a moderately low (e.g., a few percent) false positive rate may be sufficient enough for some strong censorship adversaries (e.g., countries under dictatorship) as they may be less concerned about collateral damage. However, sneaky or mild censorship adversaries may require false positive rates to be lower than the natural packet loss-rates of the Internet (e.g., in the order of 0.001 [9]) to cause only negligible degradation of non-DoH Web sessions.

In this work, we consider an adversary who is capable of monitoring all HTTPS packets of interest (i.e., Web and DoH sessions made by target citizens or employees), yet, unable to decrypt the end-to-end encrypted HTTPS packets. Our censorship adversary is able to collect and temporarily store some minimal per-packet metadata of the flows of interest, e.g., packet length, inter-packet arrival rates (see exact properties in §4.1). This can be achieved with an in-line adversarial monitoring application.

Scope and assumptions. When evaluating the effectiveness of our DoH identification attack, we consider a comprehensive list of padding mechanisms for all the features we use in the attack. This includes the existing padding proposals (e.g. RFC 8467 [51]), and a number of our own novel padding schemes.

Detection granularity: packet-level vs. flow-level. We propose a packet-level detection, where the classification between DoH vs. Web is made for each packet. A flow-based detection is also possible as presented in a recent work in [69]. We choose a packet-level detection granularity as it can be applicable to more general censorship applications for two reasons: (1) a packet-level detection can enable flow-level filtering while a flow-level detection cannot be used for packet-level filtering; and (2) a packet-level detection can perform immediate filtering (thus, immediate censorship) without waiting for a flow to complete.

4. Machine Learning for DoH Identification

When communication is encrypted, the payloads are no more available to help identifying the traffic class (DoH or Web) to which a certain packet belongs to. For a network adversary with the capability to passively monitor communications, the only data that can be observed are the visible parts of a packet stream (e.g., packet headers), which do not tell directly what kind of communication is being carried underneath the protocol (HTTPS). While traffic properties, such as packet length, do appear useful in differentiating DoH and Web traffic (§5), a simple set of rules cannot help to make the clear distinction (see later in §4.1 and Fig. 2). This leads us to apply machine learning algorithms to build models that classifies HTTPS traffic into the two classes.

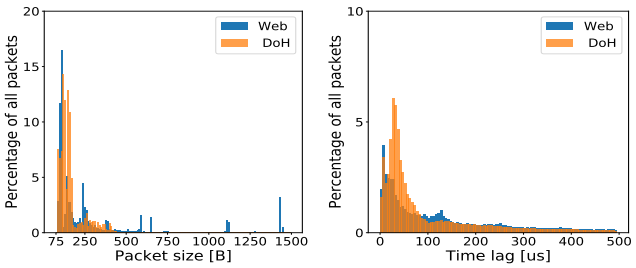
4.1. Data Gathering

Following the traditional workflow of a state-of-the-art machine learning task, we first focus on accurately capturing the data that is required for analysis as well as model building and evaluation. We create an easy-to-deploy Docker container bundled with several Python and BASH scripts that automate the whole process[2]. Within the container, a Mozilla Firefox browser is instructed via the Selenium API to visit the first 20,000 domains from Alexa's list of top 1 million websites [3]. And for resolving their domains, we configure the browser to use the four most popular DoH resolvers available (cf. Appendix A.1) — `Cloudflare`, `Google`, `Quad9`, and `Cleanbrowsing`, one at a time. Later, we gather traffic traces from 21 less popular resolvers as well, e.g., from `OpenDNS`, `LibreDNS`, `Comcast`, etc. [20]. Firefox is opened and closed between two consecutive visits to flush the DNS cache. Visiting 20,000 websites, capturing and processing the corresponding traffic traces take around 48 hours for our containers (because of safe grace periods, long timeouts for websites that do not load quickly, etc.). Thus, our website visits are always spanning across two days irrespective of the location where we record it.

Diversity. Since not all networks, browsers, architectures and DoH services behave identically, the properties (both absolute and relative) of the DoH communications can vary across time, location and environment. For instance, even though DoH is a standard, there is no further demands from the DNS protocol itself. Consequently, a DNS query might contain multiple domains to be resolved, or EDNS CS, thereby resulting in different packet sizes among different implementations. Therefore, to rigorously verify and improve the robustness of our model, we also gather data at different geographical locations using multiple environments. In particular, we collect traces on `x86` architectures from cities across different continents including South America (Loc_A^{x86}), North America (Loc_B^{x86}), and Asia (Loc_C^{x86}). By taking advantage of Cloudlab's facility [68], the North American dataset is divided into three subsets. In particular, Loc_{B1}^{x86} and Loc_{B2}^{x86} are gathered in the eastern and western side of the U.S, respectively. Loc_{B1}^{arm} is gathered at $LocB$ using `arm` architecture. See more details in Appendix A.13.

Labeling. During the automated website visits, we capture the corresponding encrypted traffic trace filtered on port 443 (default destination port for HTTPS communications) along with the TLS keys to later decrypt the communication for labeling purposes; we label *each packet* as either DoH or Web. To achieve this, we use the protocol identification feature of the most recent version (v3.2) of `tshark/Wireshark` (that already supports DoH), with some correction (see Appendix A.2 for details). Finally, we export all relevant packet header information (i.e., visible meta data and packet headers) into CSV files that

2. For the sake of reproducibility and fostering further research, we publish the Docker containers used for data gathering as well as the sources codes of our models [17]–[19].



(a) Distribution of packet length (bin size: 10 bytes)

(b) Distribution of time lag (bin size: 5 μs)

Figure 2: Distributions of the DoH and Web packets, for traffic for the four prominent DoH resolvers.

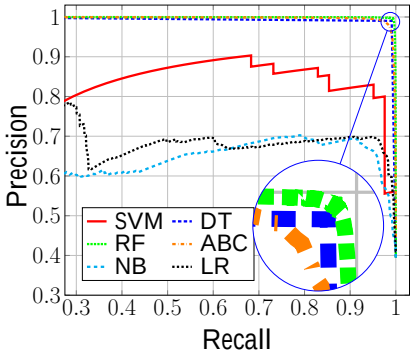


Figure 3: Precision-Recall curves for all ML models considered applying a 90:10 training-testing ratio on the data gathered using Cleanbrowsing's DoH resolver.

can be processed to carry out training and testing of different machine learning models.

Balance. The datasets we gathered are imbalanced in nature, there are more Web packets than DoH packets. While we use balanced dataset for training, we stick to the original imbalanced ratio for testing our models.

Features. Packet headers contain only absolute and static values such as source and destination IP addresses, source and destination port numbers, sequence number, timestamp, transport protocol, etc. These values cannot be used as features for building a machine learning model, since they do not capture the dynamism of the Internet; e.g., as mentioned in §2.2.3, IP address is not a reliable information for identifying DoH traffic. Instead, we engineer a small number of features from two basic information — packet size and time lag (inter-arrival time between packets) — that can be easily extracted from network traffic. Fig. 2a and Fig. 2b give the distributions of packet size and time lag, respectively. In both figures, we observe considerable overlap between DoH and Web traffic, thereby making simple rule-based differentiation of DoH and Web challenging.

Therefore, to build a DoH identification model, we engineer four simple features derived from packet size and time lag: the current packet length (`pkt_len`), previous packet's length (`prev_pkt_len`), the inter-packet arrival time between the current and the previous packet (`time_lag`) and the same time between the two preceding packets (`prev_time_lag`) *within a flow*. We use $\mathbf{f}$ to denote the feature vector; $\mathbf{f} = f f_1, f_2, f_3, f_4 g = f$`pkt_len, prev_pkt_len, time_lag, prev_time_lag`$g$.

4.2. Choosing the Right Model

We evaluated six machine learning models, namely, Support Vector Machines (SVM), Decision Trees (DT), Random Forest (RF), AdaBoost Classifier (ABC), Naïve Bayes (NB), Logistic Regression (LB), in classifying traffic on port 443 into Web and DoH. The evaluation was carried out on the data gathered using Cleanbrowsing's DoH resolver. The Precision-Recall curve (see Sec. 4.4 for explicit definitions) is given in Fig. 3. While RF, ABC and DT perform the best, we choose *RF for all further analyses*, as RF is not only fast in prediction, but is also generally known to perform well in other related scenarios involving network traffic [31], [38], [67]. Appendix A.3 discusses the model hyper-parameters.

4.3. Building the Models

To build DoH identification models, we use the Python programming language and rely on its machine learning modules and libraries (primarily, `scikit-learn` [62]). We train three different models, each time making improvements from one model to the next.

First, we take a naïve approach, and train four distinct sub-models for the four well-known resolvers considered using 4M packets for each. We refer to the four models as $M_{1\ X}$, where X 2 f `CF`, `G`, `CB`, `Q9` g. Henceforth, `CF`, `G`, `CB`, and `Q9` abbreviate `Cloudflare`, `Google`, `Cleanbrowsing`, and `Quad9`, respectively.

Second, we train one aggregated model $M_{\mathbf{2a}}$ using data of all the above-mentioned four resolvers combined (16M packets). Then, we further analyze the traces and patterns to look for possible improvements (explained in §5.2); and eventually we retrain the model with the updated dataset obtaining $M_{\mathbf{2b}}$.

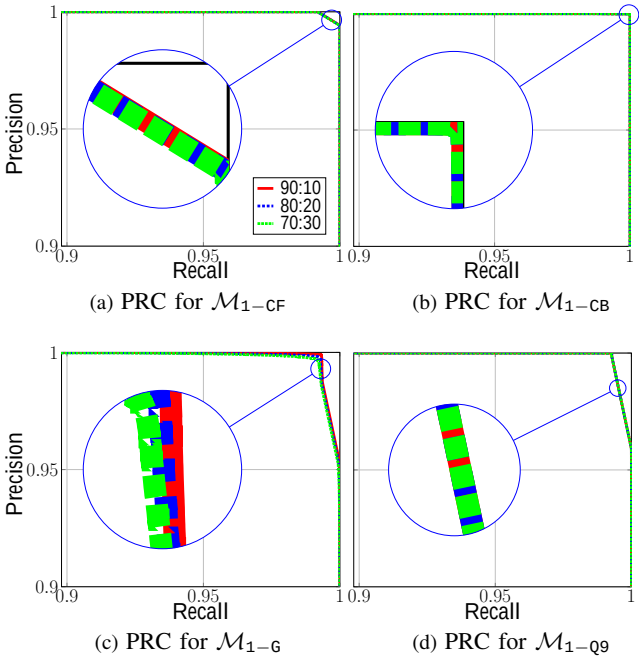
Finally, we hand-pick three additional less-known DoH resolvers (cf. §4.1) to train our final model $M_{\mathbf{3}}$. In particular, as we test the performance of $M_{\mathbf{2b}}$ on data not used for training at all (cf. §4.4), the resolvers for which the model $M_{\mathbf{2b}}$ performs the worst are also used (besides the data for the aggregated model) for training the new model. Thus, the final model $M_{\mathbf{3}}$ is trained on the data we gathered using *seven* DoH resolvers, i.e., on `CF`, `G`, `CB`, `Q9` and three further ones (`OpenDNS`, `doh.li`, `OpenDNS`). Refer to Table A.2 in Appendix for a short summary.

To evaluate the robustness of our models, we choose one location, *LocC* for training, and test on the data gathered from all other locations and environments. The only exception is, when we evaluate the performance of *localized* models in §5.4.1.

4.4. Testing the Classification Models

According to our different models, we divide our testing into different scenarios w.r.t. the number of resolvers and traffic traces considered at a time. We first describe the metrics used for evaluating the models.

Performance metrics. True positives (TP) are the packets correctly classified as DoH and false positives (FP) are the Web packets misclassified as DoH. Accordingly, the false-positive rate (FPR) and false-negative rate (FNR) are



(a) PRC for $\mathcal{M}_{1-\texttt{CF}}$ (b) PRC for $\mathcal{M}_{1-\texttt{CB}}$

(c) PRC for $\mathcal{M}_{1-\texttt{G}}$ (d) PRC for $\mathcal{M}_{1-\texttt{Q9}}$

Figure 4: Precision-Recall curves for all four prominent resolvers with different training-testing ratios. Common legend as in Fig. 4a.

calculated as $\frac{\text{FP}}{(\text{FP+TN})}$ and $\frac{\text{FN}}{(\text{FN+TP})}$, respectively (where TN is the true negatives and FN is the false negatives).

From these metrics, we derive precision, recall, F_1-score and accuracy. While precision gives a measure of the fraction of true DoH packets among all packets classified as DoH ($\frac{\text{TP}}{\text{TP+FP}}$), recall denotes the fraction of true DoH packets that are correctly classified as DoH ($\frac{\text{TP}}{\text{TP+FN}}$ or 1 FPR). F_1-score is a harmonic mean of precision and recall (2 $\frac{\text{Precision Recall}}{\text{Precision+Recall}}$). Accuracy is the fraction of the correct ones among all classifications ($\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$). For imbalanced dataset, precision, recall and F_1-score are better metric than accuracy for measuring the performance of a model, since they do not get biased by the larger class. Therefore, we use F_1-score for our discussions on results here; we also plot the Precision-Recall curves as and when necessary. In addition, when it comes to practical deployment and countermeasures, we also present the recall of the models for significantly low FPR values. Appendix A.7 details all other important metrics for our models.

Training-testing ratio. Before proceeding into the detailed experimentation, we first choose the proper training-testing ratio. In Fig. 4, the Precision-Recall curves can be seen in the domain of $[0.9 : 0.9]$ for the four baseline sub-models $M_{1\ X}$ (cf. §4.3) considering different training-testing ratios of 90:10, 80:20, and 70:30. Note the common legend denoted in Fig. 4a. As we can observe, none of the different training-testing ratios have any significant impact. Therefore, henceforth we choose to apply the 90:10 training-testing ratio for all evaluations. Note, training and testing are done on (non-overlapping) partitions of the datasets throughout all experiments in this work.

Closed-world and open-world. When it comes to testing, we use either closed- or open-world settings, different in the dataset used for testing the models. In closed-world setting, the dataset used for training and testing are for the same resolvers and the same set of domains visited. To confirm the performance in this setting, we also evaluate each model with k-fold cross-validation with $k = 5$. For open-world settings, we have two definitions. In case of OW_1, we test the models on data corresponding to resolvers unseen in the training phase (but for the same domains visited); while in OW_2, the testing dataset corresponds to both new resolvers and new domains visited. In OW_2, to gather new domains, we randomly select a batch of 5000 domains from Alexa's list within the range of $[5000, ..., 20000]$. Recall, we apply the 90:10 training and testing ratio. For instance, the model trained on the four most prominent DoH resolvers when visiting the first 5000 domains is considered as closed-world setting. In contrast, when the same model is tested on, say, Comcast's DoH resolver, it is considered as an OW_1, and if the visited domains are also different, it is considered as OW_2.

Testing scenarios. In the first scenario (§5.1), S_1, we evaluate each naïve sub-model $M_{1\ X}$ in a closed-world setting. Data-wise, considering the 90:10 training-testing ratio, this means $3.6M$ packets are used for training, and $400K$ packets are used for testing each sub-model.

Next, we evaluate each sub-model in a open-world setting, i.e., a trained sub-model for one specific resolver (e.g., $M_{1\ \texttt{CF}}$) is evaluated on the data gathered for the other prominent resolvers (e.g., `Google`), one at a time. And this is carried out for each of the sub-models. Note, in this case, each testing dataset has a size of $4M$ packets and contains no packets captured for the resolver a particular sub-model is trained on.

In the second scenario (§5.2), S_2, we evaluate the aggregated model $M_{\mathbf{2a}}$ trained on the data for the four prominent resolvers (i.e., `CF`, `G`, `CB`, `Q9`) in a closed-world setting, then refine the dataset and train and evaluate a new model $M_{\mathbf{2b}}$ as explained in §5.2. Subsequently, we randomly select *ten* more publicly available DoH resolvers from [20] for our open-world evaluations, namely `PowerDNS`, `doh.li`, `Comcast`, `DNSSB`, `Flatuslifir`, `LibreDNS`, `OpenDNS`, `CZNIC`, `42l`, and `ContainerPI`.

In the third scenario (§5.3), S_3, our further aggregated model $M_{\mathbf{3}}$ is evaluated in a similar way. First, it is evaluated in a closed-world setting, i.e., our final model trained on the data of seven DoH resolvers are tested for the same resolvers. Then, the model is tested in the open-world settings by testing on all the remaining publicly available DoH resolvers (totaling 25 in count [20]), and a new set of domains different from those used in training.

Finally (§5.4), we analyze the DoH identification models from the perspective of an adversary. For the best models, we evaluate their detection capabilities at stringent false-positive rates (FPRs).

5. Evaluation of DoH Identification Models

5.1. Scenario S_1: Naïve Sub-Models

The performance of the naïve sub-models $M_{1\ X}$ is summarized in Fig. 5, where the y axis shows the F_1-score of each sub-model tested on the data gathered on all four resolvers. We see that, while the models perform

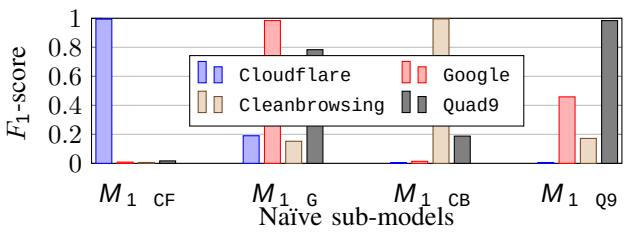


Figure 5: F_1-score of different models on different traces.

quite well (with F_1-score of 0.9894 on average) in the *closed-world* scenario, in case of the *open-world* setting, on average, the F_1-score (precision and recall) degrades by 82% (36% and 87%, respectively).

Closed-world. We observe that the model trained on `Cleanbrowsing` (beige bar at $M_{1\ CB}$) performs the best having an F_1-score of 0.9957. `Quad9` (gray bar at $M_{1\ Q9}$), on the other hand, achieves relatively the worst performance but with still a high F_1-score of 0.984. k-fold Cross-validation also gave similar results (cf. Appendix A.4).

Open-world. In this setting, we observe a slightly different tendency. While $M_{1\ G}$ gives the best results (second stack of bars) when tested on the traces of all the other resolvers (F_1-score of 0.3755 on average), $M_{1\ CF}$ (first stack of bars) provides the worst average F_1-score (0.01). On the other hand, while $M_{1\ G}$ provides 0.7883 F_1-score for `Quad9`, it is hardly useful in identifying DoH traffic to the other two resolvers.

5.2. Scenario S_2: Aggregated Model

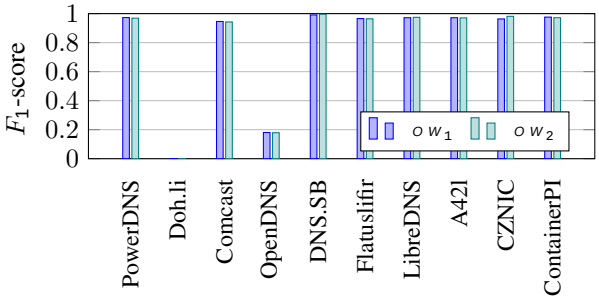
We have seen that while the individual models perform quite well in the closed-world setting, their performance is unacceptably low when we cross-evaluate them.

Closed-world. We first train an aggregated model, M_{2a}, with all traffic traces gathered using all four resolvers combined; recall, the combined trace has more than 8 million packets. M_{2a} provides an F_1-score of 0.977. We observe, the model trained on this combined data drastically decreases the false-negative rates in comparison to the naïve models, when testing on data corresponding to all resolvers (cf. §5.1). For exact values, refer to Table A.3 in the Appendix.

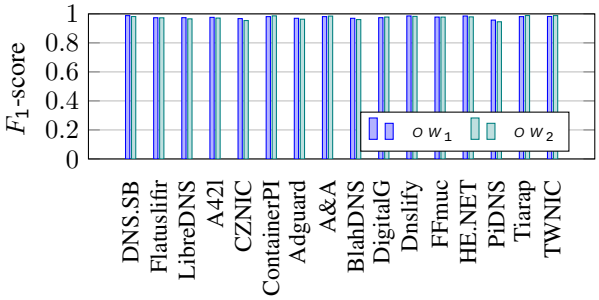
Continuing analyzing our model, we *(i)* look into the most useful features for classification. Then, to improve the performance, i.e., to reduce the false positives and false negatives further, we *(ii)* analyze the datasets.

For *(i)*, we use SHAP [49], a game theoretic approach that uses the classic Shapley values to explain the output of machine learning models. In essence, we found that the relative features `prev_pkt_len` and `prev_time_lag` are the most important ones, however the impact of the other two absolute features are also significant (cf. §A.5). This finding also justifies our relabeling endeavors after data gathering (cf. §4.1).

Subsequently, *(ii)* taking a closer look at the actual false positives after the latest testing phase, we observe that most of them are attributed to the DoH responses having similar properties with the Web requests. Particularly, most of the false positives belong to the subset of Web requests having feature values (especially for the



(a) Performance of $\mathcal{M}_{2b}$.



(b) Performance of $\mathcal{M}_3$.

Figure 6: F_1-score of M_{2b} (6a) and M_3 (6b) in both open-world settings, considering eleven and all remaining publicly available DoH resolvers, respectively.

feature `pkt_len`) similar to the DoH responses (see a comprehensive view of the statistics in Appendix A.6).

Consequently, we remove both DoH and Web response packets from our dataset[3] and *we focus on the requests, i.e., outgoing traffic, exclusively for the rest of our study* when training (and testing) a model. With this modification, F_1-score of our retrained aggregated model (M_{2b}) in the closed-world scenario increases up to 0.9911. Note, for k-fold cross-validations, refer to Appendix A.4.

Open-world. Next, we evaluate M_{2b} on traces we obtain by using additional publicly available DoH resolvers [20]. In particular, we randomly pick ten additional DoH resolvers, namely `PowerDNS`, `doh.li`, `Comcast`, `DNSSB`, `Flatuslifir`, `LibreDNS`, `OpenDNS`, `42l`, `CZNIC`, and `ContainerPI`, and evaluate our model for OW_1 and OW_2 settings.

F_1-scores for these ten resolvers are depicted in Fig. 6a. We observe that for the best five (i.e., `DNSSB`, `ContainerPI`, `PowerDNS`, `42l`, `LibreDNS`) out of the ten randomly selected resolvers having F_1-scores above 0.97, our latest model provides an average F_1-score of 0.9766 (0.9754) in OW_1 (OW_2). On the other hand, considering *only* the three worst performing ones (`Comcast`, `OpenDNS`, `doh.li`), the average F_1-score drops down to 0.3755 (0.3738) in OW_1 (OW_2). The main reason behind this significant drop can be attributed to the considerably higher false-negative rates; on average it is 0.666 (0.658) for OW_1 (OW_2). In case of `doh.li` (F_1-score: < 0.00008), furthermore, it was even observable by the human eye that for all important features, the mean values (with standard deviation) and 75-percentiles of the packets falling in the DoH class are significantly greater

3. Note, the datasets also become more balanced as the number of Web packets is highly affected by the large responses (§A.7).

than the corresponding values of the aggregated trace our model is trained on.

On the other hand, we also observe that the different open-world settings have negligible effect on the performance concluding that the accuracy of the model is not affected by the domains visited but only by DNS resolvers for which the data is collected.

5.3. Scenario S_3: Final Optimized Model

Since for some resolvers our model M_{2b} performs bad, i.e., our model operates with considerably high false-negative rate, we aggregate the corresponding traces[4] to our previous training data set (consisting of the four prominent resolvers only), retrain our final model (M_3), and evaluate its performance. The feature preferences has slightly changed (cf. Fig. A.3).

Closed-world. In this setting, we test the performance of M_3 on the traces obtained by using DoH resolvers `Cloudflare`, `Google`, `Cleanbrowsing`, `Quad9`, `Comcast`, `OpenDNS`, and `doh.li` (13M data points). The overall accuracy indicators are almost identical to the closed-world setting of (M_{2b}). In particular, this final model has an F_1-score of 0.9908 with a FPR and FNR of 0.0095 and 0.012, respectively. The k-fold cross-validation results are given in Appendix A.4.

Open-world. Finally, we test our model's performance on the traffic traces obtained by using the rest of the publicly available DoH resolvers, namely `DNSSB`, `Flatuslifir`, `LibreDNS`, `ContainerPI`, `42l`, `AdGuard`, `Andrews & Arnolds`, `BlahDNS`, `Digitale Ges.`, `Dnslify`, `FFmuc`, `HE.net`, `PiDNS`, `Tiarap`, and `TWNIC`. The results are given in Fig. 6b. We observe that the average F_1-score of M_3 becomes 0.9764 for OW_1 (0.9737 for OW_2) with FPR and FNR, of 0.006 (0.0054) and 0.042 (0.038) for OW_1 (for OW_2). In particular, on average for both open-world setting, M_3 performs the best for `DNSSB` with F_1-score of 0.9879 with FPR and FNR of 0.0064 and 0.02, respectively. On the other hand, the worst results are obtained for `PiDNS` with a low F_1-score of 0.957 with FPR=0.004 and FNR=0.08. Similar to S_2, the impact of the domains visited is negligible. Refer to Appendix A.8 for a brief performance evaluation when a DoH resolver and its regular web service reside behind the same IP address. For more accuracy metrics and additional analysis on a set of very recent resolvers, see Table A.6 and Fig. A.8 in the Appendix, respectively.

5.4. Robustness of the DoH Identification Model

When it comes to the deployment of a DoH filter, as mentioned earlier, the key deciding factor an ISP would consider is the false-positive rate (FPR). Every false positive results in unintentional filtering of a Web packet. For a model to be put to use, even an adversarial ISP would prefer the model to achieve high detection rate (i.e., recall) at low FPRs, i.e., FPRs of 10^{-2}, 10^{-3}, and 10^{-4}. A low FPR of 10^{-4} translates to falsely classifying (and thus filtering) 1 in 10K (encrypted) Web packet only, which we believe can be tolerated.

4. Due to the similar performance in both open-world settings, here, we only use the traces in $\mathcal{OW}_1$ for the aggregation.

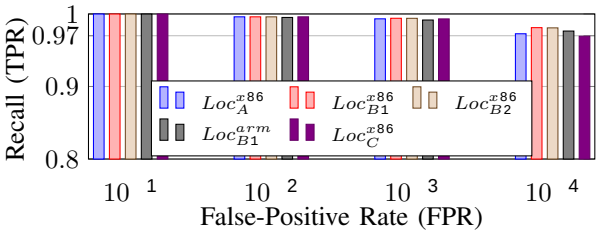


Figure 7: Recall at different FPRs, for the localized models (M_{2b}) in closed-world setting.

To this end, we first evaluate to what extent the location has an impact on the recall. Intuitively, according to the features our models consider, a model trained on one location is best suited to capture the network characteristics of that specific location. Hence, we train a model based on data from one location, and test the model at the same specific location; we refer to this as a *localized* model. However, we have data collected for all resolvers only at $LocC$; for all other locations, the common resolvers are the four prominent resolvers. Therefore, we take two steps to evaluate the localized models. We first use M_{2b} and train localized models of M_{2b} at all locations, but in a closed-world setting. Second, since we have data for all resolvers at $LocC$, we evaluate the best model M_3 in a localized way, under both closed-world and open-world settings. Finally, we also evaluate M_3 across different locations, at low FPRs.

5.4.1. Localized models based on M_{2b}. As mentioned above, for each location, we use the location-specific data to train a model M_{2b} and test it in a closed-world setting. Note, we choose M_{2b} because it is based on the four prominent resolvers that are common across all locations. Fig. 7 plots the results. We observe that localized models correctly identify more than 97% of the DoH traffic while misclassifying only 1 in 10K Web packets. See Fig. A.5a in the Appendix for a complete rundown of the recall values.

5.4.2. M_3 at $LocC$. Since at $LocC$ we have gathered data for 25 DoH resolvers, we are able to analyze the detection capability of M_3 at low FPR values in both closed- and open-world settings. Note, M_3 is trained on seven resolvers including the four prominent ones (cf. §5.3). The results (depicted in Fig. A.5b in the Appendix) are the following. We observe that for an FPR of 10^{-3} (10^{-4}) more than 99.3% (97.4%) of the DoH traffic is accurately identified in the closed-world setting. While the performance slightly drops for the open-world setting, the identification rates of DoH packets are still high; for FPRs of 10^{-3} (10^{-4}), the recall is 93% (90%).

5.4.3. M_3 at different locations. We take our best-performing model M_3 and evaluate its performance on the data gathered at locations other than $LocC$. Note, this model is trained on data from $LocC$, and the data used for testing is based on the four prominent resolvers only. We observe in Fig. 8 that even at low FPRs of 10^{-3} (10^{-4}), our model can identify more than 95% (90%) of the DoH

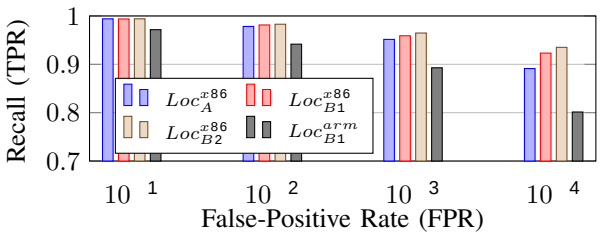


Figure 8: Recall of M_3 at different FPRs, and at different locations and environments.

traffic in x86 architectures. On the other hand, on the data gathered on arm architectures, the same F_1 scores is 90% (80). Fig. A.5b in the Appendix plots the recall values at multiple FPRs for different locations.

5.4.4. Summary. Ultimately, the results indicate that while our model is most accurate when deployed at the same location where it is trained (97.4% and 90% recall at FPR of 10^{-4} in the closed- and open-world setting, respectively), it is sufficiently robust to be used across different locations, i.e., 90% (80%) at FPR of 10^{-4} on x86 (arm) architectures. This also enables our model to be incrementally deployed, i.e., after deploying it at any location, we can gradually improve its performance by gathering data at the same location and, occasionally, re-train the model offline. Alternatively, one can also train an independent identification model for each location based on the specific location's data.

6. Countering DoH Identification w/ Padding

The results above imply that an authoritarian ISP can employ our DoH identification model to identify DoH packets accurately and deploy filtering mechanisms to force its users to fall back to traditional plain-text DNS.

In this section, we develop and evaluate multiple techniques to counter such a DoH identification model.

In other words, our goal here to make DoH traffic indeed indistinguishable from Web traffic; to achieve this privacy-enhancing vision, we investigate to what extent the detection capability of our model can be decreased.

6.1. Padding techniques

Since the features of the identification model are based on packet size and time lag between packets, manipulation of these form the crux of our techniques. This is achieved by *padding* additional bytes (time) on to the existing packet size (packet inter-arrival time). Padding has already been proposed for DoT and DoH protocols to counter traffic analysis posing [29], [51]. However, prominent implementations (e.g., Bind [42], Knot [21]) and open resolvers (e.g., Cloudflare) are still in an experimental state regarding these current padding policies, and it has been shown that such simple padding is not always a practical solution against traffic analysis [8]. Our evaluation of the padding technique in RFC 8467 [51] (defined as **PT(1)** below) also demonstrates that the proposal is ineffective in countering our DoH identification model.

Unsurprisingly, a model trained on the original non-padded dataset can be easily evaded even by naïvely padding each DoH packet with a random length (such that the packet size does not exceed the MTU); see a brief study in the Appendix A.11. An adversary, on the other hand, can also easily generate such traffic patterns and retrain her identification model using the padded dataset. Hence, for a fair and meaningful study, we need to test the padded data on a model trained on the padded data.

Therefore, we are interested in developing countering techniques that deceive even such models.

To clearly understand the effect of padding, we first train a new model using only the pkt_len feature and evaluate its performance for different padding techniques applied for all DoH packets. Subsequently, we target the manipulation of each additional feature. We consider the following padding techniques (**PT**s).

PT(0): This serves as the baseline, where no padding technique is applied on the dataset.

PT(1): We apply fixed padding according to RFC 8467 [51], which suggests each client to pad queries to the closest multiple of 128 bytes[5].

PT(2): We pad each DoH packet with a random length such that the size of the padded packet does not exceed the MTU.

PT(3): We assume the size of Web packets follow a Normal distribution and estimate its parameters (mean and standard deviation) from the Web traffic in the original dataset. For each DoH packet, we sample from this distribution a random value, say X, and pad the packet so as to set the packet size to X. If the original DoH packet size is smaller than X, we obtain a new sample from the distribution and pad accordingly.

PT(4): We continuously maintain a list of the past n consecutive Web packets; $L_n^W = [p_1^W, p_2^W, ..., p_n^W]$. For each DoH packet, we pad its size to p_r^W, the size of a randomly chosen Web packet from this list, where $r = \text{RANDOM}(1, n)$. If for a DoH packet p_k^D, there is no $p_i^W : p_i^W > p_k^D$ (i 2 $[1..n]$), we fall back to **PT(3)**.

PT(5): To increase the correlation between sizes of subsequent packets sizes, we amend **PT(4)** as follows. For a consecutive sequence of d ($d < n$) DoH packets (L_d^D), we obtain L_d^W as a sub-sequence of L_n^W, where $L_d^W = [p_r^W, p_{r+1}^W ..., p_{r+d}^W]$, and $r = \text{RANDOM}(1, n)$. Then, we pad each DoH packet p_k^D 2 $L_d^D, k = [r, ..., r+d]$ to the size of the k^{th} Web packet p_k^W 2 L_d^W. In practice, however, we do not know in advance, after transmitting a DoH packet whether the number of the subsequent DoH packets will become greater than n. Thus, when $(r + d) > n$, we take a new r and obtain $L_d^W = [p_r^W, p_{r\ 1}^W ..., p_{r\ d}^W]$ again. Furthermore, if $p_k^D > p_k^W$, i.e., when the available new size of a DoH packet is smaller than the actual one, we replace the new size with the first p_k^W 2 $L_n^W, k = \text{RANDOM}(1, n)$, such that $p_k^W < p_k^D$. Finally, similar to **PT(4)**, if $\nexists p_k^W : p_k^W > p_k^D$, we apply **PT(3)**.

For training, we consider the datasets corresponding to the four prominent resolvers only, while for testing, first, we focus on the closed-world setting as we aim to decrease the performance in the ideal case as much as possible. Then, we evaluate the performance of the final

5. Although RFC 8467 recommends the server to pad the responses to a multiple of 468 bytes, recall, our model is based only on requests.

padded model (trained on the four prominent resolvers) in the open-world setting.

6.2. Evaluating padding techniques on packet size

In this section, we focus only on the two features related to packet sizes, namely `pkt_len` and `prev_pkt_len`. First we only consider `pkt_len` as a feature for modeling, and evaluate the performance of the model trained on the original dataset as well as the padded datasets for this particular feature. Subsequently, we include the feature `prev_pkt_len` for training and testing the padding techniques described above. Note, padding the packet size inherently affects the feature `prev_pkt_len`. We refer to the models trained using feature $f_1 =$ `pkt_len` as $M^{f_1}_{\textbf{PT(i)}}$, and the one trained using features $\{f_1, f_2\} =$ {`pkt_len, prev_pkt_len`} as $M^{f_{1,2}}_{\textbf{PT(i)}}$. The subscript refers to the padding technique **PT(i)**, **i** 2 $\{0, 1, .., 5\}$, since the data used for training, and hence the model, depends on the specific technique applied for padding. A summary of the results is given in Fig. 9. Overall, the results indicate that the accuracy of the models increase when more features are considered; (by 5 10% on average). We explain more details below.

PT(0): Original data set. As baseline, we train two models, one using only `pkt_len` and another using both `pkt_len` and `prev_pkt_len` as features; both trained and evaluated on the original dataset without any padding technique applied. For the first model $M^{f_1}_{\textbf{PT(0)}}$, we observe a high F_1-score of 0.8912. The model $M^{f_{1,2}}_{\textbf{PT(0)}}$, that has both features, has an even higher F_1-score of 0.975.

The distribution of the Web and the non-padded DoH packets are plotted in Fig. 10a.

PT(1): Fixed padding. We start with the first padding technique, wherein we pad each DoH *query* to the closest multiple of 128 bytes. Not surprisingly, this approach increases the accuracy of our models regardless of whether only `pkt_len` or the compound feature set {`pkt_len, prev_pkt_len`} is considered. The F_1-score for $M^{f_1}_{\textbf{PT(1)}}$ ($M^{f_{1,2}}_{\textbf{PT(1)}}$) is 0.9993 (0.9999). The reason behind this high performance is that, after padding, much of the distribution of the DoH packet size is now reduced to 232 or 360 bytes[6], making it easy for the classification models to differentiate DoH from Web traffic.

PT(2): Full random. While this approach fully scatters the packet size values, the resulting F_1-score of $M^{f_1}_{\textbf{PT(2)}}$ ($M^{f_{1,2}}_{\textbf{PT(2)}}$) is still a high 0.9134 (0.9752). This is because, statistically, the distribution of the padded DoH packets is random, and hence not similar to the distribution of the Web packets.

PT(3): Random, based on distribution of Web packet size. Next, to make the DoH packet sizes to look similar to the Web packet sizes, we apply **PT(3)**. This results in an F_1-score of 0.868 for $M^{f_1}_{\textbf{PT(3)}}$. While this technique is better than the previous two in reducing the classification accuracy of the identification model, note that the F_1-score is still close to 90% (given we are using just one feature for classification). From Fig. 10b, we observe that

6. There is an additional overhead of 9B HTTP2 header, 28 – 32B TLS Application Data (i.e., encryption, content-type, version, length), 32B TCP, 20B IP, and 14B Ethernet (without the 4-byte checksum).

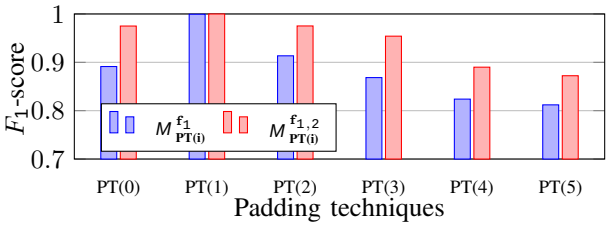


Figure 9: F_1-scores of the different padding techniques in the closed-world setting using features `pkt_len` and `prev_pkt_len`.

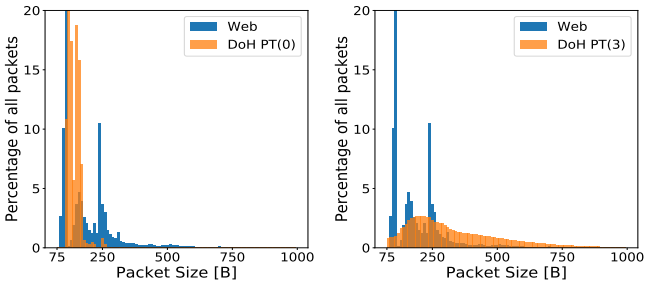
the distribution of padded DoH packet size is still different from that of Web packet size. This is also reflected in the performance of $M^{f_{1,2}}_{\textbf{PT(3)}}$; in comparison, the F_1-score of this model is at a higher value of 0.954.

PT(4): Random preceding Web packet. In order to further blend the distributions of DoH packet size and Web packet size, in **PT(4)**, we pad each DoH packet size to the size of a recent Web packet. To do this, we maintain a list of the most recent Web packets, and each DoH packet is padded to (have the size the same as) a randomly selected packet size from this list. This technique includes a parameter n to set the length of the list of the recent Web packets; therefore, we carry out experiments for different values of n in the set $\{5, 10, 20, 30\}$.

The average F_1-score (over the different values of n) of $M^{f_1}_{\textbf{PT(4)}}$ with this padding technique is 0.824 (with a negligible standard deviation of 0.0009). The resulting padded packet-size distribution is shown in Fig. 10c, where, for brevity, we show the results for $n = 20$. We observe that the distribution of the DoH packet size follows the distribution of the Web packets closely with **PT(4)**. We also note that since most of the original packet sizes of DoH traffic are around 110 170 bytes (cf. Fig. 10a), they are padded to the most frequent `pkt_len` values of the Web packets, i.e., to 240 and 310 bytes (two highest orange bars). This clearly explains why this technique provides the worst classification performance so far. However, for $M^{f_{1,2}}_{\textbf{PT(4)}}$, on the other hand, we observe a higher average F_1-score of 0.89 (with standard distribution 0.0033). This performance improvement while using the additional feature of `prev_pkt_len` is also observed with **PT(3)**; this happens because the bivariate distributions of these two features extracted from packet size are still different for the padded DoH traffic and Web traffic. The sizes of two consecutive Web packets are not independent; while in **PT(4)**, we do not consider this. To deceive the model further, we would need to ensure that consecutive DoH packets have similar size distribution as consecutive Web packets leading us to **PT(5)**.

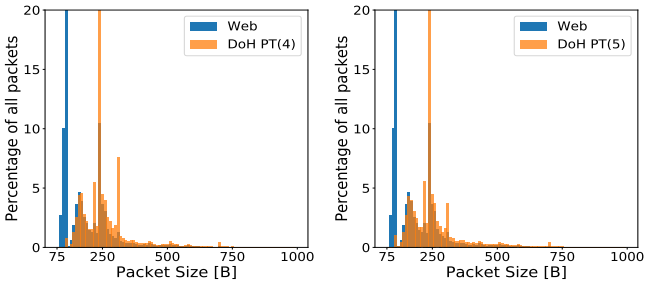
PT(5): Random sequence of the preceding Web packets. We evaluate **PT(5)**, which takes a sequence of Web packet sizes first, and pads consecutive DoH packets so that their sizes are set to the same as the consecutive Web packets. The padded distribution (for $n = 20$) is depicted in Fig. 10d. We noted a somewhat different distribution than in case of **PT(4)**, in particular, a slightly flattened distribution for packet size greater than 250 bytes. The average F_1-scores of both models $M^{f_1}_{\textbf{PT(5)}}$ and $M^{f_{1,2}}_{\textbf{PT(5)}}$ are now closer, at 0.812 and 0.8722, respectively (cf. Fig. 9).

Fig. 11a plots the Precision-Recall curves of the base-

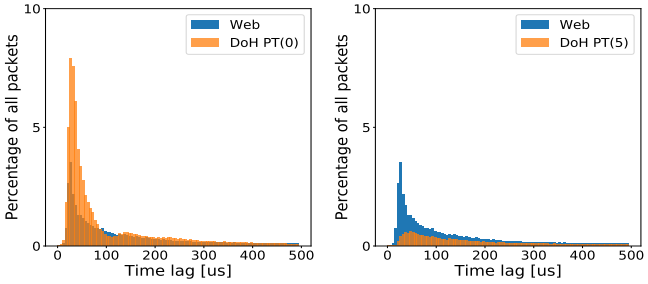


(a) PT(0) on `pkt_len`.

(b) PT(3) on `pkt_len`.



(c) PT(4) on `pkt_len`, $n = 20$. (d) PT(5) on `pkt_len`, $n = 20$.



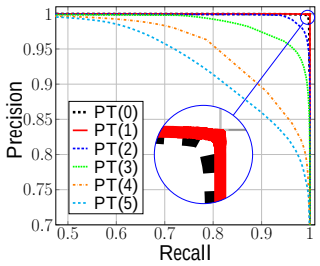
(e) PT(0) on `time_lag`. (f) PT(5) on `time_lag`, $n = 20$.

Figure 10: Distributions of packet size (bin size=10B) and time lag (bin size=5μs) of DoH packets compared to the Web packets using different padding techniques.

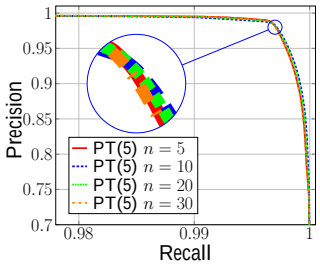
line model and all models trained and tested with the padded datasets. It clearly demonstrates the performance degradation achieved due to **PT(4)** and **PT(5)**. With the above experimental study, we conclude that the worst performance is achieved via **PT(5)** for both $M^{f_1}_{\mathbf{PT(5)}}$ and $M^{f_{1,2}}_{\mathbf{PT(5)}}$. Hereafter, for padding, we apply **PT(5)**.

6.3. Evaluating padding of all features

As we observed, applying **PT(5)** to both features related to packet size blends DoH packets the most with the Web packets making them much less distinguishable. In particular, the F_1-score of the model trained and tested on the non-padded reducedby 10% in the closed-world scenario. Next, we manipulate the other features of the model; i.e., we apply the best performing padding technique **PT(5)** to the remaining features related to the inter-arrival time of DoH packets. We train and test a new model $M^{\mathbf{f}}_{\mathbf{PT(5)}}$, where (to recall) $\mathbf{f} = f\{f_1, f_2, f_3, f_4\} = \{$`pkt_len`, `prev_pkt_len`, `time_lag`, `prev_time_lag`$\}$. We also analyze the impact of n on accuracy. Note, we evaluated all padding techniques on this new model and observed similar results as in the case of the model with the packet-size features.



(a) Based on features `pkt_len` and `prev_pkt_len`.



(b) Based on all features.

Figure 11: Precision-Recall curves for the different padding techniques on different feature sets in the closed-world setting.

Therefore, we focus only on **PT(5)** for this complete model. Similar to packet size, packet inter-arrival time can only be increased, i.e., a user might introduce extra latency but the time lag of the system is already bounded, e.g., by hardware limitations, packet processing performance.

PT(0): Original data set. The F_1-score of the baseline model $M^{\mathbf{f}}_{\mathbf{PT(0)}}$ is 0.991; and this is obtained using all four features trained and tested on the unmodified original dataset. We present the distribution of time lag for both Web and DoH traffic in Fig. 10e.

Random sequence of the preceding Web packet. The padded distribution (for $n = 20$) is depicted in Fig. 10f. We observe a much more flattened distribution of time lag for the DoH packets. The average F_1-score of $M^{\mathbf{f}}_{\mathbf{PT(5)}}$ is 0.945 (with standard deviation 0.001), which is 5% lower than the F_1-score of 0.991 achieved by the baseline model trained on the original dataset. This is also visible with the Precision-Recall curves plotted in Fig. 11b. Observe that, with the padding technique **PT(5)**, to achieve close to 100% recall, the identification model has to tolerate a limited precision of around 70%. For any useful deployment, an adversary's filtering solution would need to have very high precision. Going by this requirement, at a fixed precision of 0.9999, while the baseline model $M^{\mathbf{f}}_{\mathbf{PT(0)}}$ achieves a very high recall of 0.977, the recall of $M^{\mathbf{f}}_{\mathbf{PT(5)}}$ drastically reduced to 0.538 (not visible on Fig. 11b). This means, only around half of the DoH packets can be identified by a high-precision model. In Fig. 11b, we also plot the Precision-Recall curves for different values of n, and shows insignificant differences on classification accuracy.

Most importantly, we analyze how the padded model affects the DoH identification rate for low FPRs. Observe in Fig. 12 that, with **PT(5)**, even at a practically unacceptable FPR of 10^{-3} (wherein, 1 in 1000 Web packets would be misclassified as DoH and hence blocked), the detection capability of the identification model degrades to 83%. For even lower value of FPR required to be considered for deployment (i.e., FPR $= 10^{-4}$), the recall drops down to 0.53, thereby removing any practical use of the identification model.

Open-world experiments. After attaining the worst average F_1-score of 0.9458 with $M^{\mathbf{f}}_{\mathbf{PT(5)}}$ in the closed-world scenario, we analyze this model's performance in the open-world setting. In particular, to be on par with our evaluations in §5, we merge and pad the traces for the five resolvers from §5.2 that performed the best when tested

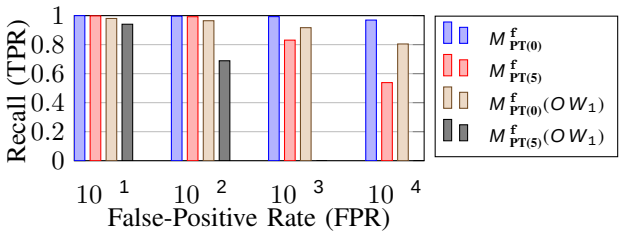
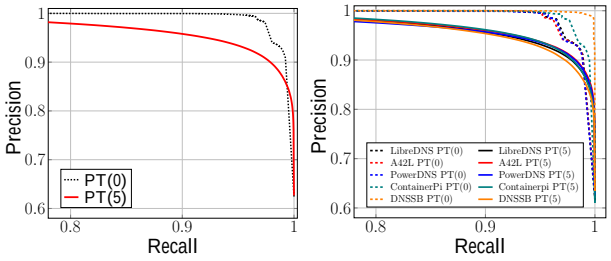


Figure 12: Recall at different FPRs, for the (non-)padded models in the closed- and open-world settings.



(a) On aggregated data. (b) On individual resolvers.

Figure 13: Precision-Recall curves for the best performing resolvers in the open-world setting on $M^{f}_{PT(0)}$ and $M^{f}_{PT(5)}$.

on M_{2b} (not M_3, as it was trained on seven resolvers instead of the prominent four ones). While $M^{f}_{PT(0)}$ tested on the merged non-padded dataset provides an F_1-score of 0.977, the model $M^{f}_{PT(5)}$ trained on the padded dataset results in an F_1-score of 0.9414.

Fig. 13a plots the PRC for the two models. Observe that, $M^{f}_{PT(0)}$ identifies almost 98% DoH packets with more than 94% precision; whereas, the precision of $M^{f}_{PT(5)}$ drops to below 90% for the same recall. This essentially means that 10% of those predicted as DoH packets are in fact Web packets; and hence a network adversary cannot rely on this model for blocking DoH traffic (10% of the time it would end up blocking Web packets). On the other hand, if one wants to use the model with a high precision setting[7], say at 99.5%, then the recall that $M^{f}_{PT(5)}$ achieves is only 60%, which is significantly lower than the recall of 0.961 that $M^{f}_{PT(0)}$ achieves for the same precision. Fig. 13b plots the Precision-Recall curves for the datasets corresponding to the individual resolvers, for both $M^{f}_{PT(0)}$ and $M^{f}_{PT(5)}$. Observe in Fig. 13b that in case of DNSSB, for instance, the recall of 99% at precision 99.3% can drop down to 65%.

Revisiting Fig. 12 for the open-world setting (OW_1), we see a reflection of the above results when analyzing the recall at different FPRs. We observe that the padded model can identify only 68.8% of the DoH traffic correctly at an FPR of 10^{-2}. For lower FPR values, the recall reduces to insignificant values (< 0.01). For a more detailed recall v/s (low) FPR plot, refer to Fig. A.6 in the Appendix.

We conclude that our **PT(5)** can downgrade the performance of the best DoH identification model to the extent that its practical use is diminished severely.

7. A threshold to classify packets gives one set of results on which we compute precision and recall. Therefore, setting a specific precision is achieved by finding the right threshold via multiple runs with different values for the threshold. This is also why we do not have a fixed precision throughout the paper.

7. Applicability

Identification model. According to our threat model (§3), we assume an in-line adversarial monitoring application, based on our model, is deployed in the network to block DoH traffic. Once a model is trained, ISPs have multiple ways to practically deploy it. Programmable data-plane is one option, considering a recent advancement in supporting ML inference at Tbps-rate [73]. Fast packet processing libraries (e.g., DPDK [44]) can be another option.

Padding-based countermeasures. According to our findings in §6, a victim can pad its own DoH traffic to deceive the model and eventually establish a connection to any DoH resolver for domain resolution. Since we need *a priori* knowledge on which packets to pad, padding techniques *must* be implemented on the victim's own system. The ideal solution would be to have the padding technique implemented as part of the DoH protocol. **PT(1)** and **PT(2)** are examples of such techniques; however, our study has revealed that they are not effective in countering the identification model(§6.2).

To beat any identification model that successfully differentiates DoH traffic from Web traffic, we argue that the DoH traffic would need to have characteristics that are very similar to Web traffic. This requires information from outside the DoH protocol. Based on our study, padding techniques requiring knowledge of recent Web packets, (e.g., **PT(5)**), should be implemented on top of a (Smart) NIC instead, leveraging libraries, e.g., XDP [41] or DPDK [44], to avoid performance penalty. Note, however, the padding technique running on the NIC has to be aware of the DoH traffic going to port 443, so that it knows which packets to pad; this can be achieved via cross-layer communication within the same system.

8. Conclusions

In this work, we studied the privacy enhancing aspects of the DoH protocol. We developed a machine-learning-based identification model that a network adversary, such as an authoritarian ISP, can use to differentiate DoH packets from encrypted Web packets (using the same HTTPS protocol) with high accuracy. We carried out a rigorous study testing the model on multiple and diverse traffic traces. To counter a censorship based on our identification model, we developed a padding technique that effectively brings down the classification accuracy such that it is not useful for practical adversarial purposes. Since DoH is still in its early stage of deployment, we hope our work triggers more studies on enhancing the privacy aspects of the protocol. As future work, we plan to have a practical in-network implementation and use further traffic traces generated via other clients at other locations.

Acknowledgement This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd.

The research has also been supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1C1C1008462).

References

[1] A. Abusnaina, R. Jang, A. Khormali, D. Nyang, and D. Mohaisen. DFD: Adversarial Learning-based Approach to Defend Against Website Fingerprinting. In *IEEE INFOCOM*, pages 2459–2468, 2020.

[2] A. Abusnaina, A. Khormali, H. Alasmary, J. Park, A. Anwar, and A. Mohaisen. Adversarial Learning Attacks on Graph-based IoT Malware Detection Systems. In *ICDCS*, pages 1296–1305, 2019.

[3] Alexa. The top 500 sites on the web. [Online], https://www.alexa.com/topsites, [Accessed: Oct 2020].

[4] J. Althouse. TLS Fingerprinting with JA3 and JA3S. Blog post, https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967, Jan 2019 [Accessed: Oct 2020].

[5] J. Althouse. Open Sourcing JA3. Blog post, https://engineering.salesforce.com/open-sourcing-ja3-92c9e53c3c41, Jul 2017 [Accessed: Oct 2020].

[6] P. Bishoff. Which Countries Have the Strictest Internet Censorship? Blogpost, https://bit.ly/2SyUVR7, Jul 2019 [Accessed: Oct 2020].

[7] H. Brown, E. Guskin, and A. Mitchell. The Role of Social Media in the Arab Uprisings. Pew Research Center Journalism & Media, https://www.journalism.org/2012/11/28/role-social-media-arab-uprisings/, Nov 2012 [Accessed: Oct 2020].

[8] J. Bushart and C. Rossow. Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS. *CoRR*, abs/1907.01317, 2019.

[9] M. Candela, V. Luconi, and A. Vecchio. Impact of the COVID-19 pandemic on the Internet latency: A large-scale study. *Computer Networks*, 2020.

[10] S. Captain. Here's how to stop your ISP from spying on you. FastCompany Blogpost, https://www.fastcompany.com/90421616/heres-how-to-stop-comcast-verizon-and-other-isps-from-spying-on-you, Oct 2019 [Accessed: Oct 2020].

[11] C. Catalin. Google to run DNS-over-HTTPS (DoH) experiment in Chrome. ZDNet blogpost, https://www.zdnet.com/article/google-to-run-dns-over-https-doh-experiment-in-chrome/, Sep 2019 [Accessed: Oct 2020].

[12] C. Cimpanu. Mozilla Is Testing "DNS over HTTPS" Support in Firefox. Blogpost, https://www.bleepingcomputer.com/news/software/mozilla-is-testing-dns-over-https-support-in-firefox/, Mar 2018 [Accessed: Oct 2020].

[13] CleanBrowsing. Articles and News, 2018 May. [Online], https://cleanbrowsing.org/articles/, [Accessed: Feb 2021].

[14] Cloudflare. What is DNS cache poisoning? — DNS spoofing. [Online], https://www.cloudflare.com/learning/dns/dns-cache-poisoning/, [Accessed: Oct 2020].

[15] Comcast. Xfinity Internet Joins Firefox's Recursive Resolver Program, Committing to Customer Privacy Protection. Press release, https://corporate.comcast.com/press/releases/comcast-xfinity-internet-firefox-trusted-recursive-resolver-program-customer-privacy, Jun 2020.

[16] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari. Client Subnet in DNS Queries. RFC 7871, Internet Engineering Task Force, May 2016.

[17] cs.lev. doh_docker. Github repository, https://github.com/cslev/doh_docker, [Accessed: Feb 2021].

[18] cslev. doh_docker. DockerHub image, https://hub.docker.com/r/cslev/doh_docker, [Accessed: Feb 2021].

[19] cs.lev. doh_ml. Github repository, https://github.com/cslev/doh_ml, [Accessed: Feb 2021].

[20] Curl. DNS over HTTPS - Publicly available servers. [Online], https://github.com/curl/curl/wiki/DNS-over-HTTPS, [Accesed: May 2020].

[21] CZ.NIC. KNOT DNS. Online, https://www.knot-dns.cz, [Accessed: Oct 2020].

[22] S. Dickinson, D. Gillmor, and T. Reddy. Usage Profiles for DNS over TLS and DNS over DTLS. RFC 8310, Internet Engineering Task Force, March 2018.

[23] DNSCrypt project. DNSCrypt version 2 protocol specification. [Online], https://dnscrypt.info/protocol/, [Accessed: Oct 2020].

[24] DNSCrypt project. Frequently Asked Questions. [Online], https://dnscrypt.info/protocol/faq, [Accessed: Oct 2020].

[25] E. Rescorla and K. Oku and N. Sullivan and C.A. Wood. TLS Encrypted Client Hello. IETF Draft, https://tools.ietf.org/html/draft-ietf-tls-esni-07, June 2020.

[26] E. Rescorla and K. Oku and N. Sullivan and C.A. Wood. Encrypted Server Name Indication for TLS 1.3. IETF Draft, https://tools.ietf.org/html/draft-ietf-tls-esni-06, March 2020.

[27] efficient iP. Why You Shouldn't Rush Into DoH. Blogpost, https://www.efficientip.com/dont-rush-into-doh/, Sep 2019 [Accessed: Oct 2020].

[28] S. Gatlan. Microsoft adds Windows 10 DNS over HTTPS settings section. BleepingComputer Blog, https://www.bleepingcomputer.com/news/security/microsoft-adds-windows-10-dns-over-https-settings-section/, Aug 2020, [Accessed: Oct 2020].

[29] D. Gillmor. Empirical DNS Padding Policy. Online, https://dns.cmrg.net/ndss2017-dprive-empirical-DNS-traffic-size.pdf, Mar 2017 [Accessed: Oct 2020].

[30] Google. HTTPS encrpytion on the web. Google Transparency Report, https://transparencyreport.google.com/https/overview?hl=en, [Accessed: Oct 2020].

[31] M. Gunes and B. Charyyev. Iot event classification based on network traffic. 02 2020.

[32] J. Hayes and G. Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security*, pages 1–17, 2016.

[33] S. Hazarika. BraveDNS is an open-source DNS-over-HTTPS client, firewall, and adblocker for Android. XDA-developers post, https://www.xda-developers.com/bravedns-open-source-dns-over-https-client-firewall-adblocker-android/, Aug 2020 [Accessed: Oct 2020].

[34] I. Herrera. How Venezuela's vice grip on the internet leaves citizens in the dark during crises. NBC News, https://www.nbcnews.com/tech/tech-news/how-venezuela-s-vice-grip-internet-leaves-citizens-dark-during-n1006146, May 2019 [Accessed: Oct 2020].

[35] D. Herrmann, C. Banse, and H. Federrath. Behavior-Based Tracking: Exploiting Characteristic Patterns in DNS Traffic. 2013.

[36] N. P. Hoang, A. Akhavan Niaki, N. Borisov, P. Gill, and M. Polychronakis. Assessing the privacy benefits of domain name encryption. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '20, page 290–304, New York, NY, USA, 2020. Association for Computing Machinery.

[37] P. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). RFC 8484, Internet Engineering Task Force, 2018.

[38] R. Houser, Z. Li, C. Cotton, and H. Wang. An Investigation on Information Leakage of DNS over TLS. In *ACM CoNEXT*, page 123–137, 2019.

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

[40] Q. Huang, D. Chang, and Z. Li. A Comprehensive Study of DNS-over-HTTPS Downgrade Attack. In *10th USENIX FOCI*. USENIX Association, Aug. 2020.

[41] IOVisor. eXpress Data Path. Online, https://www.iovisor.org/technology/xdp, [Accessed: Oct 2020].

[42] ISC. BIND 9. Online, https://www.isc.org/bind/, [Accessed: Oct 2020].

[43] ISPreview. Firefox Says – NO DNS Over HTTPS (DoH) by Default for UK. Blog post, https://www.ispreview.co.uk/index.php/2019/09/firefox-says-no-dns-over-https-doh-by-default-for-uk.html, Sep 2019.

[44] L. Jill. Data Plane Development Kit (DPDK) Further Accelerates Packet Processing Workloads, Issues Most Robust Platform Release to Date. DPDK announcement, https://www.dpdk.org/news/press/, Jun 2018 [Accessed: Oct 2020].

[45] T. Kent. TLS Fingerprinting: Rethinking Encrypted Traffic Analysis Strategies . Security Boulevard blog, https://securityboulevard.com/2019/07/tls-fingerprinting-rethinking-encrypted-traffic-analysis-strategies/, Jul 2019 [Accessed: Oct 2020].

[46] D. W. Kim and J. Zhang. You Are How You Query: Deriving Behavioral Fingerprints from DNS Traffic. In B. Thuraisingham, X. Wang, and V. Yegneswaran, editors, *Security and Privacy in Communication Networks*, pages 348–366, Cham, 2015. Springer International Publishing.

[47] H. Klein. Improved Home Network Privacy With NextDNS. Blogpost, https://helgeklein.com/blog/2020/05/improved-home-network-privacy-with-nextdns/, May 2020 [Accessed: Oct 2020].

[48] J. Livingood, A. Mayrhofer, and B. Overeinder. DPRIVE Phase 2 Requirements . Internet Draft, https://tools.ietf.org/html/draft-ietf-dprive-phase2-requirements-01, Jun 2020.

[49] S. Lundberg. SHAP. [Online], https://shap.readthedocs.io/en/latest/, 2018 [Accessed: Oct 2020].

[50] M. Erwin. Trusted Recursive Resolvers – Protecting Your Privacy with Policy and Technology. Blog post, https://blog.mozilla.org/netpolicy/2019/12/09/trusted-recursive-resolvers-protecting-your-privacy-with-policy-technology/, 2019.

[51] A. Mayrhofer. Padding Policies for Extension Mechanisms for DNS (EDNS(0)). RFC 8467, Internet Engineering Task Force, March 2018.

[52] P. Mockapetris. Domain names: Concepts and facilities. RFC 0882, Internet Engineering Task Force, November 1983.

[53] S. O'Dea. Consumer Internet data traffic worldwide by application category from 2016 to 2022. Statista Survey, https://www.statista.com/statistics/454951/mobile-data-traffic-worldwide-by-application-category/, Feb 2020 [Accessed: Oct 2020].

[54] L. O'Donnell. Microsoft Adds DNS-Over-HTTPS Support for Windows 10 Insiders. ThreatPost Blog, https://threatpost.com/microsoft-dns-over-https-windows-10/155746/, May 2020, [Accessed: Oct 2020].

[55] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website fingerprinting at internet scale. In *NDSS Symposium*, pages 1–15, 2016.

[56] S. Patil and N. Borisov. What Can You Learn from an IP? In *ANRW*, page 45–51, 2019.

[57] T. Pauly. Enable encrypted dns. WWDC 2020 video, https://developer.apple.com/videos/play/wwdc2020/10047, 2020 [Accessed: Oct 2020].

[58] R. Prakash. Build trust through better privacy. WWDC 2020 video (11:57), https://developer.apple.com/videos/play/wwdc2020/10676, 2020 [Accessed: Oct 2020].

[59] Quad9. DoH with Quad9 DNS Servers. Blog, https://www.quad9.net/doh-quad9-dns-servers/, 2019 [Accessed: Feb 2021].

[60] R. Roxana and H. Michael. Consolidation in the DNS resolver market – howmuch, how fast, how dangerous? *Journal of Cyber Policy*, pages 46–64, 02 2020.

[61] R. Schuster, V. Shmatikov, and E. Tromer. Beauty and the Burst: Remote Identification of Encrypted Video Streams. In *USENIX Security*, pages 1357–1374, Vancouver, BC, Aug. 2017. USENIX Association.

[62] scikit learn. Machine Learning in Python. [Online], https://scikit-learn.org/stable/, [Accessed: Oct 2020].

[63] scikit-learn developers. sklearn.ensemble.RandomForestClassifier. Documentation, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2020 [Accessed: Feb 2021].

[64] Selena Deckelmann. Firefox continues push to bring DNS over HTTPS by default for US users. Mozilla Blog, https://blog.mozilla.org/blog/2020/02/25/firefox-continues-push-to-bring-dns-over-https-by-default-for-us-users/, Feb 2020.

[65] S. Siby, M. Juárez, C. Díaz, N. Vallina-Rodriguez, and C. Troncoso. Encrypted DNS -> Privacy? A Traffic Analysis Perspective. In *NDSS Symposium*, 2020.

[66] P. Sirinam, M. Imani, M. Juarez, and M. Wright. Deep fingerprinting: Undermining web-site fingerprinting defenses with deep learning. In *ACM CCS*, pages 1928–1943, 2018.

[67] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy. DEFT: A Distributed IoT Fingerprinting Technique. *IEEE Internet of Things Journal*, 6(1):940–952, Feb 2019.

[68] The OneLab consortium. OneLab Future Internet Testbeds. Online, https://onelab.eu/services, [Accessed: Oct 2020].

[69] D. Vekshin, K. Hynek, and T. Cejka. DoH Insight: Detecting DNS over HTTPS by Machine Learning. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, 2020.

[70] T. Wang and I. Goldberg. On realistically attacking torwith website fingerprinting. In *PETS*, pages 21–36, 2016.

[71] S. Weiler and D. Blacka. Clarifications and Implementation Notes for DNS Security (DNSSEC). RFC 6840, Internet Engineering Task Force, February 2013.

[72] C. V. Wright, S. E. Coull, and F. Monrose. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *NDSS*, 2017.

[73] Z. Xiong and N. Zilberman. Do switches dream of machine learning? toward in-network classification. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, HotNets '19, page 25–33, New York, NY, USA, 2019. Association for Computing Machinery.

Appendix A.
Further details on datasets, features and evaluations

A.1. Most popular resolvers

We pick the most prominent resolvers according to their penetration; e.g., Cloudflare and Google are defaults for Firefox and Google Chrome, respectively, providing DoH services since the beginning. Besides, `Cleanbrowsing` and `Quad9` are further well-known resolvers providing reliable DoH services since 2018 and 2019, respectively [13], [59]. We also observed that the most number of successful DNS resolutions and website visits were achieved when using these four resolvers. In particular, on average, the dataset corresponding to each of these resolvers has $2M$ packets, while other resolvers result in around $1.2M$ packets $300k$.

A.2. Mislabeled Web Packets

We identify that not all HTTPS packets destined to a DoH resolver is labeled as DoH by Wireshark. For instance, before sending the actual encrypted DNS query (labeled as DoH), there is always a preceding HTTPS POST message labeled as Web (cf. Fig. A.1 in the Appendix). Therefore, whenever a packet within a flow is labeled as DoH, we relabel every packet of the same flow as DoH.

```
104.16.248.249    HTTP2    114 HEADERS[121]: POST /dns-query
104.16.248.249    DoH      136 Standard query 0x0000 A www.google.com OPT
104.16.248.249    HTTP2    110 HEADERS[123]: POST /dns-query
104.16.248.249    DoH      136 Standard query 0x0000 AAAA www.google.com OPT
74.125.200.99     HTTP2    236 Magic, SETTINGS[0], WINDOW_UPDATE[0], PRIORITY[3],
74.125.200.99     HTTP2    404 HEADERS[15]: GET /pagead/lvz?evtid=AKB78ciq5Xx_R5A
```

Figure A.1: Mislabeled HTTPS POST queries preceding the DoH requests sent to Cloudflare's DoH resolver.

A.3. Model Hyper-parameters

All our models are based on the Random Forest classifier. The number of trees is set to 300. For all other parameters, we used the default values set by the Python `scikit-learn` library. The most relevant hyper-parameters are listed in Table A.1, while the remaining default settings can be found in [63].

Parameter	Value
#trees	300
criterion (quality of the split)	entropy
max depth	not defined
min samples to split a node	2
min samples to be at a leaf	1
max leaf nodes	unlimited

TABLE A.1: Hyper-parameters of our models.

A.4. k-fold Cross-Validation

We carry out k-fold cross-validation for each of our models in a closed-world setting. Note, cross-validation is not meaningful in the open-world settings, since training and testing need to be carried out on different datasets (§4.4). We choose $k = 5$ for the evaluations. In scenario S_1 (§5.1), we find the average F_1-scores (with standard deviations) for $M_{1\ \texttt{CF}}$, $M_{1\ \texttt{G}}$, $M_{1\ \texttt{CB}}$, and $M_{1\ \texttt{Q9}}$ are 0.9967 (0.00013), 0.9882 (0.00085), 0.9994 (0.00005), and 0.9954 (0.00120).

In scenario S_2 (§5.2), k-fold cross-validation of the model M_{2b} resulted in an average F_1-score of 0.9897 with standard deviation 0.00432.

Our final model M_3 in scenario S_3 (§5.3) has an average F_1-score of 0.9833 with standard deviation 0.01322.

A.5. Feature preferences

In Fig. A.2, the SHAP values of the features of our model M_{2a} can be seen (cf. §5.2). After adjusting the dataset used for training (cf. §5.2, the SHAP values of our final model M_3 is depicted in Fig. A.3.

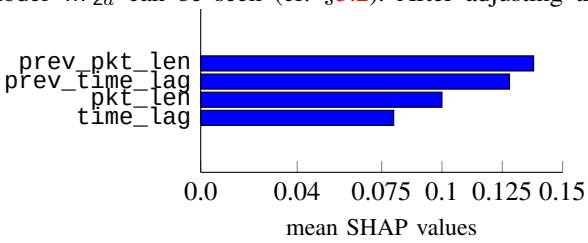


Figure A.2: The average impact of the features on the output magnitude of M_{2a}.

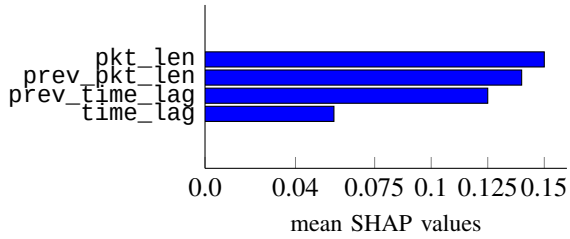


Figure A.3: The average impact of the features on the output magnitude of M_3.

A.6. Feature value statistics for model M_{2a}

Here, we show that after analyzing the false positives and false negatives, we identified that most of them belong to the subset of Web requests and DoH responses as having feature values more similar than the other two subsets, i.e., DoH requests and Web responses. Consider Fig. A.4, where the statistics of the four features are shown for DoH and Web requests and responses. We see that for the `pkt_len` and `prev_pkt_len` features, the statistical values of DoH responses and Web requests are much closer to each other than any other two relevant types, such as DoH requests compared to Web requests. In case of the feature `time_lag` (and the more important `prev_time_lag` feature), on the other hand, we can observe DoH requests and Web requests are much easier to differentiate than the corresponding responses. Therefore, as stated in §5.2, we removed the responses from the dataset when training model M_{2b} (§5.2), our final model M_3 (§5.3), as well as for the rest of our study, e.g., localized models and the countering part in §5.4 and §6, respectively. Table A.2 provides a brief summary about

how the models evolved and which packet direction we considered in the datasets for training and testing.

A.7. Detailed Accuracy Metrics

Here, we summarize all performance metrics of each models in all scenarios (cf. §5) mentioned in Sec. 4.4. First, we show the performance of the naïve sub-models, M_{2b}, and M_3 in scenario S_1 (§5.1), S_2 (§5.2), and S_3 (Sec. 5.3), respectively, in the closed-world setting (see Table A.3). Note, in case of S_3, the resolvers used for training are `Cloudflare`, `Google`, `Cleanbrowsing`, `Quad9`, `OpenDNS`, `doh.li`, `jcdns`, `PowerDNS`, and `CZNIC`.

Subsequently, we present separately the results for the same models in the open-world setting, i.e., the results for the naïve sub-models, M_{2b}, and M_3 are shown in Table A.4, Table A.5, and Table A.6, respectively

The performance of M_{2b} in the open-world setting is detailed in Table A.5. Each row corresponds to a DoH resolver not used for training.

The accuracy metrics of M_3 in the open-world setting is detailed in Table A.6. Similar to Table A.5, each row represents the accuracy metrics for each different DoH resolver not used for training. The ratios of the DoH and Web packets in the corresponding trace are also shown.

Lastly, Table A.7 summarizes the accuracy metrics of the localized models (§5.4.1) trained at different locations and environments and evaluated in a closed-world setting.

A.8. DoH resolver and Web Service Behind the Same IP address

Next, we briefly investigate whether our model's features help to distinguish DoH from Web when both the DoH resolver and the visited domain reside behind the same IP address. To this end, we used our final model M_3 and tested it (in an open-world setting) on data where the resolver used is LibreDNS (https://doh.libredns.gr) and the visited domain is its website (https://doh.libredns.gr); note, both websites are hosted behind `116.202.176.26` at the time and location of writing). Our experiment confirms that even though the `time_lag` related features are more similar, the most preferred `pkt_len` feature is still significantly different. As a result, M_3 successfully identifies LibreDNS DoH from Web traffic with a high F_1-score of 0.9895, with recall and precision of 0.9793 and 1.0, respectively.

A.9. Recall of the Models at Different FPRs

Here, we give a more comprehensive overview of the recall of the localized models M_{2b} (cf. Fig. A.5a) and the final model M_3 in different settings (cf. Fig. A.5b) as discussed in §5.4.

A.10. Recall of Padded Models at Different FPRs

A give brief summary of the recall values of the padded model $M^f_{\mathbf{PT(5)}}$ for the best performing resolvers in the open-world setting is shown in Fig. A.6.

A.11. Countering the model with padded traces

We evaluate how a model trained on the original unpadded data performs on a padded data. For this purpose, we consider our final identification model M_3. For testing, we consider most of the traces used explicitly in the open-world setting in scenario S_3 (§5.3). We apply **PT(2)**, i.e., random padding, (cf. §6.2) on all traces to be tested. The F_1-scores of our model is presented in Fig. A.7; mind the scale of the y axis. We observe that modifying the packet length of the DoH traffic by a simple padding already counters the identification model trained on non-padded data; the model's performance barely reaches an average F_1-score of 0.09811.

A.12. DoH Domain Resolution by Using Different Cloudflare IP addresses

The below simple script (relying on the command-line utility `cURL`) can be used to show that any Cloudflare IP within the `104.16/16` network prefix replies to a DoH query having the URI of `https://mozilla.cloudflare-dns.com/dns-query`. The `--resolve` command line argument requires the following pattern: `domain:port:exact IP to use`. Output of the `cURL` commands are suppressed, and if `Status:0` is received, the lookup was successful. This is checked via `grep`.

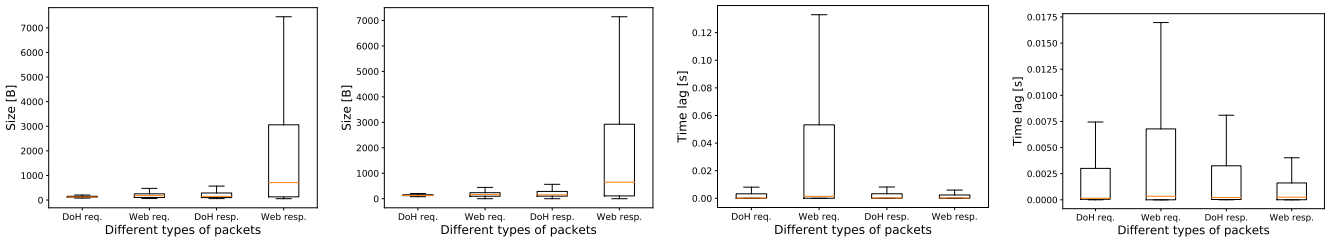
```
resolved=0
unresolved=0
for i in {1..255}
  do
    for j in {1..255}
    do
      echo -e "Testing IP 104.16.${i}.${j}..."
      curl -H 'accept: application/dns-json' --resolve
     mozilla.cloudflare-dns.com:443:104.16.${i}.${j} '
     https://mozilla.cloudflare-dns.com/dns-query?name=
     google.com&type=A' 2>&1 |grep "\"Status\":0" -q
      if [ $? -eq 0 ]
      then
        resolved=`expr $resolved + 1`
        echo "[SUCCESS]"
      else
        unresolved=`expr $unresolved + 1`
        echo "[FAIL]"
      fi
    done
  done
echo "Resolved: ${resolved}"
echo "Unresolved: ${unresolved}"

----------------- OUTPUT -----------------
...
Testing IP 104.16.255.251...[SUCCESS]
Testing IP 104.16.255.252...[SUCCESS]
Testing IP 104.16.255.253...[SUCCESS]
Testing IP 104.16.255.254...[SUCCESS]
Testing IP 104.16.255.255...[SUCCESS]
Resolved: 65536
Unresolved: 0
```

Listing 1: Simple BASH script to issue DoH queries (for the `A` record(s) of `google.com`) to all Cloudflare IPs within the `104.16/16` network prefix

A.13. Location-specific details

Loc_A^{x86} in South America is at the University of Campinas, Campinas, Brazil. We would like to thank to Fabricio Rodriguez and Christian Esteve Rothenberg for providing



(a) Statistics of `pkt_len` (b) Statistics of `prev_pkt_len` (c) Statistics of `time_lag` (d) Statistics of `prev_time_lag`

Figure A.4: Box plot for all features, for DoH and Web requests and responses.

Model	Resolvers used for training	Traffic direction
$\mathcal{M}_{1-CF}$	`Cloudflare` (`CF`)	bi-directional traffic
$\mathcal{M}_{1-G}$	`Google` (`G`)	bi-directional traffic
$\mathcal{M}_{1-CB}$	`Cleanbrowsing` (`CB`)	bi-directional traffic
$\mathcal{M}_{1-Q9}$	(`Quad9`)	bi-directional traffic
$\mathcal{M}_{2a}$	`CF`, `G`, `CB`, `Quad9`	bi-directional traffic
$\mathcal{M}_{2b}$	`CF`, `G`, `CB`, `Quad9`	requests only
$\mathcal{M}_{3}$	`CF`, `G`, `CB`, `Quad9`, `Comcast`, `OpenDNS`, `doh.li`	requests only

TABLE A.2: All models with the corresponding training datasets. Model M_3 is our final model.

Scen.	Model	Accuracy	Precision	F_1-score	Recall (TPR)	Sensitivity (TNR)	FPR	FNR	DoH:Web
S_1	$\mathcal{M}_{1-CF}$	0.993545	0.990489	0.994004	0.997545	0.988918	0.011082	0.002455	56%:43%
	$\mathcal{M}_{1-G}$	0.979515	0.983556	0.984114	0.984673	0.970170	0.029830	0.015327	48%:51%
	$\mathcal{M}_{1-CB}$	0.995418	0.994467	0.995726	0.996988	0.993609	0.006391	0.003012	58%:41%
	$\mathcal{M}_{1-Q9}$	0.983817	0.982893	0.983997	0.985103	0.982504	0.017496	0.014897	60%:40%
S_2	$\mathcal{M}_{2a}$	0.974066	0.974756	0.977001	0.979257	0.967392	0.032608	0.020743	54%:46%
	$\mathcal{M}_{2b}$	0.988836	0.993109	0.991096	0.989091	0.988404	0.011596	0.010909	54%:46%
S_3	$\mathcal{M}_{3}$	0.988966	0.993551	0.990764	0.987991	0.990422	0.009578	0.012009	54%:46%

TABLE A.3: All accuracy metrics (with the ratios of the DoH and the Web packets in the corresponding traces) in the closed-world setting of the naïve sub-models, M_{2b}, and M_3 in scenario S_1 (§5.1), S_2 (§5.2), and S_3 (Sec. 5.3), respectively.

Sub-model	Tested on	Accuracy	Precision	F_1-score	Recall (TPR)	Sensitivity (TNR)	FPR	FNR
$\mathcal{M}_{1-CF}$	Google	0.354911	0.416642	0.008831	0.004463	0.988699	0.011301	0.995537
	Cleanbrowsing	0.462329	0.258708	0.006334	0.003206	0.989453	0.010547	0.996794
	Quad9	0.493292	0.431161	0.017003	0.008673	0.988312	0.011688	0.991327
$\mathcal{M}_{1-G}$	Cloudflare	0.508443	0.812389	0.190418	0.107848	0.971227	0.028773	0.892152
	Cleanbrowsing	0.496063	0.753890	0.152500	0.783668	0.968205	0.031795	0.915170
	Quad9	0.816752	0.971182	0.783668	0.656846	0.980091	0.019909	0.343154
$\mathcal{M}_{1-CB}$	Cloudflare	0.462416	0.330897	0.005680	0.002864	0.993309	0.006691	0.997136
	Google	0.358208	0.652218	0.014128	0.007141	0.993113	0.006887	0.997136
	Quad9	0.544040	0.938696	0.188037	0.104484	0.993030	0.006970	0.895516
$\mathcal{M}_{1-Q9}$	Cloudflare	0.457154	0.162250	0.006009	0.003061	0.981740	0.018260	0.996939
	Google	0.542815	0.966403	0.458405	0.300464	0.981109	0.018891	0.699536
	Cleanbrowsing	0.507649	0.850443	0.171929	0.095631	0.980692	0.019308	0.904369

TABLE A.4: All accuracy metrics of the naïve sub-models in Sec. 5.1 in the open-world setting. The sub-models CF, G, CB, and Q9 are trained on Cloudflare, Google, Cleanbrowsing, and Quad9, respectively, and are cross-evaluated (Column "Tested on").

Tested on	Accuracy	Precision	F_1-score	Recall (TPR)	Sensitivity (TNR)	FPR	FNR	DoH:Web
PowerDNS	0.966496	0.994984	0.972640	0.951278	0.991972	0.008028	0.048722	53%:47%
Doh.li	0.585120	0.016971	0.000076	0.000038	0.998433	0.001567	0.999962	41%:59%
Comcast	0.943168	0.994552	0.945801	0.901607	0.993964	0.006036	0.098393	46%:54%
DNS.SB	0.988495	0.996030	0.990890	0.985802	0.993173	0.006827	0.014198	60%:40%
Flatuslifir	0.959292	0.996679	0.966000	0.937153	0.994969	0.005031	0.062847	65%:35%
LibreDNS	0.966144	0.996400	0.971795	0.948375	0.994527	0.005473	0.062847	60%:40%
OpenDNS	0.454426	0.913205	0.180734	0.100291	0.985700	0.014300	0.899709	58%:42%
CZNIC	0.958253	0.995945	0.963274	0.932677	0.9946030	0.005397	0.067323	27%:73%
42L	0.965119	0.996227	0.971909	0.948751	0.993721	0.006279	0.051249	50%:50%
ContainerPI	0.971233	0.995338	0.975969	0.957340	0.992981	0.007019	0.042660	56%:44%

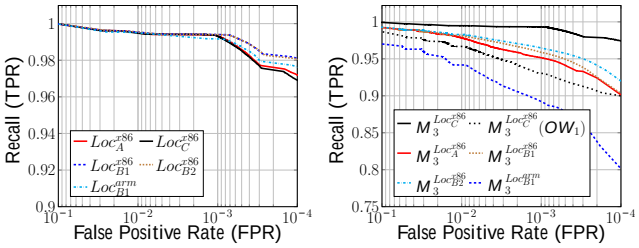
TABLE A.5: All accuracy metrics (with the ratios of the DoH and the Web packets in the corresponding traces) of model M_{2b} of scenario S_2 (§5.2) in the open-world setting OW_1.

Tested on	Accuracy	Precision	F_1-score	Recall (TPR)	Sensitivity (TNR)	FPR	FNR	DoH:Web
Adguard	0.960474	0.997292	0.969459	0.943138	0.994913	0.005087	0.056862	65%:35%
DNS.SB	0.984806	0.996218	0.987931	0.979781	0.993537	0.006463	0.020219	60%:40%
42L	0.970015	0.996775	0.975934	0.955947	0.994596	0.005404	0.044053	50%:50%
ContainerPI	0.977204	0.995659	0.981047	0.966858	0.993401	0.006599	0.033142	56%:44%
Flatuslifir	0.967963	0.996910	0.973430	0.951030	0.995250	0.004750	0.048970	65%:35%
LibreDNS	0.968576	0.996586	0.973869	0.952165	0.994790	0.005210	0.047835	60%:40%
AA	0.976634	0.996840	0.981206	0.966054	0.994755	0.005245	0.033946	60%:40%
BlahDNS	0.960832	0.996955	0.969460	0.943442	0.994432	0.005568	0.056558	60%:40%
DigitalG	0.968555	0.996763	0.973744	0.951765	0.995111	0.004889	0.048235	58%:42%
Dnslify	0.982284	0.992390	0.985627	0.978955	0.987726	0.012274	0.021045	60%:40%
FFmuc	0.973949	0.996749	0.977952	0.959851	0.995266	0.004734	0.040149	58%:42%
HE.NET	0.981434	0.996985	0.985226	0.973741	0.994860	0.005140	0.026259	62%:38%
Pi-DNS	0.947179	0.997111	0.957081	0.920140	0.995260	0.004740	0.079860	54%:47%
Tiarap	0.973956	0.995697	0.980986	0.966703	0.990482	0.009518	0.033297	65%:35%
TWNIC	0.978066	0.996416	0.981861	0.967725	0.994477	0.005523	0.032275	57%:43%

TABLE A.6: All accuracy metrics (with the ratios of the DoH and the Web packets in the corresponding traces) of model M_3 of scenario S_3 (§5.3) in the open-world setting OW_1.

Tested on	Accuracy	Precision	F_1-score	Recall (TPR)	Sensitivity (TNR)	FPR	FNR
Loc_A^{x86}	0.989162	0.993570	0.991514	0.989467	0.988621	0.011379	0.010533
Loc_{B1}^{x86}	0.990984	0.995855	0.993048	0.990257	0.992336	0.007664	0.009743
Loc_{B2}^{x86}	0.991154	0.995790	0.993095	0.990414	0.992482	0.007518	0.009586
Loc_{B1}^{arm}	0.988902	0.995458	0.991409	0.987393	0.991686	0.008314	0.012607
Loc_C^{x86}	0.988836	0.993109	0.991096	0.989091	0.988404	0.011596	0.010909

TABLE A.7: All accuracy metrics of the localized model M_{2b} trained at different locations and environments in the closed-world setting. Accordingly, the results for $LocC$ is the same as row M_{2b} in Table A.3.



(a) Localized $\mathcal{M}_{2b}$ models. (b) Final model $\mathcal{M}_3$.

Figure A.5: Complete rundown of the recall values of the final and the localized models for considerably low FPR values.

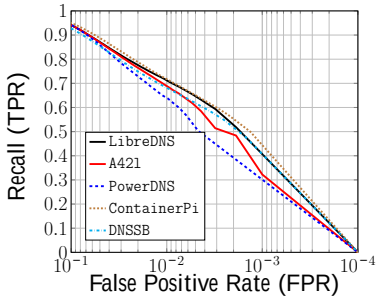


Figure A.6: Complete rundown of the recall values of model $M^f_{\mathbf{PT(5)}}$ for considerably low FPR values.

access to their facility. Loc_{B1}^{x86} and Loc_{B2}^{x86} are two anchors of Cloudlabs; while the former is at the University of Utah (western side), the latter is at the University of Wisconsin (eastern side). Loc_{B1}^{arm} is the publicly available ARM64-based facility of Cloudlabs at the University of Utah. Loc_C^{x86} is at the National University of Singapore, where the authors carried out their research.

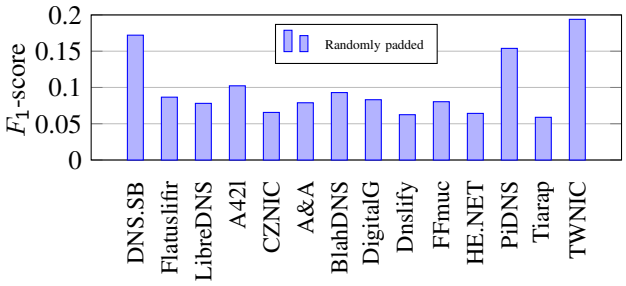


Figure A.7: F_1-score of M_3 for the resolvers from the open-world setting OW_1 in S_3, when each DoH packet is padded using **PT(2).**

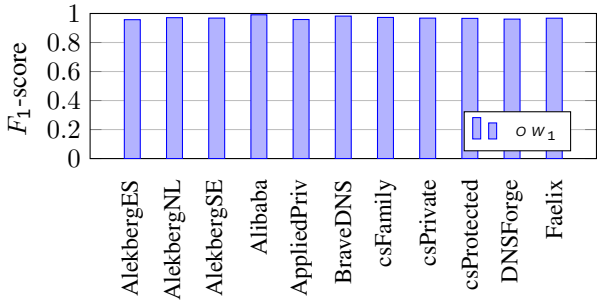


Figure A.8: F_1-score of M_3 in the open-world settings, considering eleven more DoH resolvers (published recently [20].)

A.14. F_1-score of M_3 on recent resolvers

Very recently, some new resolvers have been published in [20]. Therefore, we run a brief experiment with our final

model M_3 on these resolvers in an open-world setting, i.e., in OW_1. We observe an average F_1-score of 0.97 with standard deviation of 0.01, which renders our final model still efficient in DoH identification (cf. Fig. A.8).