Privacy Analysis of Oblivious DNS over HTTPS: A Website Fingerprinting Study

# Privacy Analysis of Oblivious DNS over HTTPS: a Website Fingerprinting Study

Mohammad Amir Salari
*Computer Science Department*
*Saint Louis University*
St. Louis, USA
mohammadamir.salari@slu.edu

Abhinav Kumar
*Computer Science Department*
*Saint Louis University*
St. Louis, USA
abhinav.kumar@slu.edu

Federico Rinaudi
*Department of Control & Computer Engineering*
*Politecnico di Torino*
Turin, Italy
federico.rinaudi@polito.it

Reza Tourani
*Computer Science Department*
*Saint Louis University*
St. Louis, USA
reza.tourani@slu.edu

Alessio Sacco
*Department of Control & Computer Engineering*
*Politecnico di Torino*
Turin, Italy
alessio_sacco@polito.it

Flavio Esposito
*Computer Science Department*
*Saint Louis University*
St. Louis, USA
flavio.esposito@slu.edu

***Abstract*—As our digital presence expands, safeguarding private data and preserving online privacy becomes paramount. Thus, motivating the development of secure DNS systems, such as DNS over TLS or HTTPS. The vulnerability of these protocols against privacy attacks has led to the development of the Oblivious DNS-over-HTTPS (ODoH) protocol. Nevertheless, the extent of ODoH's effectiveness in protecting clients' privacy is still unknown. This study investigates ODoH resiliency against website fingerprinting attacks in the open-world setting. We deploy an ODoH testbed on GENI for data collection and employ deep learning techniques such as ensemble learning for data analysis. Our findings reveal that a passive adversary can identify targeted websites using ODoH traces with an accuracy of 94%. Additionally, we analyze the impact of various factors, including clients' locations, available resolvers, and time stability, on the attack's success. Finally, we prototype a mitigation strategy and demonstrate its effectiveness in safeguarding clients privacy.**

***Index Terms*—DNS privacy, traffic analysis, website fingerprinting, deep learning.**

## 1. Introduction

The increasing reliance of users on the Internet for various daily activities, ranging from social media and video streaming to online financial services, has led to a significant rise in security and privacy threats. Among these threats, Internet surveillance and monitoring of users' online activities are the most prevalent privacy concerns [1]. Website fingerprinting (WF) is one form of surveillance in which an adversary inspects packet content or communication traffic metadata, such as packet sizes and latency, to identify the websites visited by the victim [2]–[8]. Previous studies have shown the feasibility of WF attacks targeting various protocols, such as HTTPS, Tor, and VPN [5], [9]–[15]. Although these techniques offer valuable insights, they often focus on specific aspects of user interactions or communication, *e.g.,* Tor monitoring typically focuses on the entry and exit points of the Tor network. Lately, DNS-based WF has gained more attention [7], [8], [16], in part due to its crucial role in online activities and the importance of secure DNS protocols [17]–[19] in protecting privacy.

However, DNS-based WF is challenging because DNS packets are generally smaller in size and fewer in number, primarily provide coarse-level information about domain names and IP addresses, and lack the specificity and granularity inherent in HTTPS analysis. Nonetheless, compared to web-based WF, using DNS traffic offers distinct advantages, including earlier identification of the target website (before HTTPS traffic transmission) with higher resource efficiency and lower computational complexity. While DNS-based website fingerprinting is a widely used technique for censorship enforcement [20], [21], it can also aid circumvention. In particular, early detection of blocked content allows censorship-circumvention systems to dynamically adjust relay selection, enhancing resilience against evolving censorship tactics. To detect censorship, systems like Tor periodically test connectivity to known websites and bridges, identifying interference when DNS resolution fails or responses are inconsistent. Upon detecting such signals, pluggable transport tools such as Snowflake [22] can activate their decentralized proxies to reroute traffic, ensuring continued access while bypassing filters. By responding proactively to failed DNS resolutions and tampered responses, circumvention tools can refine routing strategies, optimize network resources, and minimize delays. Additionally, DNS-based WF can be used to blend DNS queries of blocked websites with legitimate ones. Thus, enhancing the effectiveness of censorship circumvention and hindering detection and filtering by systems that focus on blocking DNS queries or initial connections.

These advantages, combined with the scarcity of DNS traffic, are the primary driving forces behind the rise of successful deep learning-based WF efforts, particularly against the DNS-over-TLS (DoT) [17] and DNS-over-HTTPS (DoH) [18] protocols [16], [23]. However, the privacy limitations of the DoT and DoH protocols have led to the development of the Oblivious DNS over HTTPS (ODoH) protocol [24] (RFC 9230), which aims to enhance anonymity by decoupling users'

identities from their DNS queries. Although ODoH offers stronger privacy guarantees, its resilience to privacy attacks has yet to be investigated.

To fill this knowledge gap, we aim to study the resiliency of the ODoH protocol against WF attacks and further investigate the impact of the spatiotemporal diversity of ODoH queries on the effectiveness of such attacks. To achieve this, we need to address the following challenges: *(i)* Given the scarcity and homogeneity of ODoH traffic compared to HTTPS traffic, what are the most effective methods for using ODoH traffic in WF? *(ii)* How can a sufficiently diverse ODoH dataset be constructed, given the lack of publicly available datasets and limited adoption and support from common browsers? *(iii)* How can the uncertainty in an open-world experimental setting, where many visited websites fall outside the target list, be effectively addressed? *(iv)* How can an effective defense mechanism against WF attacks be designed to meet the stringent latency requirements of DNS systems?

To address these challenges, we first collect a diverse ODoH dataset from geographically dispersed clients over a two-month period using the GENI (Global Environment for Network Innovations) infrastructure [25], which is a network testbed infrastructure designed to support experimental research in network and distributed systems. To the best of our knowledge, this is the first publicly available ODoH dataset, collected to facilitate and encourage further data-driven research on DNS security and privacy. To comprehensively analyze the limited and uniform ODoH traces, we employ advanced deep learning architectures, including convolutional and gated recurrent neural networks, to capture both spatial and temporal relationships within traffic flows. Finally, we introduce a novel early exit ensemble neural network for WF in the open-world setting. The ensemble model combines individual neural networks and incorporates an early exit module, enabling early inference termination for non-target samples. Thus, reducing computational resources while maintaining model performance. Our extensive evaluation of over 100 target websites demonstrates the practicality of WF against the ODoH protocol, achieving 94% accuracy in identifying target websites. Our analysis highlights the effectiveness of our early exit ensemble model in identifying non-target websites in an open-world setting. Finally, the results suggest that the location diversity of ODoH clients improves attack's performance, while frequent updates to website content adversely impact attack accuracy. We also introduce WFSafe TLS, a defense mechanism for effective mitigation of website fingerprinting attacks on the ODoH protocol. WFSafe TLS uses techniques such as padding, fragmentation, and dummy packet injection, to obfuscate traffic patterns while maintaining protocol compatibility and ensuring minimal performance impact.

**Contributions:** The contributions of this paper include the following:

**(i)** We compile a comprehensive ODoH traffic dataset by deploying a testbed on the GENI infrastructure[1]. Over a two-month period, we collected ODoH traffic traces from various locations using three well-known resolvers. This dataset represents the first ODoH traffic trace, providing a valuable resource for field researchers and practitioners.

**(ii)** We demonstrate the effectiveness of deep learning-based WF attacks against the ODoH protocol. Additionally, we present the design and development of an early exit ensemble neural network architecture tailored for the open-world experimental setting, enabling the early elimination of non-target websites. Although ODoH offers enhanced security features compared to existing secure DNS protocols, our findings reveal a vulnerability that could impact clients' privacy.

**(iii)** We assess the influence of various factors, such as geographical location, DNS resolver providers, and time stability, on the effectiveness of WF. Our results indicate that time stability has a marginal effect on WF performance, while variations in clients' geographical locations exert a more pronounced influence.

**(iv)** We prototype and evaluate WFSafe TLS, a practical defense solution that integrates padding, fragmentation, and dummy packets for traffic pattern obfuscation. This effective mechanism not only mitigates the risk of website fingerprinting attacks but also maintains compatibility and ensures minimal impact on performance.

The paper is organized as follows: Section 2 reviews the background and related work. Section 3 includes the threat model. Section 4 includes challenges, data collection, and preprocessing. We describe the evaluation methodology and present the results in Section 5. We further propose a defensive mechanism in Section 6. and conclude our work in Section 7.

## 2. Background and Related Work

This section includes the state-of-the-art website fingerprinting techniques and the evolution of secure DNS protocols, including *Oblivious DNS over HTTPS*.

### A. Website Fingerprinting

In website fingerprinting (WF), the attacker aims to infer the visited website by extracting and identifying patterns from encrypted traffic, using features such as packet sizes and inter-packet latency. WF has been applied in various domains, including HTTPS [2], [3], [12], Tor [4]–[6], [9], VPNs [10], [11], and encrypted DNS protocols [8], [16], [26], [27]. Over the years, WF techniques have evolved from statistical modeling [2], [28] and traditional machine learning [3], [10], [29], [30] to advanced deep learning-based methods [5], [6], [9], [31]–[35], aiming to achieve higher detection accuracy.

Most statistical modeling-based approaches are based on web object sizes [2], [28]. However, the use of persistent connections and pipelining in modern HTTP protocols undermines the efficacy of these solutions. This has led to the development of techniques based on traditional machine learning algorithms, such as support vector machines [29], k-nearest neighbors [10], decision trees [3], [30], and naïve Bayes [36]. These techniques outperform statistical modeling approaches by utilizing additional features, such as traffic flow direction

[1]The dataset is available at: https://doi.org/10.5281/zenodo.15115716

and the number of packets. The success of deep learning models in extracting hidden features and identifying temporal traffic patterns has made them an attractive choice for WF. Previous works have used a variety of models, including long-short-term memory (LSTM) networks [32], gated recurrent units (GRU) [34], fully connected neural networks [31], autoencoders [9], and convolutional neural networks (CNNs) [5], [6]. Hybrid approaches [35] combine models, such as CNNs and k-nearest neighbors, to extract hidden features for website classification. Other works have employed RNNs and CNNs to learn both the temporal characteristics and spatial features of network traffic [34]. Ensemble learning has also been explored [33] to reduce the variance of deep learning models and improve generalization to unseen data.

### B. Secure DNS Evolution

DNSSEC [37] was proposed to mitigate DNS spoofing attacks through digital signatures on DNS responses, but does not protect their confidentiality. DNScrypt [38] overcomes this limitation by improving DNS confidentiality through encryption of traffic between users and resolvers, preventing eavesdropping, tampering, spoofing, and cache poisoning attacks. DNS-over-TLS (DoT) [17] emerged as an alternative to DNScrypt, providing confidentiality, integrity, and data authenticity for DNS communication using long-term transport layer security between the user and the resolver. However, DoT has limitations in concealing encrypted DNS traffic, allowing passive adversaries to distinguish it from other network data, potentially leading to DNS-based censorship and traffic monitoring. DNS-over-HTTPS (DoH) [18] overcomes this limitation by disguising DNS packets inside HTTPS traffic. DoH also benefits from the existing infrastructure for HTTP-based content delivery networks, caching, and proxies, facilitating its deployment and adoption. Recently, DNS-over-QUIC [19] has emerged to reduce resolution latency by adopting the QUIC protocol. However, recent studies have exposed vulnerabilities in DoT and DoH in safeguarding user privacy [16], [23], [27], particularly against learning-based [16], [23] and IP-based [27] WF attacks.

This has led to the design of the Oblivious DNS over HTTPS (ODoH) [24] protocol, which aims to promote unlinkability between the client's identity (*i.e.,* IP address) and requests. ODoH consists of three components: *Client*, *Oblivious Proxy*, and *Oblivious Target*. To protect the client's identity, the oblivious proxy forwards the client's query, encrypted using a hybrid public-key scheme with the target's public key, to the oblivious target, while replacing any identifiable details. Note that communication between the client and the oblivious target occurs over two sequential secure channels, preventing the proxy from obtaining the client's query in plaintext [24]. Upon receiving the query from the proxy, the oblivious target decrypts and resolves it using the existing DNS infrastructure. It then encrypts the response using the symmetric key from the client's initial query and sends it back to the proxy, which forwards the encrypted response to the client. Despite the similarities between the ODoH and DoH protocols, their design principles and specifications differ noticeably, resulting in different traffic behaviors and patterns. Therefore, an in-depth study of ODoH privacy is necessary. To our knowledge, this work is the first to investigate the privacy of ODoH.

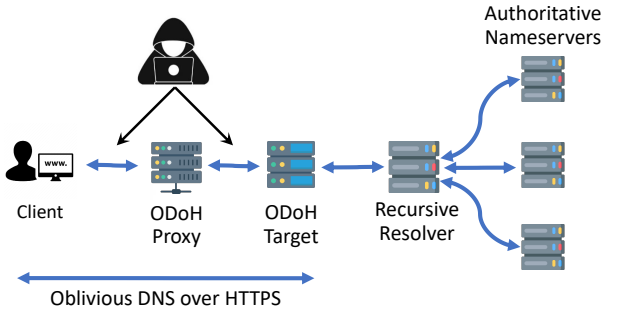


Fig. 1: The passive adversary monitors the ODoH proxy's encrypted traffic, aiming to identify if the client is visiting a target website.

## 3. Threat Model

The ODoH protocol provides the following three guarantees [24]: *(i)* The queries are known only to the client's stub resolver and the target resolver, chosen by the client; *(ii)* The oblivious proxy knows the client's IP address and the chosen target's IP address, but nothing about the query; *(iii)* The oblivious target knows the DNS query and the proxy's IP address, but not the client's IP address. Together, these guarantees ensure unlinkability.

Despite ODoH's guarantees of confidentiality, under the given attack model, it fails to provide privacy protection. We highlight the potential ramifications of privacy violations, which could lead to large-scale Internet surveillance or censorship attacks [39]. In this context, we consider an adversary who seeks to determine whether the client is visiting a target website by analyzing the encrypted DNS traffic and comparing traffic characteristics, such as packet timings and sizes.

We consider a passive adversary capable of intercepting network flows at two locations. Specifically, the adversary may either run a curious proxy to intercept ODoH traffic or monitor the ODoH traffic passing through the proxy (Fig.1). It is important to note that adversaries with the ability to observe both ingress and egress traffic are commonly assumed in existing traffic analysis research [40]–[42]. Recall that the client's DNS query is protected by two security mechanisms: the DNS query is encrypted with the target's public key and encapsulated within secure channels on both the client-proxy and proxy-target links. As a result, the adversary can only collect the client's IP address and the encrypted DNS traffic.

## 4. Data Collection and Attack Methodology

This section outlines the challenges of studying ODoH privacy and details the data collection and pre-processing steps. It also elaborates on the design of proposed deep learning models used for ODoH-based website fingerprinting.

### A. Challenges and Design Choices

*1) DNS Traffic Homogeneity:* Despite some similarities between DNS- and HTTPS-based WF, relying solely on DNS traffic to identify a website is far more challenging, primarily due to the sparsity of DNS traffic. Additionally, unlike HTTPS traffic, which exhibits heterogeneous packet sizes [43], DNS packets are generally smaller and more consistent in size. For example, in UDP-based DNS protocols, packet sizes are typically less than 512 bytes [44]. This homogeneity poses a significant challenge in extracting features and recognizing patterns from DNS traces. To overcome this challenge, we employ advanced deep learning techniques, such as ensemble learning, to capture the subtle differences between ODoH messages from various websites.

*2) Lack of High-quality Data:* Performing data-driven studies requires a large volume of accurate, reliable, and diverse data. However, the absence of a publicly available ODoH trace dataset hinders such research. To address this gap, we utilized the GENI infrastructure to deploy multiple geographically distributed ODoH instances. Over the course of four months, we collected a spatiotemporally diverse ODoH dataset.

*3) Distinguishability of ODoH Traffic:* In ODoH, DNS messages are encrypted and transmitted over two sequential HTTPS channels, *i.e.,* client-proxy and proxy-target. The use of HTTPS in ODoH disguises DNS messages among general HTTPS packets, making it challenging to distinguish ODoH traffic during the data collection process. Prior works have extracted DoH packets from HTTPS traffic traces using TLS fingerprinting and traffic analysis techniques. Specifically, prior work has shown that leveraging unique TLS handshake characteristics [45], packet size distributions and timing patterns [46], and flow-based statistical features [47] can reliably distinguish DoH traffic from generic HTTPS. While DoH and ODoH differ in architecture due to the additional proxy layer in ODoH, this added layer introduces extra encryption and relay mechanisms, increasing the complexity of identifying ODoH traffic. Nonetheless, both protocols use HTTPS, making these techniques viable for distinguishing ODoH traffic from HTTPS. To ensure the integrity of our dataset, we redirected ODoH traces over a dedicated port, preventing the introduction of noisy data and ensuring reliable ground truth labeling.

*4) Out-of-Distribution Data:* WF in the open-world setting often faces the out-of-distribution (OOD) problem, where target websites constitute a small subset of the client's browsing activities, leading to degraded performance. To address this challenge, we adopt open-set classification methods, specifically ensemble learning with an early exit, where the WF neural network identifies OOD instances and terminates the inference task early.

### B. Testbed and Experiment Setup

One of the objectives of this project is to compile an ODoH traffic dataset with diverse spatiotemporal characteristics, specifically ODoH queries originating from various geographical locations over an extended period of time. To achieve this, we set up our experimental environment on GENI [25],

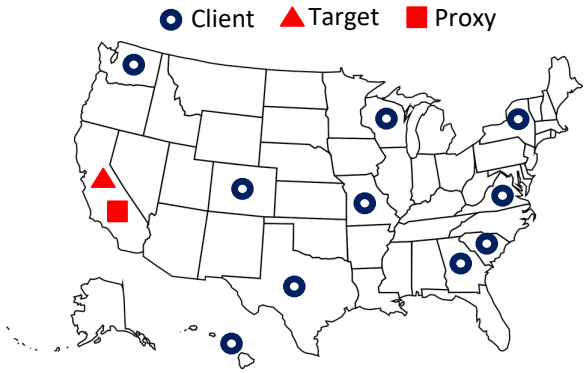



Fig. 2: To collect a usable ODoH dataset with spatiotemporal diversity, we used GENI infrastructure and distributed ODoH clients across various geographical locations, while placing the proxy and the target on two separate nodes.

an infrastructure providing computing and networking resources for the research and education communities which has since transitioned to FABRIC [48]. We ran ten ODoH client instances, distributed across different geographical locations (represented by blue rings in Fig. 2). These nodes were hosted by various institutions, each with its own Internet service provider (ISP), ensuring the diversity of ISP configurations in our dataset. The oblivious proxy and oblivious target (depicted by the red rectangle and red triangle in Fig. 2, respectively) were placed in the western zone, separate from all client nodes, to better mimic realistic traffic characteristics. The ODoH clients, proxy, and target instances were all independent virtual machines running Ubuntu 18.04.1 LTS with Intel Xeon E5-2630 v2 2.60GHz CPUs and 1GB of RAM. For querying DNS, the target was configured to use three publicly available DNS resolvers: Google, Cloudflare, and Quad9.

To add a temporal dimension to our dataset, we collected ODoH traces in two phases over the course of four months, at different times of the day and week. In the first phase, we gathered the traces over a period of two months. After a six-week pause, we resumed data collection for the second phase, which lasted about one month. This approach allowed us to create a comprehensive dataset that provides insights into the performance of ODoH resolvers under varying conditions.

Given the scale and volume of our data collection, we used the GNU Wget tool [49] to automate ODoH queries. Specifically, we utilized the **-p**, **-H**, **–no-cache**, and **–no-dns-cache** options to download web pages and associated resources without caching. The **-p** option (also known as **–page-requisites**) ensures that all elements required to display the page correctly, such as images, CSS, and JavaScript files, are downloaded. The **-H** option enables Wget to span across hosts and domains in recursive retrieval, following links to external domains and resolving web objects even if they are hosted on different servers. The **–no-cache** option disables the caching of downloaded files, ensuring that the most up-to-date versions are retrieved. Finally, the **–no-dns-cache** option disables DNS caching to test the performance of DNS resolvers. While disabling DNS caching can impact DNS-based fingerprinting,

it is important to note that the most common DNS record TTLs are set to shorter durations (*e.g.,* one hour [50]), despite the default 48-hour setting. Additionally, browsers like Safari and Chrome have implemented policy-based cache partitioning methods to determine when the browser cache should be cleared for fresh content. While these methods improve DNS performance, they also reduce the negative impact of DNS caching on WF.

We used **tcpdump** to capture individual ODoH traces, ensuring dataset accuracy and preventing temporal overlap between ODoH queries. To prevent external interference, we ensured that no other network-intensive or background processes were running on the VMs provided by the GENI nodes during data collection. Since the GENI infrastructure did not use the ODoH protocol as part of its native functionality, no unintentional ODoH traces were introduced into our dataset. For each website, we started tcpdump at the beginning of the query and stopped it once the page had fully loaded, ensuring that all ODoH queries and their corresponding responses for loading all objects were captured in full without missing packets or interruptions. To ensure data consistency, we loaded each website three times from each client at different times of the day and across multiple days during each data collection phase. We then post-processed the collected traces to retain only ODoH queries and responses, removing all other packets. To further eliminate noisy traces, we verified trace completion, connection closure, and correct labeling, discarding any timed-out or incomplete traces.

The ODoH client is implemented using the DNScrypt-proxy tool [38], which listens on the loopback address and forwards received DNS messages to the recursive resolver via ODoH. However, relaying DNS messages through ODoH complicates trace identification and labeling. To address this, we deployed a network proxy named SSLSplit [51] as a man-in-the-middle between the proxy and the target. Using SSLSplit, we decrypt the ODoH traffic received from the oblivious proxy to examine the fields solely for labeling purposes. After labeling, SSLSplit re-encrypts the traffic and forwards it to the target. It is important to note that the decryption process is performed exclusively for accurate labeling and none of the decrypted data is used as a feature for WF.

### C. Dataset and Feature Significance

With approximately 1.13 billion websites worldwide[2], attempting to fingerprint each one is impractical. As a result, the attackers often select a subset of all websites (*i.e.,* target) for fingerprinting, where the choice is based on the use case, *e.g.,* censorship or surveillance. Motivated to assess the privacy of the ODoH protocol, we selected *a target set of 100 popular websites* based on the Alexa Top 100 list. Given that Alexa retired its website research and analysis tools in 2022, we cross-referenced the list with other website ranking services, such as SimilarWeb, to ensure the websites' ranks are up-to-date. We found a significant overlap between Alexa and SimilarWeb's suggested websites. Using this target list, we gathered 60390 ODoH traces for our dataset and allocated 60% for training (*i.e.,* 36000 training records are split evenly across 100 classes) and the rest 40% for testing.

Current websites typically include a variety of web objects, such as text, images, and video content, which are hosted on different web servers. As a result, visiting a website often requires a few DNS queries and responses with varying numbers of packets, packet sizes, timestamps, and inter-packet latency. As a result, we compiled two datasets from the collected traces, where one includes the raw PCAP traces and the other includes the aforementioned extracted features. In the second dataset, we represent the ODoH trace of website $i$ as: $F_i = [T_i^q; S_i^q; T_i^r; S_i^r; N_i; TT_i]$, in which $q$ indicates a query and $r$ represents a response, $T$ is a vector of the timestamp, $S$ is a vector of packets' sizes, $N$ is the total number of packets, and $TT$ is the total DNS resolving time for website $i$.

These features play a significant role in website fingerprinting. Firstly, query and response packet sizes are influenced by website complexity, where sites with extensive resources necessitate extra DNS queries for full loading. Additionally, factors such as DNS records and subdomains further impact packet sizes, often requiring multiple queries [52]–[54]. Secondly, the total number of queries and responses serves as a distinguishing factor in website loading behavior. Websites rich in content, such as news or entertainment portals, generate numerous DNS resolutions due to external content sources, while others like educational or government websites yield fewer queries/responses [55]–[57]. Lastly, inter-packet latency characterizes traffic flow behavior due to its dependency on factors, including website complexity, DNS server location, and network conditions. Our analysis captures inter-packet latencies for queries ($T_i^q$) and responses ($T_i^r$), alongside total resolving latency ($TT_i$).

### D. Deep Learning Architectures

In this study, we utilized various classifiers, including a *Fully-Connected Neural Network (FCNN), Convolution Neural Network (CNN),* and *Gated Recurrent Units (GRU)*. For the FCNN model, the preprocessing step calculates per-query or per-trace statistics (Section 4.3), which include features such as the sum and count of the packet sequences. In contrast, the CNN and GRU models use more granular preprocessing, which calculates information for each packet in the trace independently. This allows the models to do a more thorough analysis that utilizes detailed information about each packet in the sequence. This approach facilitates the extraction of spatial and temporal features of the traffic. Additionally, we incorporated a decision tree model to provide a baseline comparison with traditional machine learning techniques. We designed the FCNN model with six hidden layers, incorporating dropout and batch normalization layers to enhance generalizability. For the CNN model, we devised three convolutional layers to capture spatial patterns in the traffic data, followed by three fully connected layers to integrate higher-level features. The GRU model, designed with five GRU layers, leverages its

[2]https://siteefy.com/how-many-websites-are-there/

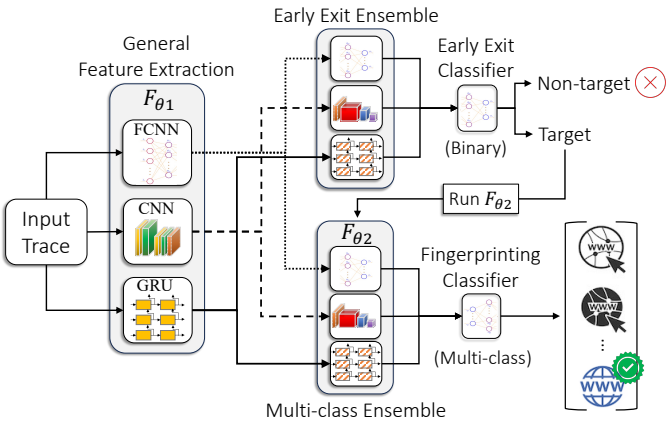



Fig. 3: The proposed early exit ensemble ($E^3$) neural network architecture, including FCNN, CNN, and GRU models, features an early exit module, which terminates data processing when identifying a non-target trace. For target samples, the model processes the sample using the multi-class ensemble.

ability to capture temporal dependencies in sequential data. All architectures employ the ReLU or LeakyReLU activation functions.

Given the complex interplay of spatiotemporal patterns in the data, using GRU or CNN architectures independently results in knowledge representations with limited generalizability and robustness. To address this, we developed an ensemble model (Fig. 3), which combines multiple feature extractors to capture unique spatiotemporal data features. By integrating features extracted from different architectures, this approach enhances robustness and ensures a more comprehensive mapping between feature vectors and learned abstractions. Specifically, our architecture leverages the preprocessed features utilized by the FCNN model while addressing the limitations of preprocessing, which inadvertently removes critical spatiotemporal dependencies present in the raw data. These dependencies are effectively captured by the CNN and GRU architectures. By integrating the preprocessed features with those extracted by the GRU and CNN, this ensemble architecture generates a unified representation that balances efficiency and spatiotemporal richness.

To adapt our ensemble model for open-world scenarios, we incorporate an early exit module that distinguishes between target and non-target websites, terminating processing early for non-target samples. The first block of our Early Exit Ensemble model ($E^3$), *i.e.,* $F_{\theta 1}$, comprises the first few layers of each independent neural network, responsible for general feature extraction. The processed data from $F_{\theta 1}$ is fed into the early exit ensemble block, which uses a subset of layers from the three neural networks to aggregate and evaluate features, providing an initial determination of whether the sample is a target or non-target. The output of this block is then passed to a binary classifier head for final confirmation. Processing is terminated early if the classifier identifies the sample as non-target. For a target sample, the ensemble model forwards the general extracted features from $F_{\theta 1}$ to the multi-class ensemble block, *i.e.,* $F_{\theta 2}$, which combines and refines features from all three neural networks to prepare them for specific target identification. The output of $F_{\theta 2}$ is then fed to a multi-class classifier head to determine the specific target website. We note that our evaluation employed a similar architecture for both the early exit and multi-class ensemble blocks.

## 5. Evaluation

For open-world setting experiments, we generated synthetic data traces with similar statistical characteristics using Synthetic Data Vault [58], a widely used framework for generating background data with the same format and characteristics [59]. Specifically, the synthetic traces closely match the collected traces in key traffic features such as the number of packets, packet sizes, and timing distributions, given their significance in traffic analysis attacks and their influence on model predictions (refer to Figure 6 for feature importance analysis). To achieve this, Synthetic Data Vault captures the joint distribution of these features from real traces and generates new samples that preserve the same statistical properties, ensuring the synthetic traces blend seamlessly with actual traffic. By leveraging synthetic data, we simulate a worst-case scenario for the attacker, as differentiating synthetic traces from real data becomes significantly more challenging when they share strong statistical similarities. This further underscores the strength of the attack in an environment where background traffic is indistinguishable from real-world patterns.

In what follows, we use the term *background* to refer to the synthetic data. We ran multiple experiments with various portions of background traffic (5%, 10%, and 15%) to assess the robustness of our proposed WF classifiers. These ratios are commonly used in the literature [16] and are resting based on the observation that website popularity conforms to a power law distribution [60], which suggests a few popular websites attract the most requests so that an adversary can observe most (80%) of websites that a client is likely to visit during the training.

### A. Ethics of Data Collection

The authors privately disclosed the findings and outcomes of this research on the privacy vulnerability of the ODoH protocol to Cloudflare in February 2024 and offered collaboration in addressing this issue. The data collection and DNS queries were automated to avoid human subject involvement. None of the websites in the target group are illegal or banned in the United States. To prevent introducing excessive overhead on the public recursive resolver, the number of webpage visits was set to a rate similar to what is expected for a human user browsing the Internet.

### B. Reproducibility and Hyperparameter Tuning

The convolution architecture contains a (2x2) kernel, and the output sizes of the layers are 32, 128, and 256. The first and last hidden layers of the FCNN contain 64 neurons, and the rest contain 128 neurons. The units in our GRU layers are 1028, 512, 256 128, and 128 respectively.

TABLE I: WF performance with different ratios of background traffic. 0% represents the close-world setting while other ratios represent the open-world setting.

| Ratio | Algorithm | Accuracy | F1-score | FNR | FPR |
|---|---|---|---|---|---|
| 0% | $E^3$ | **0.9434** | **0.9413** | **0.0513** | **0.0006** |
| | FCNN | 0.9133 | 0.9065 | 0.1518 | 0.0015 |
| | CNN | 0.9109 | 0.9084 | 0.0908 | 0.0009 |
| | GRU | 0.8800 | 0.8789 | 0.1195 | 0.0012 |
| | Decision Tree | 0.8413 | 0.8440 | 0.1586 | 0.0016 |
| 5% | $E^3$ | **0.9231** | **0.9241** | **0.0649** | **0.0008** |
| | FCNN | 0.9040 | 0.8968 | 0.1582 | 0.0014 |
| | CNN | 0.8676 | 0.8794 | 0.0997 | 0.0013 |
| | GRU | 0.8390 | 0.8502 | 0.1283 | 0.0016 |
| | Decision Tree | 0.8251 | 0.8234 | 0.1748 | 0.0017 |
| 10% | $E^3$ | **0.9196** | **0.9211** | **0.0649** | **0.0008** |
| | FCNN | 0.8546 | 0.8746 | 0.1669 | 0.0015 |
| | CNN | 0.8281 | 0.8613 | 0.0996 | 0.0017 |
| | GRU | 0.8009 | 0.8349 | 0.1282 | 0.0020 |
| | Decision Tree | 0.8233 | 0.8219 | 0.1766 | 0.0012 |
| 15% | $E^3$ | **0.9144** | **0.9187** | **0.0649** | **0.0009** |
| | FCNN | 0.8322 | 0.8100 | 0.2032 | 0.0020 |
| | CNN | 0.7922 | 0.8461 | 0.0996 | 0.0021 |
| | GRU | 0.7661 | 0.8206 | 0.1283 | 0.0023 |
| | Decision Tree | 0.7497 | 0.7455 | 0.2500 | 0.0025 |

We trained and tested the models using a 60%-40% train-test split. We searched through different hyperparameter combinations using grid search. We searched across learning rates of 0.01, 0.001, 0.0001, and 0.00001 using Adam and SGD optimizers. The models were also tested using different seeds initialized by Xavier Initialization.

### C. Evaluation Metrics

We evaluate classification models using accuracy, F1-score, false negative rate (FNR), false positive rate (FPR), receiver operating characteristic (ROC) curve, and area under the ROC curve (AUC score).

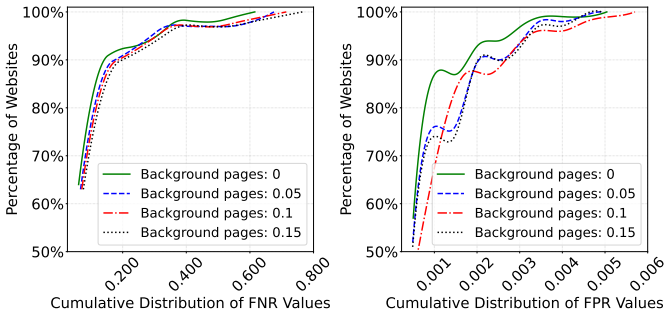
The F1-score calculates the harmonic mean of precision and recall, where precision is the ratio of true positives to predicted positives, and recall is true positives divided by actual positives. FPR is the proportion of negative samples falsely predicted as positive, while FNR measures the model's failure to identify positive instances, indicating the proportion of actual positives classified as negatives. The ROC curve graphically depicts the true positive rate (TPR) against the false positive rate (FPR). It can be extended to multiple classes using the One-vs-Rest (OvR) method, treating each class as positive while others are negative. In multi-class problems, the AUC score is calculated for each class using the OvR method. To obtain a single AUC score for the entire problem, one can average the AUC scores calculated per class.

### D. Evaluation Results

With no background traffic, the $E^3$ model achieves 94% accuracy and outperforms other architectures (Table I). This is in part due to the nature of the ensemble model that effectively learns the spatiotemporal features using three architectures. These results prove that the information leakage of the ODoH protocol can be effectively used for WF. We employed neural network architectures utilized in representative literature for comparison purposes. Specifically, convolutional architectures in DeepCorr [40] and Deep Fingerprinting [5] achieved 91% and 75% attack accuracy, respectively. Additionally, the KNN-based K-FP [55] algorithm resulted in 86% accuracy. In contrast, our proposed $E^3$ architecture achieved a superior accuracy of 94%. We also applied traditional machine learning approaches to ODoH website fingerprinting and observed that the $E^3$ model consistently outperforms these techniques across a range of configurations (Table II).

TABLE II: WF performance with different ratios of background traffic, scaling from 5% to 15%, using traditional machine learning algorithms. We use the sklearn implementation with the default hyperparameters.

| Ratio | Random Forest | Gradient Boosting | AdaBoost |
|---|---|---|---|
| 5% | 0.84 | 0.82 | 0.82 |
| 10% | 0.83 | 0.81 | 0.81 |
| 15% | 0.82 | 0.80 | 0.83 |



(a) FNR (b) FPR

Fig. 4: CDF of FNR and FPR values for the different ratios of background data.

We used the cumulative distribution of FNR (Fig. 4(a)) and FPR (Fig. 4(b)) to study the class-wise performance of the ensemble model. Our results show that 90% of websites have an FNR less than 0.2. Our further investigation identified that only five websites have FNR values higher than 0.3, with the two highest FNR values belonging to the media website category and the other three websites belonging to the retail category, which often change their content. Moreover, all websites have FPR values less than 0.006, and more than 85% of the websites have FPR values of 0.002. These results suggest that the ensemble model can robustly identify almost all of the target websites.

We note that in real-world scenarios, such as large-scale censorship systems, even a small FPR can lead to significant collateral damage – with millions of legitimate web requests, a 0.06% FPR could result in thousands of lawful website visits being erroneously flagged and potentially blocked. For example, a recent study [61] demonstrated that collateral damage in censorship systems can lead to unintended blocking of non-targeted domains due to shared hosting infrastructure and content delivery networks. Nonetheless, censoring authorities often tolerate some level of collateral blocking, particularly in high-control environments [20], [21]. We note that FPR in

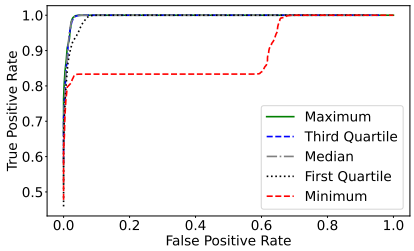



Fig. 5: ROC curve illustrates five websites, each representing the best, worst, and different quartiles of performance results (higher is better).

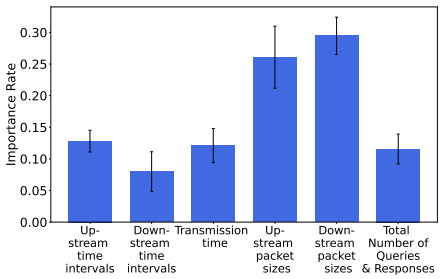



Fig. 6: Feature importance of classifiers; query and response length impact classification most.

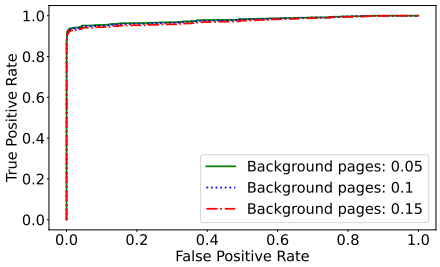



Fig. 7: The ROC curves show that adding background traffic has negligible impact on the model performance.

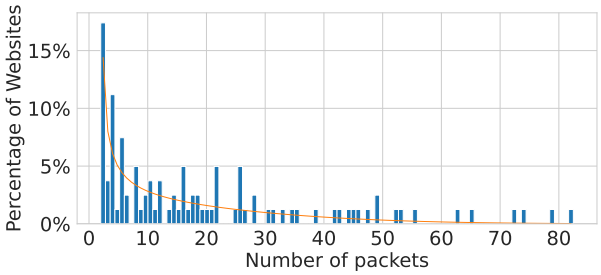



Fig. 8: Distribution of the number of DNS packets for the 100 target websites. The analysis shows that resolving most of the target websites (roughly 80%) requires less than 20 packets.

traffic analysis attacks can be further reduced using advanced techniques, such as metric learning [42] or adaptive filtering based on refined aggregated feature representations [62].

Since we modeled WF as a multi-class classification problem, we used the OvR method to assess the trade-off between sensitivity and specificity of our multi-class ensemble model by evaluating each class with respect to other classes (Fig. 5). We only plotted the ROC curve of five representative websites – the best, worst, and different quartiles. The ensemble model can discriminate between different classes, partly due to the utilization of diverse architectures, which significantly increased the model's generalizability and reduced bias.

Knowing the importance of ODoH traffic features is crucial when designing defensive measures. As such, we analyzed the features' significance and quantified their contribution to predicting the target class using the Gini impurity from the Scikit-Learn package. From Fig. 6, we can observe that packet size (query and response) is the most significant feature. Time-specific features, such as upstream and downstream time intervals and total DNS resolving time are considerably less important, which can be associated with the lower latency of DNS traffic compared to HTTP. We have done further analysis to understand why the number of packets is a less significant feature. The distribution of packets for 100 target websites indicated that over 80% of the target websites required fewer than 20 packets (Fig. 8). This similarity in packet numbers across websites suggests that the total number of packets is less important for the classification task.

*In summary, these results confirm the difficulty of WF using encrypted DNS due to homogeneity and scarcity of DNS traffic; most websites require less than 20 DNS packets. In contrast, HTTPS traffic typically involves more packets as it features additional layers of encryption and larger data payloads. Moreover, HTTPS traffic includes multiple types of packets, such as handshake packets, data packets, and acknowledgment packets. Nonetheless, our results show that information leakage of the ODoH protocol leads to performant WF by solely relying on encrypted DNS traffic.*

*1) Impact of Open-world Setting on Website Fingerprinting:* Aiming to achieve high accuracy in the open-world setting, we devised an early exit module in our ensemble model to identify non-target websites and terminate the process. To evaluate the efficacy of our ensemble model in an open-world setting, we generated a set of background data with the same statistical characteristics as the primary dataset, to measure the ability of the model to distinguish background traffic from the target classes. Starting with $5\%$ background data, we gradually increased the ratio to $10\%$, and eventually $15\%$.

From Table I, we can observe that increasing the background traffic rate decreases the model's accuracy across all architectures. However, the accuracy loss of the ensemble model is marginal. At $15\%$ background traffic ratio, the ensemble model only shows a $3\%$ decrease in accuracy, compared to an $8\%$ decrease in accuracy for FCNN or around a $12\%$ decrease in accuracy for CNNs. More importantly, for the ensemble model, the rate of accuracy drop decreases as the background traffic increases. The results confirm the robustness of the ensemble model to background traffic. Fig. 7 depicts the trend of the ROC curve changes in the open-world setting for various ratios of background data. For this analysis, we consider each target website as a class and the entirety of the background traffic as one class. All the ROC curves achieve more than 0.92 true positive rates with less than 0.001 FPR, indicating that increasing the ratio of background data has a negligible impact (around $2\%$) on the model's accuracy.

*In summary, we generated background traffic with a similar distribution of the primary data for the open-world setting evaluation. The results suggest that the early exit ensemble model can address the challenges of the open-world setting, despite different ratios of background data. The performance*

TABLE III: Summary of distribution of ODoH query instances across different resolvers.

| Client \ Resolver | Google | Cloudflare | Quad9 |
|---|---|---|---|
| Washington | 1938 | 2049 | 1701 |
| Colorado | 2009 | 2083 | 1830 |
| New York | 2001 | 2049 | 1804 |
| Texas | 2723 | 2810 | 2473 |
| Wisconsin | 2124 | 2206 | 1887 |
| Hawaii | 1745 | 1805 | 1544 |
| Georgia | 2071 | 2104 | 1872 |
| Missouri | 2020 | 2084 | 1806 |
| Virginia | 1992 | 2043 | 1800 |
| South Carolina | 1992 | 2046 | 1779 |
| **Average** | 2062.5 | 2127.9 | 1849.6 |

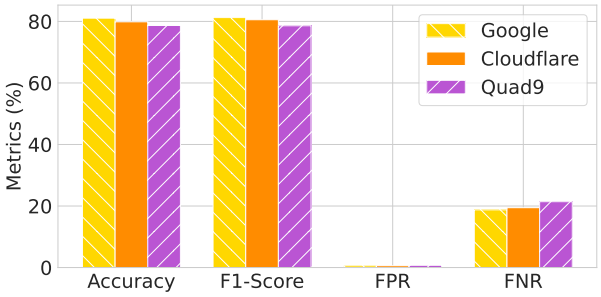
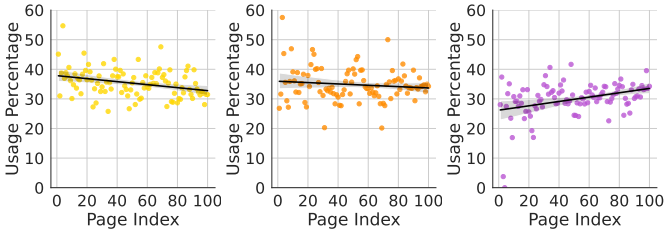



Fig. 9: Attack accuracy against Google, Cloudflare, and Quad9 resolvers. The proposed WF performance is agnostic of the selected resolver. The lower attack accuracy on Quad9 is due to processing less number of queries.




(a) Google (b) Cloudflare (c) Quad9

Fig. 10: Percentage of the queries resolved by each DNS resolver. Google and Cloudflare have been used more frequently for lower rank (more common) websites. In contrast, Quad9 usage has increased with the rank of the website.

TABLE IV: Measuring the impact of ODoH clients' locations on fingerprinting accuracy. The results suggest that clients' location has minimal impact on attack accuracy.

| Client | Accuracy | Macro AVG F1-Score | Weighted AVG F1-Score |
|---|---|---|---|
| Washington | 0.8203 | 0.82 | 0.82 |
| Colorado | 0.84 | 0.82 | 0.82 |
| New York | 0.8779 | 0.86 | 0.87 |
| Texas | 0.821 | 0.80 | 0.80 |
| Wisconsin | 0.85 | 0.84 | 0.84 |
| Hawaii | 0.81 | 0.81 | 0.82 |
| Georgia | 0.9228 | 0.92 | 0.92 |
| Missouri | 0.8918 | 0.88 | 0.88 |
| Virginia | 0.86 | 0.84 | 0.84 |
| South Carolina | 0.875 | 0.84 | 0.85 |
| **Average** | 0.8568 | 0.8465 | 0.846 |

*is in part due to the robustness of the ensemble model in identifying non-target samples.*

*2) Impact of DNS Resolver on Website Fingerprinting:* DNS resolvers can show unique characteristics, *e.g.,* response time, based on network congestion, server availability, caching policy, etc. Some resolvers may prioritize speed by returning cached results even if stale. Others may prioritize accuracy, potentially causing longer response times. In addition, a resolver's location and network infrastructure can impact the response time. For instance, if a user's DNS resolver is far from the authoritative DNS server, the DNS response time may be longer due to increased network latency. Our analysis shows that popular resolvers, *i.e.,* Google, Cloudflare, and Quad9, are equally utilized, with Quad9 resolving slightly less number of queries (Table III). We also did not find any correlation between the client's locations and the selected resolver.

Furthermore, we partitioned the training dataset based on the selected resolver and evaluated the performance of the attack model on each data partition. Fig. 9 summarizes the results of the attack performance against each resolver. From this figure, one can observe that changing the resolver does not impact the accuracy of the attack. We attributed the slight decrease in attack accuracy against the Quad9 resolver to the lower amount of DNS traffic it received from the client. We also examined the utilization of different resolvers based on the page index (Fig. 10). Our analysis indicates that websites with a higher rank in the primary dataset use Google and Cloudflare more often. Moreover, the utilization of Google and Cloudflare resolvers decreases with the increase in the websites' rank. In contrast, Quad9 utilization increases for the higher rank and less popular websites.

*In summary, we observed that changing the resolver does not reduce the information leakage of ODoH and cannot protect clients against ODoH-based WF.*

*3) Impact of Location and Time on Website Fingerprinting:* The geographical location of the clients can affect the DNS response time; the closer the client is to the DNS resolver, the lower the network latency and the shorter the response. To study the impact of location factor, we trained an ensemble model for each location by using the data from the other nine locations. We then deployed the attack in locations not in the training set, which allows us to evaluate how well the ensemble model performs on the data from a new, unseen location. As a result, we trained ten instances of our ensemble model, excluding one location in every instance to evaluate the attack efficacy. In these experiments, we did not use background traffic to better understand the impact of location on the attack accuracy; introducing background traffic would have made it difficult to separate the location's influence from background traffic impact.

The results (Table IV) indicate that geographical location plays a recognizable role in WF. While the attack remains successful, one can observe an average accuracy drop of 9%. We measured the number of packets from each location to analyze

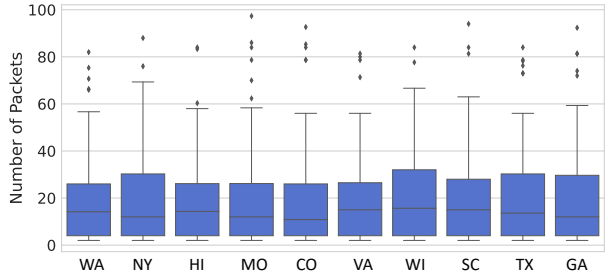



Fig. 11: Distribution of the number of generated packets during resolving 100 websites for different clients.

TABLE V: Measurement factors for time stability

| Dataset | Accuracy | F1-score | FNR | FPR |
|---|---|---|---|---|
| Original Collection | 0.9434 | 0.9413 | 0.0513 | 0.0006 |
| $2^{nd}$ Round Collection | 0.9108 | 0.9108 | 0.0808 | 0.00091 |

the similarity in clients' behavior from different geographical locations. As shown in Fig. 11, the traffic behavior (number of packets) remains the same except in extreme cases, where network instability results in retransmission. Our observations suggest that the impact on WF accuracy for the excluded state is directly correlated with the availability of correlated latency features, which could be extracted if clients are present in closer proximity. For instance, states like Texas and Hawaii see the highest decrease in accuracy due to the lack of nearby clients in the training set, while the attack stays fairly effective in Georgia due to its proximity to South Carolina.

The stability of a website, *i.e.,* changes in the content, is another important factor that affects the accuracy of website fingerprinting. As time passes, the content of the websites can be updated. Such changes can impact DNS traffic characteristics, causing classifiers, which have been trained on older information, to fail in recognizing the new traffic patterns. This is particularly important for websites with frequent content updates, *e.g.,* media or news channels, or those that provide content from various sources. As such, we performed a series of experiments to investigate the consequences of these changes on the ability to classify web pages through time.

For these experiments, we collected the second batch of ODoH traces for the primary set of websites (from one client) after six weeks. To quantify the accuracy loss of the ensemble model, we trained our model using the traffic collected in the first round and verified the model using the new traffic (Table V). The result indicates a 3% accuracy loss and a similar decrease in the F1-score. To identify the reasons for this accuracy loss, we analyzed various features including the number of queries generated for loading a page as an indicator of radical changes in the content of the websites. For each website, we calculated the difference in the number of queries between the old and new datasets (Fig. 12). Our results suggest that 90% of websites have experienced an insignificant change in their traffic fingerprint – in the order of a few packets between the two data collection rounds. We observed that only two websites resulted in a high difference in the number

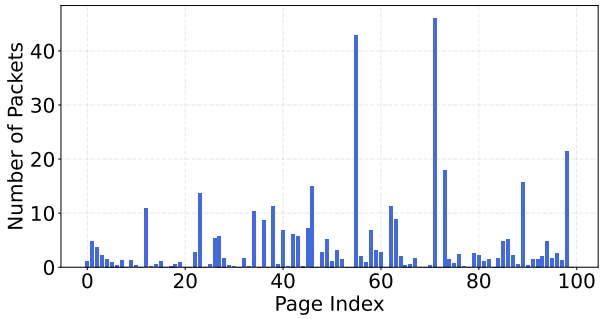



Fig. 12: Change in the number of packets with six weeks between training and test data collection.

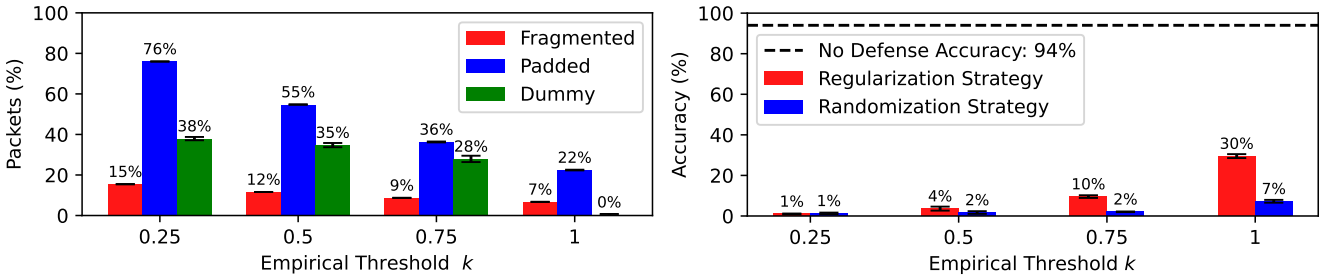
of packets. Our further investigation showed that these two websites are related to health and news/entertainment, which are expected to have moderate to frequent content updates.

*In summary, our analysis indicated that the lack of geographical diversity in the data can negatively impact the accuracy of WF, despite the similarity of the traffic in terms of the number of packets. Moreover, while the instability of websites' content over time negatively impacts WF accuracy, only a small set of popular websites (from our list) frequently update their content. Nonetheless, the inevitable accuracy drop resulting from the target websites' content updates necessitates occasional WF model recalibration by the adversary.*

## 6. In-Transit Defense Mechanism

In this section, we propose a defense mechanism based on traffic perturbation to obfuscate ODoH traffic patterns and mitigate website fingerprinting attacks. Beyond traffic obfuscation, other techniques for protecting DNS privacy include utilizing VPNs [63], anonymization networks like Tor [64], decoy routing [65], and domain fronting [66]. These approaches enhance privacy by relaying traffic through trusted networks, redirecting requests via non-censored paths, or increasing collateral damage by sharing the IP addresses of blocked services with uncensored ones. However, these techniques are vulnerable to various attacks [42], [62], [67] or introduce significant bandwidth and latency overhead [68], making them impractical for DNS traffic.

Our defensive approach perturbs the traffic pattern (in terms of packet sizes and the number of packets sent) by applying padding, fragmentation, and injecting dummy packets. Thus, making it more challenging for the attacker to identify the websites that have been requested. We implemented our solution as a TLS module, operating on the sender and receiver (*e.g.,* the ODoH client and proxy in upstream), to facilitate broader adoption. Building on existing solutions [69], we investigate two strategies to identify the optimal application of traffic perturbation, *regularization* and *randomization*, as detailed in the next subsection. Our results indicate that the *randomization strategy*, despite its simplicity, is more effective in thwarting WF attacks.




(a) Percentages of added padding, fragmentation, and dummy packets

(b) Attack Success Rate

Fig. 13: Analysis of defense strategies against website fingerprinting attacks on ODoH traffic: accuracy reduction and packet modifications across different empirical thresholds.

### A. Traffic Pattern Perturbation Strategies

To perturb the ODoH traffic pattern, we consider two different strategies: *regularization*, which aims to make packet sizes and counts more uniform, and *randomization*, which introduces deviations from typical traffic patterns.

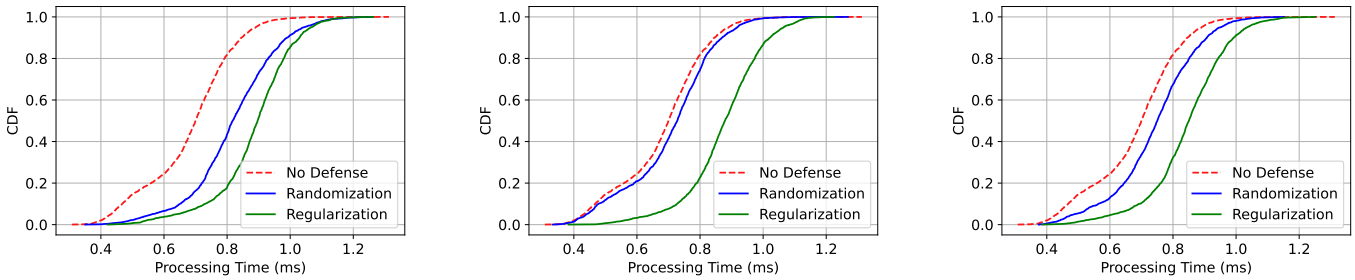
Our *regularization strategy* defines a range of acceptable packet sizes, allowing packets within this range to be transmitted without modification. Packets smaller than the lower limit are padded to meet the minimum size requirement, while packets exceeding the upper limit are fragmented into smaller packets to comply with the specified range. A similar approach is applied to the number of packets per flow: if a request contains fewer packets than the lower limit, dummy packets are injected to meet the threshold. Note that our choice to perturb packet sizes and numbers is motivated by their critical role in attack success, as highlighted by their importance in Fig. 6. We define the acceptance range as $\mu \pm \sigma \cdot k$, where $\mu$ represents the mean value of the distribution (either per packet size and packet count), $\sigma$ is the standard deviation, and $k$ is a tunable threshold that will be set empirically to achieve the desired level of protection. Lower values of $k$ lead to greater obfuscation, though with a higher performance penalty.

In contrast, the *randomization strategy* determines whether to apply padding, fragmentation, or dummy packet injection based on three probabilities assigned to each individual packet. These probabilities govern the likelihood of each transformation, aiming to obscure patterns and make it more challenging for the adversary's model to identify consistent features.

To assess the effectiveness of the two strategies, we conducted a set of experiments using the attack model on data adjusted to mimic the traffic patterns generated by our solution. Specifically, for the regularization strategy, we measured the attack accuracy and tracked the number of padded, fragmented, and dummy packets generated for different values of the empirical threshold $k$. Based on these measurements, we derived the input probabilities for the randomization strategy and applied them to modify the test dataset of the attack model. This approach ensured that, for each value of $k$, the randomization strategy produced a comparable number of padded, fragmented, and dummy packets as those generated by the regularization strategy. By aligning these parameters and modifying the test datasets accordingly, we conducted a fair comparison of the two strategies under similar conditions and evaluated their respective accuracies.

As illustrated in Fig. 13(a), when $k = 0.25$, the regularization strategy resulted in $38\%$ dummy packets, $76\%$ padded packets, and $15\%$ fragmented packets. However, when $k = 1$, these percentages significantly decreased to $7\%$, $22\%$, and $\sim 0\%$, respectively. These values were used as probabilities in the randomization strategy, ensuring that all packets have an equal likelihood of being modified as in the regularization strategy for the corresponding $k$ value. From Fig. 13(b), one can observe that the randomization strategy leads to a less successful attack. Notably, with $k = 1$, the attack accuracy fell to only $7\%$ using the randomization strategy, compared to $30\%$ with the regularization strategy. This suggests that the randomization strategy is a more effective defense mechanism, especially under performance constraints with a limited modification budget. By employing independent and probabilistic perturbations, it introduces greater unpredictability, hindering the attack model's ability to identify patterns.

To enhance the indistinguishability of real, dummy, and padded packets, we designed the system to exclude the transmission of any auxiliary metadata, even in encrypted form. Additionally, we improved performance by eliminating the decryption of unnecessary data, such as padding and dummy packets. When sending a dummy packet, our TLS module appends an HMAC, a cryptographic algorithm commonly used to verify data authenticity and integrity, to the packet. For this purpose, we used SHA-256 as the underlying hash function to compute the HMAC with a 32-byte HMAC key derived from the TLS session key, following the specifications of RFC 5705 [70]. The HMAC message is then created by extracting the last 32 bytes of the encrypted body. To identify a dummy packet, the receiver derives the HMAC key, extracts the HMAC message from the corresponding portion of the encrypted body, and compares it to the computed HMAC. A match between the two HMAC values confirms that the packet is a dummy and should be discarded. This approach is more efficient than decrypting the entire packet. The padding process is similar, but involves iteratively appending multiple HMACs to reach the desired length; in each iteration, the




(a) Padding Overhead (b) Fragmentation Overhead (c) Dummy Packets Overhead

Fig. 14: Introduced overhead in terms of the processing time for defense mechanisms with (a) padding, (b) fragmentation, and (c) injected dummy packets.

HMAC message is derived from the HMAC value generated in the previous step.

### B. Defense Implementation and Performance Evaluation

We prototyped WFSafe TLS as an operating system-independent website fingerprinting (WF) mitigation defense by integrating our solution into the TLS library. This mechanism can be seamlessly adopted by incorporating our custom TLS module into ODoH client or proxy implementations.

To evaluate its performance, we conducted experiments using a locally deployed ODoH client and server configured with our TLS module. The proxy server was setup with mocked DNS responses to eliminate external factors, such as network latency. Additionally, we enabled connection reuse between the client and server, following the approach in [24]. We analyzed the average number of DNS queries per webpage for the Alexa Top 100 list and empirically determined this number to be approximately 20 (consistent with our earlier analysis in Fig. 8). To simulate typical usage, we sent 20 DNS queries over the same connection and measured the total processing time, normalizing it to compute the per-query processing time. Each experiment was repeated 2000 times to ensure statistical significance. The tests were conducted on commodity hardware featuring an Intel Core i5 processor (i5-1240P), 16GB of LPDDR5 RAM, and running Ubuntu 22.04. In these experiments, we simulated the worst-case scenario in terms of additional bytes introduced, applying the defense mechanisms more intensively than would occur in a realistic setting. Specifically, in the padding experiment, we added 320 bytes of padding to each packet. In the fragmentation experiment, every packet was fragmented, and in the dummy packet experiment, we inserted one dummy packet of 640 bytes for every two packets.

For DNS traffic, minimizing delays is crucial to maintaining acceptable query response times. To evaluate the potential impact of our defense mechanisms, we measured the imposed processing latency. Fig. 14 illustrates the Cumulative Distribution Function (CDF) of the processing time for applying (a) padding, (b) fragmentation, and (c) dummy packets in DNS queries. Our results indicate that the randomization strategy is faster than the regularization strategy, as the latter requires retrieving mean value and standard deviation data at the start of each connection. Even under worst-case conditions, the additional delay introduced by our defense mechanisms is minimal, differing only by tenths of milliseconds from the baseline measurement without defense. This overhead is negligible compared to typical ODoH response times, which range from approximately 100 to 1000 ms [24], a significant achievement given the low-latency requirements of DNS traffic.

*In summary, our proposed WFSafe TLS effectively mitigates website fingerprinting attacks while introducing only a negligible latency of tenths of milliseconds, which is significantly lower than typical ODoH response times.*

## 7. Conclusion

In this work, we studied the resiliency of the ODoH protocol, a recently developed secure DNS design, against traffic correlation attacks. In particular, we orchestrated a deep learning-based website fingerprinting attack on ODoH traffic. The results indicated that a passive adversary can identify the client's visited website with 94% accuracy, in open- and closed-world settings. Moreover, our findings indicated that time stability marginally impacts attack accuracy, primarily due to modification of the website's content over time. In addition, clients' geographical locations can have a more pronounced impact on the WF accuracy. To address this vulnerability, we introduced an in-transit defensive measure called WFSafe TLS, designed to effectively mitigate website fingerprinting attacks against the ODoH protocol. WFSafe TLS perturbs ODoH traffic patterns through padding, fragmentation, and dummy packet injection, without compromising protocol compatibility or significantly impacting performance.

## References

[1] S. Farrell and H. Tschofenig, "Pervasive Monitoring Is an Attack," RFC 7258, May 2014.

[2] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proceedings 2002 IEEE Symposium on Security and Privacy*. IEEE, 2002, pp. 19–30.

[3] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas, "Circuit fingerprinting attacks: Passive deanonymization of Tor hidden services," in *24th USENIX Security Symposium*, 2015, pp. 287–302.

[4] G. Cherubin, R. Jansen, and C. Troncoso, "Online website fingerprinting: Evaluating website fingerprinting attacks on Tor in the real world," in *31st USENIX Security Symposium*, 2022, pp. 753–770.

[5] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1928–1943.

[6] S. E. Oh, S. Sunkam, and N. Hopper, "p-FP: Extraction, classification, and prediction of website fingerprints with deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 191–209, 2019.

[7] N. P. Hoang, A. A. Niaki, P. Gill, and M. Polychronakis, "Domain name encryption is not enough: privacy leakage via IP-based website fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, pp. 420–440, 2021.

[8] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted DNS → Privacy? a traffic analysis perspective," *arXiv preprint arXiv:1906.09682*, 2019.

[9] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," *arXiv preprint arXiv:1708.06376*, 2017.

[10] T. G. Ejeta and H. J. Kim, "Website fingerprinting attack on Psiphon and its forensic analysis," in *Digital Forensics and Watermarking: 16th International Workshop, IWDW 2017, Magdeburg, Germany, August 23-25, 2017, Proceedings 16*. Springer, 2017, pp. 42–51.

[11] Y. Shi and S. Biswas, "Website fingerprinting using traffic analysis of dynamic webpages," in *2014 IEEE Global Communications Conference*, 2014, pp. 557–563.

[12] M. Di Martino, P. Quax, and W. Lamotte, "Realistically fingerprinting social media webpages in HTTPS traffic," ser. ARES '19. Association for Computing Machinery, 2019.

[13] G. Huang, C. Ma, M. Ding, Y. Qian, C. Ge, L. Fang, and Z. Liu, "Efficient and low overhead website fingerprinting attacks and defenses based on TCP/IP traffic," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 1991–1999.

[14] S. Oh, M. Lee, H. Lee, E. Bertino, and H. Kim, "AppSniffer: Towards robust mobile app fingerprinting against VPN," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2318–2328.

[15] A. Gómez-Boix, P. Laperdrix, and B. Baudry, "Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 309–318.

[16] R. Houser, Z. Li, C. Cotton, and H. Wang, "An investigation on information leakage of DNS over TLS," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT '19. ACM, 2019, p. 123–137.

[17] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)," Internet Requests for Comments, RFC Editor, RFC 7858, May 2016.

[18] P. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," Internet Requests for Comments, RFC Editor, RFC 8484, October 2018.

[19] C. Huitema, S. Dickinson, and A. Mankin, "DNS over Dedicated QUIC Connections," Internet Requests for Comments, RFC Editor, RFC 9250, May 2022.

[20] H. Nebuchadnezzar, "The collateral damage of internet censorship by DNS injection," *ACM SIGCOMM CCR*, vol. 42, no. 3, pp. 10–1145, 2012.

[21] N. P. Hoang, A. A. Niaki, J. Dalek, J. Knockel, P. Lin, B. Marczak, M. Crete-Nishihata, P. Gill, and M. Polychronakis, "How great is the great firewall? measuring China's DNS censorship," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3381–3398.

[22] C. Bocovich, A. Breault, D. Fifield, X. Wang *et al.*, "Snowflake, a censorship circumvention system using temporary WebRTC proxies," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 2635–2652.

[23] T. Dahanayaka, Z. Wang, G. Jourjon, and S. Seneviratne, "Inline traffic analysis attacks on DNS over HTTPS," in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE Computer Society, 2022, pp. 132–139.

[24] S. Singanamalla, S. Chunhapanya, J. Hoyland, M. Vavruša, T. Verma, P. Wu, M. Fayed, K. Heimerl, N. Sullivan, and C. Wood, "Oblivious DNS over HTTPS (ODoH): A practical privacy enhancement to DNS," *Proceedings on Privacy Enhancing Technologies*, vol. 4, pp. 575–592, 2021.

[25] C. Elliott, "Geni-global environment for network innovations." in *LCN*, 2008, p. 8.

[26] J. Bushart and C. Rossow, "Padding ain't enough: Assessing the privacy guarantees of encrypted DNS," 2020.

[27] N. P. Hoang, A. A. Niaki, P. Gill, and M. Polychronakis, "Domain name encryption is not enough: privacy leakage via IP-based website fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, pp. 420–440, 2021.

[28] A. Hintz, "Fingerprinting websites using traffic analysis," in *Privacy Enhancing Technologies (Workshop)*. Springer, 2003, pp. 171–178.

[29] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at Internet scale." in *NDSS*, 2016.

[30] D. Kim, L. Ho, Y.-H. Kim, W.-g. Kim, and D. Hwang, "Poster: A pilot study on real-time fingerprinting for Tor onion services," in *The Network and Distributed System Security Symposium (NDSS) 2021*, 2021.

[31] K. Abe and S. Goto, "Fingerprinting attack on Tor anonymity using deep learning," *Proceedings of the Asia-Pacific Advanced Network*, vol. 42, pp. 15–20, 2016.

[32] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN: A data-efficient website fingerprinting attack based on deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 292–310, 2019.

[33] Y. Wang, H. Xu, Z. Guo, Z. Qin, and K. Ren, "snWF: Website fingerprinting attack by ensembling the snapshot of deep learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1214–1226, 2022.

[34] X. He, J. Wang, Y. He, and Y. Shi, "A deep learning approach for website fingerprinting attack," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. IEEE, 2018, pp. 1419–1423.

[35] M. Guo, J. Fei, and Y. Meng, "Deep nearest neighbor website fingerprinting attack technology," *Security and Communication Networks*, vol. 2021, pp. 1–14, 2021.

[36] M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 255–263.

[37] S. Nowaczewski and W. Mazurczyk, "Improving security of future networks using enhanced customer edge switching and risk-based analysis," *Electronics*, vol. 10, no. 9, p. 1107, 2021.

[38] DNSCrypt, "DNSCrypt Protocol," https://github.com/DNSCrypt/dnscrypt-protocol/blob/master/DNSCRYPT-V2-PROTOCOL.txt, 2020.

[39] G. Aceto and A. Pescapé, "Internet censorship detection: A survey," *Computer Networks*, vol. 83, pp. 381–421, 2015.

[40] M. Nasr, A. Bahramali, and A. Houmansadr, "DeepCorr: Strong flow correlation attacks on Tor using deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1962–1976.

[41] K. Kohls and C. Pöpper, "Digestor: Comparing passive traffic analysis attacks on Tor," in *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*. Springer, 2018, pp. 512–530.

[42] S. E. Oh, T. Yang, N. Mathews, J. K. Holland, M. S. Rahman, N. Hopper, and M. Wright, "DeepCoFFEA: Improved flow correlation attacks on Tor via metric learning and amplification," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1915–1932.

[43] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Understanding website complexity: measurements, metrics, and implications," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 313–328.

[44] P. Mockapetris, "Domain names - implementation and specification," Internet Requests for Comments, RFC Editor, RFC 1654, July 1987. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1035

[45] 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*, ser. ARES '20. ACM, 2020.

[46] M. Zhan, Y. Li, G. Yu, B. Li, and W. Wang, "Detecting DNS over HTTPS based data exfiltration," *Comput. Netw.*, vol. 209, no. C, may 2022.

[47] A. Aggarwal and M. Kumar, "An ensemble framework for detection of DNS-over-HTTPS (DOH) traffic," *Multimedia Tools and Applications*, vol. 83, no. 11, pp. 32 945–32 972, 2024.

[48] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47.

[49] F. S. Foundation, "GNU wget," http://www.gnu.org/software/wget/, Nov. 2010. [Online]. Available: http://www.gnu.org/software/wget/

[50] G. C. M. Moura, J. Heidemann, R. de O. Schmidt, and W. Hardaker, "Cache me if you can: Effects of DNS Time-to-Live," in *Proceedings of the ACM Internet Measurement Conference*. Amsterdam, the Netherlands: ACM, oct 2019, p. to appear.

[51] D. Roethlisberger, "SSLsplit-transparent SSL/TLS interception," *Rö's Wiki,[Online]. Available: https://www. roe. ch/SSLsplit (visited on 2021-04-25)*, 2016.

[52] M. Shen, Y. Liu, S. Chen, L. Zhu, and Y. Zhang, "Webpage fingerprinting using only packet length information," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[53] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12, 2012, p. 605–616.

[54] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," in *Network and Distributed System Security Symposium*, 2016.

[55] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium*, 2016, pp. 1187–1203.

[56] S. Li, H. Guo, and N. Hopper, "Measuring information leakage in website fingerprinting attacks and defenses," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1977–1992.

[57] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at internet scale," 02 2016.

[58] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," 10 2016, pp. 399–410.

[59] C. G. Cordero, E. Vasilomanolakis, A. Wainakh, M. Mühlhäuser, and S. Nadjm-Tehrani, "On generating network traffic datasets with synthetic attacks for intrusion detection," *ACM Transactions on Privacy and Security (TOPS)*, vol. 24, no. 2, pp. 1–39, 2021.

[60] L. A. Adamic and B. A. Huberman, "Power-law distribution of the world wide web," *science*, vol. 287, no. 5461, pp. 2115–2115, 2000.

[61] M. Wu, J. Sippe, D. Sivakumar, J. Burg, P. Anderson, X. Wang, K. Bock, A. Houmansadr, D. Levin, and E. Wustrow, "How the great firewall of China detects and blocks fully encrypted traffic," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 2653–2670.

[62] T. Chawla, S. Mittal, N. Mathews, and M. Wright, "ESPRESSO: Advanced end-to-end flow correlation attacks on Tor," in *Proceedings of the 8th Asia-Pacific Workshop on Networking*, 2024, pp. 219–220.

[63] D. Nobori and Y. Shinjo, "VPN Gate: A Volunteer-Organized public VPN relay system with blocking resistance for bypassing government censorship firewalls," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 229–241.

[64] R. Dingledine, N. Mathewson, P. F. Syverson *et al.*, "Tor: The second-generation onion router," in *USENIX security symposium*, vol. 4, 2004, pp. 303–320.

[65] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer, "Decoy routing: Toward unblockable internet communication," in *USENIX workshop on free and open communications on the Internet (FOCI 11)*, 2011.

[66] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant communication through domain fronting," *Proceedings on Privacy Enhancing Technologies*, 2015.

[67] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper, "Routing around decoys," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 85–96.

[68] M. Nasr, H. Zolfaghari, and A. Houmansadr, "The waterfall of liberty: Decoy routing circumvention that resists routing attacks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 2037–2052.

[69] X. Xiao, X. Zhou, Z. Yang, L. Yu, B. Zhang, Q. Liu, and X. Luo, "A comprehensive analysis of website fingerprinting defenses on Tor," *Comput. Secur.*, vol. 136, no. C, Feb. 2024. [Online]. Available: https://doi.org/10.1016/j.cose.2023.103577

[70] E. Rescorla, "Keying Material Exporters for Transport Layer Security (TLS)," RFC 5705, Mar. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5705