Privacy Leakage in Encrypted DNS Traffic: Analysis and Countermeasure

by

GUANNAN HU

Dissertation

submitted to the Department of Informatics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI
July 2024

Committee

Advisor Dr. Kensuke Fukuda
Professor of National Institute of Informatics/SOKENDAI, Japan

Subadvisor Dr. Yusheng Ji
Professor of National Institute of Informatics/SOKENDAI, Japan

Examiner Dr. Megumi Kaneko
Professor of National Institute of Informatics/SOKENDAI, Japan

Examiner Dr. Michihiro Koibuchi
Professor of National Institute of Informatics/SOKENDAI, Japan

Examiner Dr. Hideya Ochiai
Associate Professor of the University of Tokyo, Japan

Acknowledgements

I would like to thank my supervisor, Prof. Kensuke Fukuda, for guiding me and supporting me when I need support during these years. His strictness and patience have been invaluable in helping me to learn how to do research work and give me brilliant ideas on how to think about the program.

I am thankful to my advisory committee, Prof. Yusheng Ji, Prof. Shunji Abe, Prof. Michihiro Koibuchi, Prof. Megumi Kaneko, and Prof. Hideya Ochiai for their helpful comments and suggestions.

I appreciate my lab members for discussing and provide intellectual comments on my research. Also, sharing the research experience with me.

I am pleased to my family for staying with me forever. Typically, I also want to thank SOKENDAI and the National Institute of Informatics for academic support and provide the opportunity to do further work.

Abstract

The Domain Name System (DNS) translates domain names into one or more IP addresses by asking the caching server when the users enter a URL into their browser. The DNS caching resolver stores a database of domain names and their corresponding IP addresses in the global network. While the user types a domain name into the web browser, the host first checks the caching resolver to see if it already has the corresponding IP address. If no, it begins by querying a root server, then the top-level-domain (TLD) server, and finally the authoritative name server.

DNS is designed to be sent in plain text according to RFC 1035 [1], allowing the adversary to eavesdrop on DNS communication by monitoring the network. DNS messages are unencrypted and contain domain names representing the users' private information, such as health, finance, and religion, which leads to information leakage while users visit websites. Intending to protect the person's private information, several encrypted DNS protocols have been proposed: DNS over HTTPS (DoH), DNS over TLS (DoT), and DNS over QUIC (DoQ). These protocols encrypt DNS packets between the client and DNS caching resolver with different underlying transports. Some public DNS caching resolvers have already supported DoT and DoH, such as Google, Cloudflare, and Quad9. Only the AdGuard and NextDNS support the DoQ protocol, at the time of writing. Also, the user could change some browser settings (e.g., Chrome and Firefox) to enable the DoH. However, recent studies showed that the adversary could still infer the category of websites even using DoT and DoH by analyzing the encrypting DNS traffic. This dissertation aims to investigate the information leakage problem of encrypted DNS protocols and develop countermeasures to protect user privacy against website fingerprinting attacks.

In the first half of this dissertation, we study the privacy leakage problem of

encrypted DNS traffic (i.e., DoT, DoH, and DoQ) with three different DNS caching resolvers (NextDNS, Bind, and Google). Depending on the DNS software configurations (public and local DNS caching resolvers), we consider two threat models to simulate the website fingerprinting on binary and multi-classification. We choose 30 categories from Alexa's top 300,000 websites and select the top-400 websites for each category. For the binary classification, we split the dataset into *'Sensitive'*, related to personal information such as health, finances, religion, and government, and *'Non-Sensitive'*. As the baseline analysis, we evaluate the classification performance of DoQ traffic with balanced (10 categories from *'Sensitive'*, and 10 categories from *'Non-Sensitive'* randomly) and imbalanced (10 categories from *'Sensitive'*, and remaining 20 categories from *'Non-Sensitive'*) datasets on Bind and NextDNS. We also examine the performance of DoQ, DoH, and DoT with Google resolver. We find that the classification performance of the websites is high both in NextDNS, Bind, and Google resolvers for identifying whether the user visits the category of websites. We confirm no significant influence on whether the local resolver is cached and caching order. More particularly, we indicate that discriminative features are mainly related to the inter-arrival time of packets and packet length. For the multi-classification, we notice the performances decrease as the number of categories increases for the Bind resolver, meaning that the impact of the leakage is limited. We also notice the performances are not directly related to the number of crawls.

From the important features, a promising approach is to control the inter-arrival distribution and packet length for the mitigation. In the second half of this dissertation, we further investigate four possible countermeasures that could affect the classification results: using AdBlocker extension, disabling DNS prefetch, adding random delay in responses, and padding the DNS payload. 1) We show that using AdBlocker and disabling DNS prefetch are less effective in mitigating the attack. 2) We find that mean F1 scores decrease as the delays increase. Specifically, it decreases the classification performance by 22% with NextDNS and 18% with Bind. 3) DNS padding decreases the classification performance by 9%. We further investigate the combination of the two countermeasures: both adding random (0-60ms and 0-100ms) delays and padding the DNS payload. We confirm that the combined method could greatly reduce the classification performance, on average 27% of binary and 22% of multi-classification in Bind. These results indicate that adding random time and padding can protect users'

information from the website fingerprinting attack.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

As we rely more on the Internet, the network becomes the most indispensable part of human life that involves study, work, and entertainment. Figure 1.1 shows the simple process while the user visiting the websites. The user enters the URL (i.e., *www.example.com*) of the website, and the browser initiates the DNS query to DNS caching resolver the domain name into an IP address (A of Figure 1.1). Then, the browser initiates the TCP connection to the IP address obtained from the DNS resolver, once the connection is established, the browser sends the HTTP request to the web server for the resource request (i.e., webpage, image, video, and javascript). The web server receives the HTTP request and processes it (B of Figure 1.1). We always search for information such as financial, social media, government, and business through the network. As a result, the network communication may contain personal private information. Without proper security measures, this information is vulnerable to the malicious adversary. HTTPS [2](Hypertext Transfer Protocol Secure) is often used to

encrypt web communication using TLS (Transport Layer Security) or SSL (Secure Socket Layer) protocol.

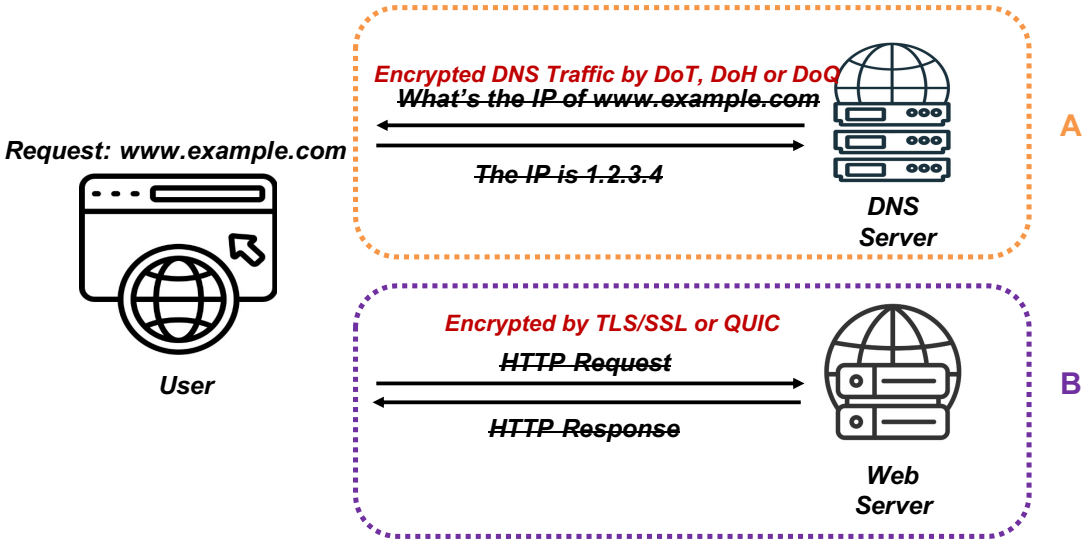


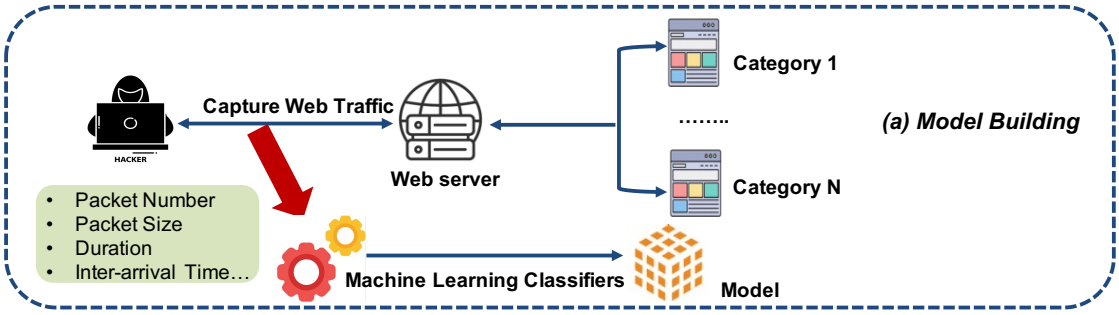
Figure 1.1: What happened when you visit the website?

However, some research [3, 4] reported that web traffic flow information, such as packet arrival time and size, is still exposed even though using these encryption protocols (i.e., HTTPS and QUIC) to protect the web contexts. For example, "Pervasive Monitoring" [5] is the main attack in that the adversary could monitor the network communication and infer the users' private information by analyzing the packet features (i.e., packet length, count, and time). The attack flow in the pervasive monitoring is as follows (see also Figure 1.2):

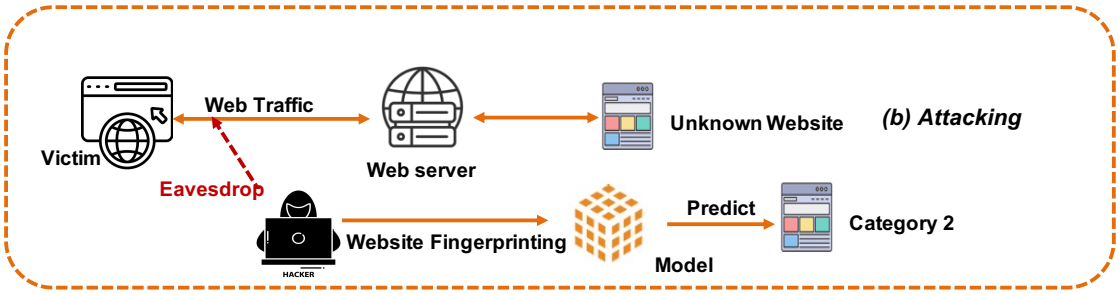
1. **Data Collection and Feature Extraction:** The attacker first collects encrypted web traffic by visiting websites using his/her environment that is similar to the user. The attacker could label the category based on the domain names in advance and extract various flow features, such as packet sizes, timing, and packet number. Divide the dataset into two sets: training and testing in Figure 1.2 (a).

2. **Model Building:** The attacker uses cross-validation with the training dataset to measure the F1 score. After evaluating performance, the attacker selects the best algorithm and parameter to build the model in Figure 1.2 (a).

3. **Model Evaluation:** The attacker evaluates the model with the testing data. When the attacker eavesdrops on the network communication from an unknown user, the attacker predicts which websites the user is visiting by asking the well-trained model in Figure 1.2 (b).



(a) Model Building



(b) Evaluation

Figure 1.2: Website Fingerprinting Attack on Website

Similar to information leakage in encrypted web traffic, we focus on another protocol related to the website for information leakage, Domain Name System (DNS) [1]. When the user types the website (i.e., www.example.com) into the browser, DNS translates this human-readable domain to the numerical IP address. The DNS is a global service for mapping between hostnames and IP addresses in network communication. While the user accesses the webpage, the host first sends queries related to the webpage to the DNS caching resolver. After the DNS resolver lookups the domains, then returns the results to the client. In this process, the DNS packets are unencrypted and easily be eavesdropped by the adversary. There are three privacy disclosure problems: client-side, transmission process, and DNS resolver-side. With the growing awareness of privacy and security concerns on the DNS, there has been increasing adoption of

encrypted DNS protocols such as DNS over HTTPS (DoH) [6] and DNS over TLS (DoT) [7], DNS over QUIC (DoQ) [8], and DNSCrypt [9]. The purpose of encrypted DNS protocols is to encrypt DNS queries made by the client and the corresponding DNS responses received from the DNS caching resolver. This encryption ensures that the content of DNS queries and responses remains classified and cannot be tampered with by unauthorized parties. Currently, various public DNS services support the encrypted DNS protocol. For example, Google [10], Cloudflare [11], and Quad9 [12] have already supported the DoT and DoH. Furthermore, AdGuard [13] and NextDNS [14] support DoQ. Here, we intend to understand whether the encrypted DNS, especially DoQ has information leakage problem, and which factors affect the performance. The process of website fingerprinting attack on the encrypted DNS is similar to traditional web traffic. We describe it as follows (see also Figure 1.3):

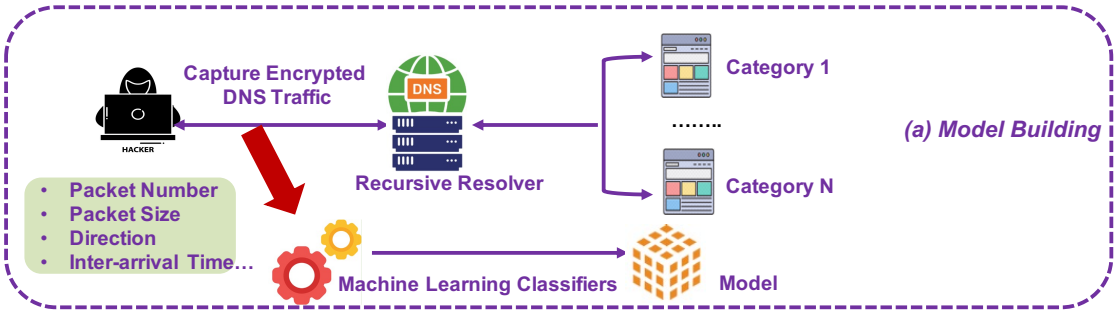
1. **Data Preparation:** The attacker first needs to collect encrypted DNS traffic by visiting websites using an environment that is similar to the user. For example, using Google or Firefox browser, connect to the popular DNS caching resolver that supports encrypted DNS on the home or public Wi-Fi network. The attacker could label the category based on the domain names in advance. Then, the attacker extracts flow features, like packet size, timing, and packet number in Figure 1.3 (a).

2. **Model Building:** The attacker uses the preprocessed dataset to build a fingerprint model that maps traffic features to classify the types of websites with machine learning techniques in Figure 1.3 (a).

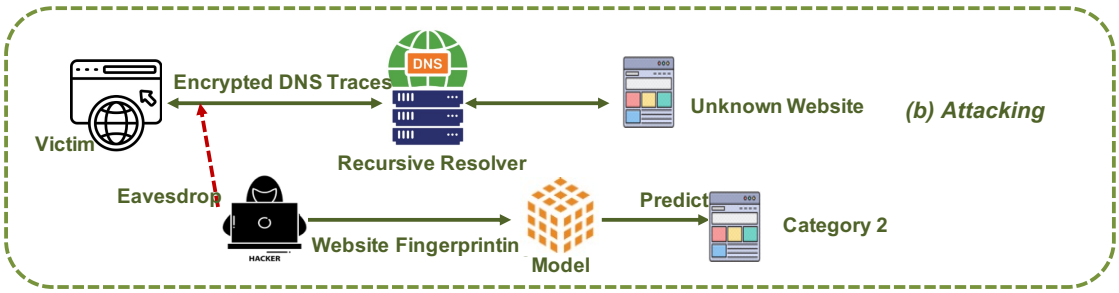
3. **Model Evaluation:** The attacker evaluates the performance of the trained model. When the attacker eavesdrops on the network communication from an unknown user, they predict which websites the user is visiting by asking the well-trained model in Figure 1.3 (b).

1.2 Problem Statement and Research Questions

In this thesis, we investigate the information leakage problem of encrypted DNS and clarify its countermeasure to mitigate the attack. Recent studies investigated



(a) Model Building



(b) Evaluation

Figure 1.3: Website Fingerprinting Attack on Encrypted DNS

the information leakage by DoT [15] and DoH [16, 17]. They found that information leakage is still possible even in DoT and DoH. Ref. [15] also showed the help of padding DNS payload on DoT is limited.

In the light of the above discussion, our research questions will be broken down into four research questions (RQ):

1. RQ1: Could the encrypted DNS protocol (especially for DoQ) defend against the website fingerprinting attack?

2. RQ2: If we examine different encrypted DNS protocols or DNS resolvers, will we get different results?

3. RQ3: What are the main factors affecting the results ?

4. RQ4: How to mitigate the leakage on DoQ?

1.3 Contributions

Our contributions are based on the four research questions above. The goal of this paper is to investigate the information leakage of encrypted DNS protocols, mainly focusing on the DNS over QUIC, and present four possible countermeasures to mitigate the privacy leakage of DoQ. In the first part of this study, we investigate the privacy leakage problem of DoQ protocol with three different DNS recursive resolvers (Bind, NextDNS, and Google) for binary and multi-classification. Also, we analyze the binary classification performance of DoT and DoH on Google. The second part is to mitigate the privacy leakage of DoQ using four possible countermeasures: using AdBlocker extension, disabling DNS prefetch, adding random delay, and padding the DNS payload.

1.3.1 Characterizing Privacy Leakage in Encrypted DNS Traffic

Due to several encrypted DNS protocols have been proposed, we intend to understand whether these could protect the information of web privacy. For binary classification, we confirm that the information leakage against website fingerprinting attacks exists in DoQ, as well as DoT and DoH. The classification performance of the website is all high in Bind, NextDNS, and Google resolvers. For binary classification, we find high discriminative features to infer the website which is mainly related to inter-arrival time and packet length. By analyzing the caching effect of DoQ at the local resolver, we find the randomization of the caching order does not change the performance; Thus, the caching effect is considerably small. We further confirm that the classification performance with and without caching also does not change it so much. Also, for the multi-classification, we observe the performances decrease as the number of categories increases, meaning that the impact of leakage is limited.

1.3.2 Mitigate the privacy leakage of DoQ

Based on our findings, controlling the inter-arrival time and padding the DNS payload are promising ways to mitigate the website fingerprinting attack on the DoQ. Thus, we investigate three countermeasures that control the inter-arrival time: using AdBlock extension, disabling DNS prefetch, and adding random delays in the DNS process. Padding DNS payload [18] is always used to change the packet length.

Using AdBlocker and disabling DNS prefetch is less effective at mitigating the website fingerprinting attack. Adding random delay decreases the performance by 19% (0.9 to 0.71) with Bind and 21% (0.9 to 0.69) with NextDNS for binary classification and 18% with Bind and 22% with NextDNS for multi-classification. DNS padding also mitigates the classification performance by 10%. We finally demonstrate that combining the two countermeasures: adding random delay (0-60ms and 0-100ms) and padding DNS payload with binary and multi-classification. The degradations of performance are 27% for binary class and 22% for multi-class; they are useful in protecting user's privacy.

1.4 Dissertation outline

The next chapters of the dissertation are summarized as follows.

- **Chapter 2**: We describe the encrypted DNS in detail, including the standard method, deployment, and configuration. Then, we review the work related to the website fingerprinting attack and encrypted traffic like QUIC, HTTPS, and DNS over encrypted protocols with machine learning techniques.
- **Chapter 3**: We explain the simulation of two threat models depending on the configurations. Then, we describe the process of dataset collection, category selection, classifier selection, and feature extraction.
- **Chapter 4**: The chapter discusses the evaluation of binary classification performance for DoQ with three DNS resolvers: Bind, NextDNS, and Google with balanced and imbalanced datasets. We also investigate the classification performance of other encrypted DNS protocols: DoT and DoH. Finally, we describe the multi-classification performance for DoQ traffic in the two configurations.
- **Chapter 5:** We evaluate four possible countermeasures: using AdBlocker extension, disabling DNS prefetch, adding random delay, and padding DNS payload.
- **Chapter 6**: This chapter summarizes the dissertation and discusses our limitations and future works.

2 Background and Related Works

In this chapter, we introduce three types of encrypted DNS protocols in §2.1 and two configurations of encrypted DNS techniques in the network in §2.2, used throughout this thesis to ease the understanding of the following chapters. Then, we present the existing research on the website fingerprinting attack and encrypted traffic (i.e., HTTPS, TLS, and QUIC) with machine learning algorithms. Also, we describe the deployment and analysis of several encrypted DNS protocols, such as DoQ, DoT, and DoH in §2.3.

2.1 DNS Encryption Protocols

2.1.1 Encrypted DNS protocols

Traditional DNS [1] queries are sent in plain text, making them vulnerable to monitoring by eavesdroppers. The DNS does not provide encryption, which leads to information leakage. To address the privacy issues, several protocols have been

proposed: DNS over TLS (DoT), DNS over HTTPS (DoH), and DNS over QUIC (DoQ). These protocols establish the encrypted connections between the client and the DNS resolver to increase user privacy and security.

Like traditional DNS, the IP/TCP and IP/UDP header (5-tuple information) of encrypted DNS protocols are still unencrypted. Encrypted protocols (i.e., HTTPS, TLS, and QUIC) are used to protect DNS data. Therefore, various features of DNS packet headers and payloads (i.e., query name and type, TTL) are unavailable. In contrast, except for the unencrypted IP header, other characteristics like packet size, count, and time-series pattern are still useful for identification. Table 2.1 lists the still visible features in encrypted DNS traffic.

Table 2.1: Compare the available DNS information between DNS and encrypted DNS traffic

Features	Plain DNS	Encrypted DNS
5-tuple information	✓	✓
Packet size	✓	✓
Packet count	✓	✓
Packet timing	✓	✓
Domain Name	✓	
Record types (i.e., A, AAAA, and PTR)	✓	
Record class (i.e., IN)	✓	
TTL	✓	

DNS over TLS (DoT) DoT uses Transport Layer Security (TLS) based on TCP protocol for encryption and authentication in network transmission and standardized in RFC 7858 [7]. DoT uses port **TCP/853**.

DNS over HTTPS (DoH) The Internet Engineering Task Force (IETF) specified DoH in RFC 8484 [6]. DoH is similar to DoT. However, DoH queries and responses are sent by HTTPS or HTTP/2 as transport protocols instead of TCP. DoH uses port **TCP/443**, which is the standard port for HTTPS traffic. DoH has seen wider adoption in web browsers like Firefox and Chrome.

DNS over QUIC (DoQ) More particularly, DoQ is a new protocol to encrypt DNS messages using QUIC (Quick UDP InternetConnections) protocol as an underlying

transport. QUIC is a new transport protocol based on UDP and standardized by IETF RFC 9000 [19]. The implementation of QUIC is based on UDP. TCP and UDP work in the kernel model, while QUIC is implemented in the user space. QUIC is similar to TCP + TLS secure data transmission. Compared to TCP, QUIC is faster and supports better encryption because it enables zero-RTT and TLS 1.3. DoQ uses the special service port, **UDP/784** and **UDP/853** that are currently not widely deployed in the DNS resolver.

2.1.2 Deployment of encrypted DNS protocols

DNS encryptions are applied for the queries and responses between clients and DNS resolvers. DNS resolvers can generally be categorized into two main types: public resolvers and local resolvers. Table 2.2 shows the most popular and widely used DNS resolvers. A few public resolvers, like Google [20], Cloudflare, and Quad9 support DoT and DoH. [21, 22] reported there are 7 DoT-enabled, 62 DoH-enabled, and AdGuard [13] DoQ support resolvers. For now, we find NextDNS [14] also supports the DoQ. The user could directly connect to these encrypted-enable DNS resolvers, the communication between the end user and these resolvers is encrypted.

NextDNS NextDNS [14] protects the users from all kinds of security threats, blocks ads and trackers on websites, and provides encrypted DNS (e.g., DoT, DoH, and DoQ) by configuring the user's device ID.

AdGuard AdGuard DNS [13] blocks the ads on the website and provides the AdGuard DNS servers to support encrypted DNS.

While we want to deploy the encryption protocols between the client and the local DNS resolver (i.e., Bind [23] and Unbound [24]), we need the DNS proxy to encrypt the traffic.

Selecting local DNS resolver software is essential for ensuring efficient and secure DNS resolution within a network environment. Local DNS resolver software gives users greater control over DNS resolution settings (i.e., DNS encryption) and configurations (i.e., cache and padding). We list four popular local DNS resolver software as follows:

Bind BIND [23] is one of the oldest and most widely used DNS server software, It can work as both an authoritative DNS server and a DNS resolver.

Unbound Unbound [24] is a validating, recursive, caching DNS resolver and began to support DoT and DoH, which allows clients to encrypt their communication in version 1.6.0.

Stubby Stubby [25] is a lightweight DNS stub resolver developed by the GetDNS project. It acts as a local DNS resolver that supports DoT and DoH to encrypt DNS queries sent from the client device to the DNS resolver.

Knot Stubby [26] Knot Resolver is a caching full resolver implementation developed by CZ.NIC. It supports DoT and DoH for encrypted DNS resolution at the local level.

Table 2.2: The deployment of DNS caching resolver that supports DNS encryption

	Target Resolvers	DoT	DoH	DoQ
Public	Google	✓	✓	
	Cloudflare	✓	✓	
	Quad9	✓	✓	
	NextDNS	✓	✓	✓
	AdGuard	✓	✓	✓
Local	Bind (ver.9.17.10)	✓	✓	
	Unbound (ver.1.6.0)	✓	✓	
	Stubby (ver.0.1.3)	✓	✓	
	Knot (ver.5.0.1)	✓	✓	

Similarly, some popular browsers such as Firefox [27], Chrome [28], and Opera [29] also offer DNS encryption by adding an option in the configuration setting. On the client side, the user can change the browser's settings to enable DoH. For example, we show that how to set DoH in the Chrome in Figure 2.1, if support, the 'Using DNS over HTTPS (DoH)' shows 'Yes' in the https://one.one.one.one/help/.

Table 2.3 shows browsers that support encrypted DNS and we describe these browsers as follows:

Chrome Chrome [28] is one of the most widely used web browsers globally, developed by Google. Chrome 78 enables opportunistic DoH if the system resolver address matches one of the DoH providers. This experiment is enabled for all platforms except Linux and iOS.

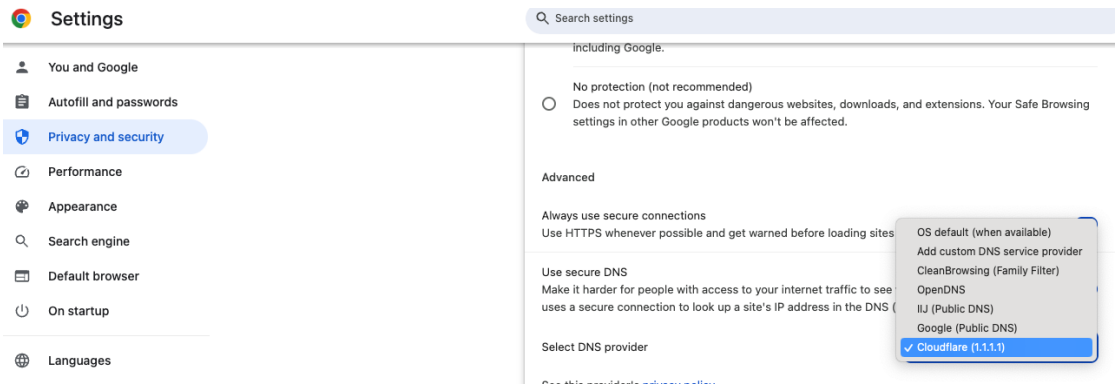


Figure 2.1: Set up DoH in the Chrome browser

Firefox Firefox [27] is an open-source web browser developed by Mozilla Corporation. Firefox supports DoH from version 62, released in May 2018.

Apple Safari Safari [30] supports DoH in version 14, released with macOS Big Sur and iOS 14 in September 2020.

Microsoft Edge Edge [31] introduces support for DoH in version 79, released in January 2020. DoH encrypts DNS queries, enhancing privacy and security for users. Edge allows users to enable DoH in the browser settings and specify their preferred DoH resolver.

Opera Opera [29] added support DoH in version 56, released in September 2018.

Brave Brave [32] supports DoH for encrypted DNS resolution. While the specific version introducing DoH support is not specified, Brave likely implemented support for DoH in earlier versions, aligning with other Chromium-based browsers.

In our work, we mainly focus on DoQ, however, no browser supports it and we need the DNS proxy to encrypt DNS traffic with different protocols. We select Adguard DNS proxy [33] and describe it as follows:

AdGuard DNS proxy It supports all existing DNS protocols including DoT, DoQ, and DoQ. Suppose we directly connect to the encrypted DNS caching resolver. In that case, we can use a simple upstream command in the client, for example, *./dnsproxy -u*

Table 2.3: The deployment of browser that supports DNS encryption

Browser	DoT	DoH	DoQ
Google Chrome (ver.78)		✓	
Mozilla Firefox (ver.62)		✓	
Apple Safari (ver.14)		✓	
Microsoft Edge (ver.79)		✓	
Opera (ver.56)		✓	
Brave		✓	

quic://dns.adguard.com (using DoQ protocol to connect AdGuard DNS resolver). It also supports the encrypted DNS server like that *./dnsproxy -l 127.0.0.1 –quic-port=853 –tls-crt=example.crt –tls-key=example.key -u 8.8.8.8:53 -p 0.*

In summary, DoT, DoH, and DoQ are encrypted DNS protocols to enhance security and privacy in network communication. There are two typical usages of encrypted DNS protocols:

(1) Stub resolvers (e.g., home routers) connect to DoT/DoH available caching resolvers with DoT/DoH;

(2) Web browsers (configure DoT/DoH in different browsers) directly connect to DoT/DoH available caching resolvers. DoT/DoH/DoQ rely on reliable connection-oriented transport, so each browser establishes the connection (with a unique 5-tuple) to the caching resolver.

2.2 Encrypted DNS Configurations

Encrypted DNS helps protect user privacy by encrypting DNS queries and responses between the end user and DNS resolver, preventing unauthorized parties from monitoring or intercepting DNS traffic. Depending on the previous section, We learned that not all DNS resolvers support encryption, especially DoQ. In our work, we intend to understand the privacy problem of encrypted DNS protocols (mainly DoQ) on the web with two kinds of DNS resolvers: public and local software. Regarding the tested recursive resolvers, we choose Google, NextDNS, and Bind. NextDNS has already deployed the DNS over QUIC, Google is a well-known resolver.

For the public DNS caching resolver that supports encrypted DNS, we can directly connect it and capture the traffic. However, we cannot control the cache in this case.

Since we want to examine the cache effect on the caching resolver, we should select the local DNS software that supports encrypted DNS. As shown in Table 2.2, we observed no local DNS software support the DoQ. In this case, we need the DNS proxy to help us to encrypt the DNS traffic with different underlying protocols. Therefore, we summarize two possible configurations to realize encrypted DNS. As shown in Figure 2.2 (a), the user visits the website, such as "example.com", and directly sends the DNS query to the recursive resolver, supporting DoQ/DoT/DoH. For example, AdGuard (dns.adguard.com) and NextDNS work as recursive resolvers. The traffic in this process is encrypted. Here, queries between the client and the resolver are encrypted.

The other configuration, as shown in Figure 2.2 (b), requires a proxy when the resolver does not support the encrypted DNS. In our paper, we use the AdGuard proxy as the DNS proxy. At first, the client establishes the connection to the DNS proxy by the TLS/SSL certificate. Once the connection is established, when the user visits the website, the encrypted query is first sent to the proxy. Then, the DNS proxy forwards the query to the recursive resolver. After the DNS resolver looks up the query and transfers the response to the DNS proxy. In the same way, the proxy encrypts the response and transfers it to the user. The proxy is just a forwarder without the cache. We can specify a recursive resolver to look up this query, such as Google Public DNS (non-support DoQ) or local resolvers.

On the client side, in either case, the traffic observed by the user is always encrypted. While visiting the website in the browser, we observe multiple DoQ packets in one session. One webpage could include multiple resources, such as browser information, images, and JavaScript files. Different websites generate different numbers of encrypted DNS queries. For example, some websites generate fewer packets due to a login screen. By contrast, a media website (i.e., youtube.com) can integrate some images and video, resulting in many DNS resolutions. In one session, these queries share the same source port. For the second configuration (b), we can control the cache in the local recursive resolver.

2.3 Related Works

There are three categories of related works. First, we describe the website fingerprinting attack of encrypted traffic in §2.3.1. Our threat model is similar to this

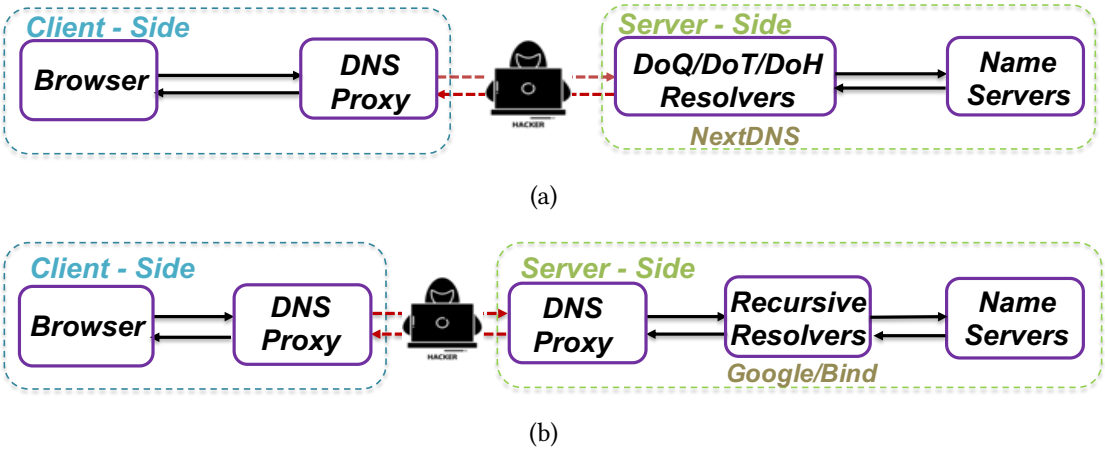


Figure 2.2: Two configurations to deploy the encrypted DNS in the network

attack, our work is based on previous approaches. Then, we list some methods for analyzing the encrypted traffic with machine learning techniques in §2.3.2. Our work focuses on the website fingerprinting attack of encrypted DNS traffic instead of the web. Therefore, we discuss the related works of encrypted DNS, such as deployment, adoption, detection, and performance analysis in §2.3.3.

2.3.1 Website Fingerprinting Attack

The adversary could monitor the network traffic between the victim and the web server, known as a website fingerprinting (WFP) attack. The adversary intends to classify the websites the victim visited by analyzing the traffic patterns. Previous works [34–40] analyzed the WFP attacks primarily based on the encrypted traffic. We list some related works of website fingerprinting in Table 2.4.

Marc and Brian [39] identified encrypted HTTP streams with two classification methods, one based on the Naïve Bayes classifier and one on Jaccard's coefficient. The flow features rely on the packet length, not time-related. Their work also examined the effects of packet padding with four methods: linear, exponential, mice and elephants, and MTU. They found the encryption is not enough to protect user privacy. Work from [40] investigated the website fingerprinting in onion routing-based anonymization networks like Tor and JAP. They selected the flow features based on the volume, time, and direction of traffic. Panchenko et al. [34] presented the website fingerprinting

approach, collected the data under the same domain, extracted the features such as packet ordering or burst behavior, and trained the set by SVM classifier.

In contrast to Panchenko, Cai et al. [35] selected the features related to the optimal string alignment distance and trained the model by the SVM classifier with a distance-based kernel. Dyer et al. [41] performed the analysis of low-level countermeasures (e.g., per-packet padding) for website identification attacks using flow features like timing, size, direction, and bandwidth. Other studies of Cai [36] presented a framework for assessing the performances of WFP attacks and defenses. This method was effective against the WFP attack in both open and closed-world models. Ref [42] demonstrated improved website fingerprinting techniques on Tor with SVM and distance-based metrics. Works from Wang et al. [43] proposed the multi-modal of web pages and calculated the distance with the KNN classifier. Hayes and Danezis [37] proposed a website fingerprinting technique, k-fingerprinting, using the Random Forest classifier. They found that the most valuable feature was the total number of incoming packets in the model. The Dolos [38] system against the website fingerprint attacks with the deep learning technique. It performed higher protection and lower information leakage and bandwidth overhead. The authors in Ref. [44] applied a novel fingerprinting technique to be used against privacy-enhancing technologies like OpenSSL, OpenVPN, CiscoVPN, and Tor with the Multinomial Naïve-Bayes classifier. They found the text mining technique had no protection against the WFP attack. These works showed that the WFP attack is possible against some privacy-enhance services, such as Tor, IPsec, and VPN.

Also, some encrypted protocols (HTTPS, TLS/SSL, and QUIC) have been proposed to protect the users' private information. Zhan et al. [4] studied the WFP attacks on the QUIC (GQUIC and IQUIC [1]) traffic and compared the classifier performance between QUIC and HTTPS in different scenarios. A work [3] aimed to find whether the TCP is more vulnerable than QUIC and which feature played an important role between the two protocols. These results indicated that information leakage is still possible even in some encrypted protocols (i.e., HTTPS, QUIC, TLS/SSL). In our work, we focus on the WFP attacks on the encrypted DNS traffic instead of web traffic and select the features of both incoming and outgoing packets, including time, packet count, packet size, and transfer bytes.

[1] GQUIC: Google QUIC; IQUIC: IETF QUIC

Table 2.4: Related works of website fingerprinting attack

Ref.	Traffic	Key metrics	Classifier
[39]	HTTPS	packet length	Naïve Bayes Jaccard
[40]	Tor, JAP	volume, time, and direction	SVM
[34]	Tor	packet ordering, size, burst behavior	SVM
[41]	HTTPS	time, size, direction, and bandwidth	Naïve Bayes, SVM
[36]	Tor, SSH	time, size, direction	SVM
[42]	Tor	upstream/downstream transmission	SVM distance-based metrics
[43]	Tor	transmission size, transmission time numbers, ordering, burst concentration of outgoing packets	KNN
[37]	Tor	ordering, inter-arrival time concentration of outgoing/incoming	Random Forest, KNN
[44]	OpenSSH, Tor OpenVPN CiscoVPN Stunnel JonDonym	packet size, direction	Jaccard, Naive Bayes, Multinomial Naïve Bayes
[3]	QUIC	packet size	DF, p-FP(C) Var-CNN
[4]	QUIC, HTTPS	packet size, count inter-arrival time cumulative size direction, burst	Random Forest Extra Trees, KNN Naïve Bayes, SVM
Our Work	Encrypted DNS	packet size, count inter-arrival time cumulative bytes, entropy throughput, duration	Random Forest AdaBoost, XGBoost LightGBM

2.3.2 Encrypted Traffic Analysis with Machine Learning Technique

Several works used packet-related features to detect the application in the encrypted traffic as shown in Table 2.5.

Wright et al. [45] detected the application classification on SSL/TLS and SSH tunnels using TCP packet features such as packet size, timing, and direction. Also, the work from [46] identified the application in SSL connections only used the size features of the first few packets with the Gaussian Mixture. In Ref. [47], the authors presented two empirical methods, signature-based and statistical analysis, to classify the application on the SSL/TLS and Tor flows with the Bayesian model. The work from Maiolini et al. [48] also developed a traffic classification method to identify SSH (SCP, SFTP, and HTTP over SSH) connections with IP-based features, like packet length, arrival time, and direction that relied on K-means cluster analysis. Meanwhile, some works focus on the effect of inter-packet time characteristics on traffic flow analysis. For example, Jaber et al. [49] presented the complete study of inter-packet time to classify Internet traffic using the K-means classifier. This work showed the results increase (from 80% to 98%) while selecting inter-packet time related features to detect the application.

Table 2.5: Related works of encrypted traffic analysis

Ref.	Traffic	Key metrics	ML Classifier
[48]	SSH	packet size, count inter-arrival time direction	K-means
[49]	SSL/TLS SSH	inter-arrival time	bayesian
[47]	SSL/TLS	packet length, count inter-arrival time duration	Naïve Bayes
[46]	SSL	packet size, number inter-arrival time	Naïve Bayes
[45]	SSL/TLS SSH	packet size, timing direction	KNN

2.3.3 Encrypted DNS: Deployment, Detecting, and Performance Analysis

Original DNS queries are unencrypted, sent in plain text, and vulnerable to eavesdropping, which leads to information leakage. Intending to protect the privacy of information, the DNS encryption techniques had been standardized. First, we present the current deployment of several encrypted DNS protocols in §2.3.3.1. Some work also detected encrypted DNS traffic from HTTPS stream in §2.3.3.2. Then, we show the performance analysis of encrypted DNS with different Network conditions and DNS resolvers in §2.3.3.3. Finally, we discuss the website fingerprinting attack on encrypted DNS and some countermeasures in §2.3.3.4.

2.3.3.1 Deployment of Encrypted DNS

Some papers [50–53] reported the recent deployment of encrypted DNS. Recent papers [50] described the deployment of DNS-Over-Encryption servers to help us to understand the DNS-over-Encryption techniques and made a comparison of different encrypted DNS protocols: DoT, DoH, DoQ, DNSCrypt [54], and DTLS.

A measurement work [51] presented the measurement, comparison, and analysis of the DoH, DoT, and DoQ in three global organizations, and DoQ was used in at least one organization. Deccio and Davis [52] reported 1,747 DoT and 9 DoH open resolvers in their measurement. They also characterized the deployment of DoT-based authoritative servers. In Ref. [53], authors overviewed the various encrypted DNS protocols, such as adaption status, performance, benefits, and security issues. This working group also surveyed the analysis for detecting encrypted DNS protocols by analyzing the encrypted DNS traffic. Works from [55] aimed to identify the DoH clients based on IP flows without IP addresses and ports. Authors of [56] studied the deployment of DoH resolvers by scanning the DoH resolvers on the whole Internet.

2.3.3.2 Detecting encrypted DNS traffic

DoT and DoQ can be easily detected by monitoring their header information: unique port number (DoT: TCP/853 and DoQ: UDP/853&784). However, DoH has the same port as HTTPS using TCP/443. Some work detected the DoH traffic from the

HTTPS stream using various flow characteristics.

Work from [55] intended to classify and recognize the DoH traffic with five machine learning algorithms: KNN, Decision Tree, Random Forest, Naive Bayes, and AdaBoost. They found that the accuracy of recognition DoH is over 99%. Ref [57] also extracted the features of packet size and inter-arrival time to build the machine learning model and detect the DoH traffic from the web.

2.3.3.3 Performance Analysis

Unlike unencrypted DNS, encrypted DNS makes it more difficult for an eavesdropper to get the information from the DNS message. Prior works were to determine whether it is possible to infer the websites the user visited from encrypted DNS traffic, like DoT and DoH. For example, Ref. [16] examined whether encrypted DNS traffic could protect users and analyzed the DoH traces with different environments (i.e., location, client application, platform, or DNS resolver). They found monitoring and censorship were feasible even using DoH, and some features used to attack HTTPS were not appropriate for DoH. In Ref. [17], the authors demonstrated that time and size were key features in reducing the classification performances on DoH traffic.

Recently, some studies reported privacy problems of encrypted DNS protocols and proposed solutions. For example, IETF [58] discussed DNS privacy problems between recursive resolvers and authoritative servers. Shulman [59] collected DNS traffic of 50K-Top Alexa domains and 568 TLDs. This work mainly considered the privacy guarantees and the effect on the overhead and examined the privacy leaks from the transitive trust. Hence, some studies proposed adding DNS padding to solve the privacy problem. For example, the work of [60] analyzed the privacy over DoH, chose Firefox to connect DoH resolvers, and selected features such as packet length, destination port, and destination IP. They suggested adding padding to the DNS queries to enhance privacy.

In another study [61], the authors concentrate on the effects that Do53 [62][2], DoT, and DoH on web performances, although they did not investigate privacy problems. They measured the impact of these three protocols on query response time and page load time in web browsers. They also found that DoH and DoT were significantly higher

[2]Traditional DNS over port 53.

than Do53 in terms of response time. However, encryption protocols perform better than Do53 regarding page load time. Hoang et al. [63] quantified the improvement to user privacy benefits of DoH, DoT, and ESNI (Encrypted Server Name Indication) using the k-anonymity model. Nguyen et al. [64] presented the privacy leakage of encrypted DNS via IP-based WFP attacks. With IP-based fingerprints, the authors could still identify 84% of the website even considering the browser cache and AdBlocker. Authors of [65] showed the limitation with DoH and ESNI to protect the users' privacy. The results indicated that eavesdroppers could classify 80% of domain names with a higher (0.8) F1 score in the campus network. In [66], the authors propose how to protect privacy using the extended Berkeley Packet Filter (eBPF) across the standard DNS, DoH, and DoT. Jin et al. [67] studied the effect of encrypted DNS (3,813 DoT and 75 of DoH resolvers) on Internet censorship from the vantage points. They characterized the use of encrypted DNS resolvers to circumvent censorship varies due to country. The authors of [68] studied the threshold-based attacks and defenses on DoH traffic, using browser-generated packet size, rate, and throughput from vantage points. They presented that encrypted DNS is still vulnerable when faced with some attacks.

2.3.3.4 Website Fingerprinting Attack on encrypted DNS: Analysis and Countermeasures

RFC 8467 [18] proposed the padding strategies to prevent information leakage on encrypted DNS. Prior works mainly studied the WFP attack on DoT and DoH as shown in Table 2.6.

In Ref. [16], authors analyzed the WFP attacks of DoH traffic using three groups of features: size, timing, and ordering on Google and Cloudflare DNS resolvers. They found the adversary could still infer the categories of websites even though DoH and the standardized padding schemes are not effective in preventing traffic analysis attacks. The work of [15] analyzed the WFP attack on DoT and compared the classification performances with padded or unpadded DNS.

The results suggested that encrypted DNS messages should be padded to protect privacy. K. Hynek et al. [69] indicated using EDNS padding extension against the website fingerprinting attack on DoH could reduce the accuracy to 17.24% (HTTP 2) and 10.73% (HTTP 1.1). They also showed DNS padding was not widely deployed in

some platforms, like the Firefox browser.

However, the work from Jonas et al. [70] measured the impact of padding strategies (128 B / 468 B block padding) on DoT and DoH. They observed padding cannot mitigate the WFP attack on DoH and DoT. Their results also presented better performance to protect users by removing the entropy of inter-arrival time between query and response packets. These works demonstrated that information leakage is still possible in encrypted protocols (HTTPS, QUIC, and DNS-Over-Encryption).

In this thesis, we examine the information leakage of three encrypted DNS protocols, especially DoQ. We also analyze several aspects potentially affecting the identification, such as DNS cache and crawling number. Furthermore, we present four countermeasures to mitigate the website fingerprinting attack on DoQ: using AdBlocker, disabling DNS prefetch, adding random delay, and padding the DNS payload.

2.4 Summary

Most existing works analyzed the website fingerprinting attack on encrypted traffic (i.e., Tor, JAP, and HTTPS) using flow features such as packet length and inter-packet time. Compared to traditional DNS, these encrypted DNS (DoT, DoH, and DoQ) have been proposed to protect DNS information using TLS, HTTPS, and QUIC to encrypt the metadata of DNS. They demonstrated that information leakage is still possible even in some encrypted protocols (i.e., HTTPS, QUIC, TLS/SSL, DNS-encryption). Also, the help of padding strategies is limited. From there, improving and developing countermeasures to protect users' private information against the website fingerprinting attack on encrypted DNS, including controlling time distribution and packet length.

Table 2.6: Comparison with other related works of website fingerprinting attack on encrypted DNS

Ref.	Protocols	Resolver	Key metrics	Classifier	Countermeasures
[16]	DoH	Google Cloudflare	Packet length time ordering	Random Forest	Padding
[15]	DoT	Google Cloudflare	Packet length time ordering cumulative bytes	Decision Tree Naïve Bayes Simple Logistic SMO Random Forest	Padding
[70]	DoH DoT	Google Cloudflare Quad9 unbound	Packet length time transfer bytes	KNN Neural Network	Overhead Padding
[69]	DoH	Google Cloudflare NextDNS	packet size	KNN Decision Tree	Padding
Ours	DoQ DoH DoT	Bind NextDNS Google	packet size count inter-arrival time cumulative bytes entropy throughput duration	Random Forest AdaBoost XGBoost LightGBM	Using AdBlocker Disabling Prefetch Adding Random Delay Padding

3 Methodology

In this chapter, we first describe two threat models of encrypted DNS to simulate the website fingerprinting attack, depending on DNS software configuration and implementation in §3.1. Then, we show the process of the machine learning approach for the website fingerprinting attack on encrypted DNS §3.2, including dataset collection, category selection, feature extraction, and classifier selection.

3.1 Measurement Setup

Here, we consider the threat scenario as shown in Figure 3.1, in which a user uses encrypted DNS to protect DNS privacy, and the adversary intends to infer the user's activities from analyzing the encrypted DNS traffic. Our threat model consists of three components; web browser (client), recursive resolvers, and authoritative servers. We connect the client and server on the home router with Wi-Fi (uplink: VDSL 100Mbps). We assume the adversary can monitor the traffic between the client (victim) and the DoH/DoT/DoQ available recursive resolver. However, it is difficult for the adversary to

monitor and collect the traffic between the recursive resolver and the authoritative nameserver. If the encrypted DNS resolver acts as the proxy between the client and the authoritative name servers. The DNS caching resolver receives encrypted DNS queries from the client, decrypts them, performs traditional DNS resolution to communicate with authoritative name servers, and then returns the results to the client over HTTPS. This scenario is easier for the attackers than web traffic monitoring, especially for high-speed networks. The work of [44] reported this kind of adversary exists in the real world. In our model, we observe the encrypted DNS queries generated from a single browser (a user, not aggregated by many users) by considering the typical deployment scenario of the encrypted DNS as described in §3.2.

Since we select two kinds of DNS recursive resolvers that support and non-support encrypted DNS, we consider two models to simulate the website fingerprint attack:

1. We set up the NextDNS that supports DoQ protocol on the client to encrypt DNS packets.

2. While Google and Bind do not support the DoQ, we set up the AdGuard DNS proxy [33] on the client to encrypt DNS packets and Raspberry Pi (Pi3 Model B+) as the forwarder. Then, the unencrypted DNS queries are forwarded to a public (i.e., google public DNS) or local recursive resolver (i.e., Bind9 [23]).

On the client side, we use the popular browser: Firefox [71]. We capture the encrypted DNS traffic between the client and the recursive resolver.

In this thesis, we investigate the impact of several aspects of encrypted DNS on privacy leakage as shown in Table 3.1 using two configurations in Figure 2.2.

Table 3.1: Summary of measurement setup on encrypted DNS

Protocol	**Resolver**	**Configuration**	**Section**
DoQ	Bind	Conf. a	§4.1.1, §4.1.3, §4.2, §5.1.1, §5.1.2, §5.1.3, §5.2, §5.3
DoQ	NextDNS	Conf. b	§4.1.1, §5.1.1, §5.1.2, §5.1.3
DoQ	Google	Conf. a	§4.1.4
DoT/DoH	Google	Conf. a	§4.1.5

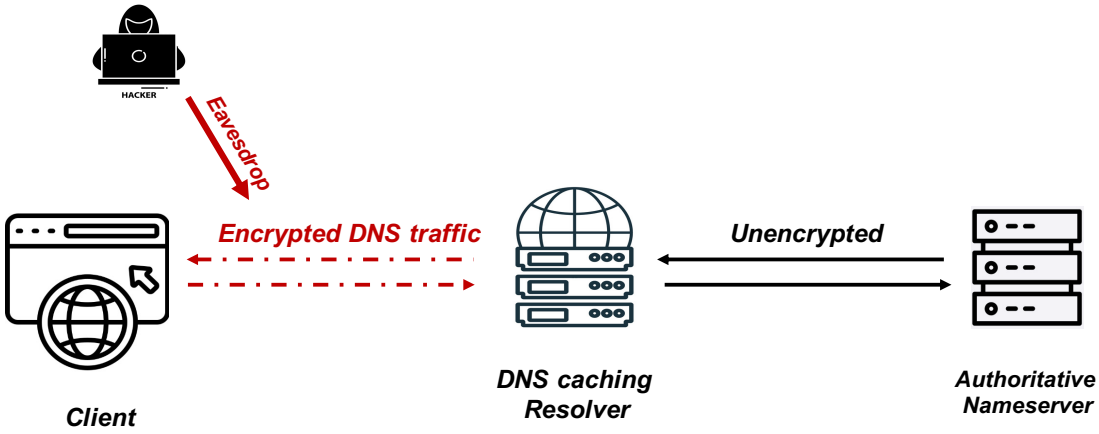


Figure 3.1: Threat Model: passive eavesdropper between a user and recursive resolver

3.2 Overview

Here, we describe four supervised machine learning algorithms to identify the category of websites. Figure 3.2 shows an overview of our method consisting of four steps: data preparation, feature extraction, model building, and evaluation. The basic procedures of this approach are as follows:

1. Capture the encrypted DNS traffic of 30 categories and divide the datasets into two sets ('*Sensitive*' and '*Non-Sensitive*') in §3.2.1;

2. Extract flow features in §3.2.2;

3. Adopt the 10-fold cross-validation, find the best parameter set for each classifier to build a model and evaluate the mean F1 score in §3.2.3;

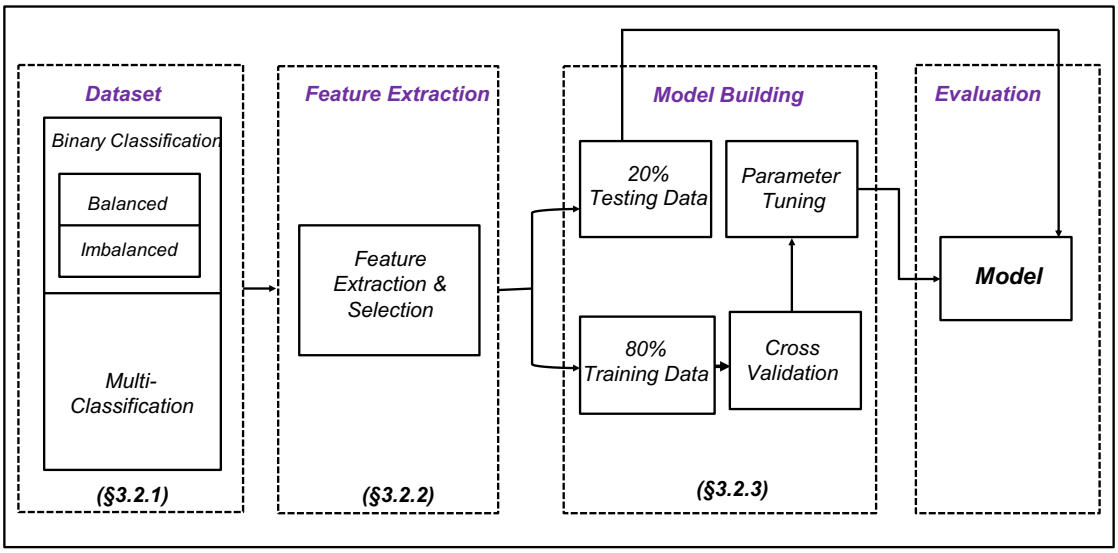


Figure 3.2: Overview of machine learning approach for website fingerprinting attack on encrypted DNS

3.2.1 Dataset Preparation

We choose 30 [1] categories of websites and determine the category of the websites based on the FortiGuard Web Filtering [72]. Then, we select the Top-400 for each category popular in Alexa's Top 300,000 [73] websites Alexa top list is created by Amazon, which ranks websites based on popularity on a global scale. from January 2020. About this list, it only shows the domain names [2] of the website, for example, example.com. We build a Python script to access the website in the list automatically, and only one website is loaded each time on the client side. For each website, we only direct the Firefox [71] browser to visit its homepage. While the access is complete (timeout 30s), we close the browser and save the traffic data as .pcap files. If a target webpage shows the cookie notification, we ignore (do not touch the button) it in the experiment.

[1]The categories are: Advertising, Art, Brokerage and Trading, Business, Dating, Domain Parking, Education, Entertainment, File sharing and Storage, Finance and Banking, Freeware and Software Downloads, Gambling, Game, General Organizations, Global Religion, Government and Legal Organization, Health and Wellness, Illegal or Unethical, Information Technology, Internet Radio and TV, Job Search, Malicious Websites, Meaningless Content, Newly Observed Domain, News and Media, News groups and Message boards, Other Adult Materials, Personal Vehicles, Phishing, Political Organizations.

[2]We don't consider the sub domains such as www.example.com, and www.example.com/sub/.

In this process, we use Selenium [74] webdriver to drive the Firefox (ver. 117.0.1) browser on the laptop (macOS Ventura ver. 13.6.7). For each website, we access its homepage and capture the encrypted DNS traffic. We treat the same domain (different TLD) as one sample, i.e., if example.com and example.net are both in the list, we prefer to choose a more popular one in the Alexa ranking. Also, we remove the ones that duplicate in different categories (at least two), to ensure no overlap. The complete process is as follows:

1. Start running the tcpdump tool.
2. Selenium starts the browser process and loads the webpage on the client.
3. Close the browser while the access is complete (timeout 30s) and save the traffic data as .pcap files.
4. Stop tcpdump after the loading, and wait three seconds for the next website.

After we prepare the targeted datasets, we conduct binary and multi-classification depending on the attacker's targets. About the binary classification, the attackers focus on distinguishing between two classes, *'Sensitive'* vs. *'Non-Sensitive'* in our work. This scenario makes it easier for attackers to develop and deploy their attack measurements. The binary classification is easier but less details than the multi-classification. But still, an attacker can dig into the target if he detects *'sensitive'* web browsing. The multi-classification allows attackers to distinguish between multiple categories of websites, and get more information from users' activities. Multi-classification can significantly increase the benefits of the attack. Some studies [15] also conducted experiments on multiple categories, like dating, finance, and health. However, this increased complexity may provide attackers with more valuable information, enabling them to analyze users' behavior and preferences better.

We split the dataset into two labels: *'Non-Sensitive'* and *'Sensitive'*. We determine the category related to personal information such as health, finances, and religion as sensitive followed by past literature [15]. For example, websites in the health category may indicate that users intend to access to obtain medical information for some specific symptoms. Similar to past literature [15], we select dating, health, and gambling categories as the sensitive. In addition, we select seven (Finance and Banking,

Global Religion, Government and Legal Organization, Illegal or Unethical, Other Adult Materials, Phishing, and Political Organization) categories as sensitive based on the description of the category by FortiGuard Web Filtering.

Especially, we evaluate the binary classification performance of balanced and imbalanced datasets. For the balanced dataset, we select all 10 categories from *'Sensitive'*, and 10 categories from *'Non-Sensitive'* randomly. For the imbalanced dataset, we select 10 categories as sensitive, and the remaining 20 categories as non-sensitive as shown in Table 3.2. The imbalanced dataset refers to a dataset in the different classes with skewed distribution. This means that one class (the minority class) has significantly fewer examples than another class or classes (the majority class or classes). The imbalanced datasets are closer to many real-world scenarios. In particular, we evaluate the binary classification performance of this imbalanced dataset with the help of the oversampling technique, SMOTE [75].

SMOTE Synthetic Minority Over-sampling Technique, is a popular technique to address the issue of class imbalance in datasets. SMOTE increases the number of minor class examples in the dataset. This helps address the imbalance and allows classifiers to learn from more balanced data, potentially improving their performance in predicting the minority class It could help to reduce over-fitting and improve classification performance.

Next, we consider the multi-classification of all 30 categories with different combinations. We randomly choose different category combinations; for example, 2 - 2 means select two categories from *'Sensitive'* and two categories from *'Non-Sensitive'*. We list all combinations in Table 3.3.

[3]Some websites in Phishing and Malicious Websites are usually rapidly shutdown. In this case, we skip this website and get the next one on the list.

[4]Phishing websites are still available on the internet. Phishing websites are often created by attackers who intend to deceive users into providing sensitive information such as usernames, passwords, credit card numbers, or other personal data. When the user attempts to visit phishing websites, the browser will show a warning message that indicates the website might be dangerous.

[5]focuses on malicious activity like spreading malware, viruses, and other malicious software designed to infect and damage computers or steal data that don't necessarily involve sensitive data.

Table 3.2: Binary class dataset

	Non-Sensitive	***Sensitive***[3]
Balanced Data	*Business/Education/Entertainment/ File sharing and Storage/ Job Search/Newly Observed Domain/ News and Media/Personal Vehicles/ General Organizations/ Internet Radio and TV*	*Dating/Gambling/ Finance and Banking/ Global Religion/ Government and Legal Organization/ Health and Wellness/ Illegal or Unethical/ Other Adult Materials/ Phishing*[4]*/ Political Organizations*
Imbalanced Data	*Advertising/Art/Brokerage and Trading/ Business/Domain Parking/Education/ Entertainment/File sharing and Storage/ Freeware and Software Downloads/ Game/General Organizations/ Information Technology/Job Search/ Malicious Websites*[5]*/Meaningless Content/ Internet Radio and TV Newly Observed Domain/News and Media/ News groups and Message boards/ Personal Vehicles*	

3.2.2 Features

Features are variables or attributes that describe the instance of the class in machine learning. From the adversary's view, only the packet size and timestamp for each query/response could be monitored due to the encrypted traffic. The destination port and IP address belong to the DNS recursive resolver, we use this information to identify the encrypted DNS traffic.

We extract 175 bidirectional flow features in traffic traces such as packet number, inter-arrival time, and packet length between consecutive DNS queries and responses,

[6]it typically involves general, publicly available information about organizations, industries, and economic activities.

[7]typically refers to specific activities related to buying and selling financial instruments (e.g., stocks and bonds) on behalf of clients. This category focuses on the operational (e.g., transaction processing) not private information.

[8]including more financial activities not only the banking, insurance but also financial information including personal data, credit history and more. The inclusion of personal financial details in financial related activities and the possibility of sensitive information to be classified as sensitive data.

Table 3.3: Multi-class dataset

Combinations	***Non-Sensitive***	***Sensitive***
2 - 2	*Business*[6]*; Game*	*Phishing; Political Organizations*
3 - 3	*Entertainment; File sharing and Storage; News and Media*	*Dating; Gambling; Health and Wellness*
4 - 4	*Art; Domain Parking; Internet Radio and TV; Personal Vehicles*	*Dating; Finance and Banking; Illegal or Unethical; Phishing*
5 - 5	*Education; Freeware and Software Downloads; Information Technology; Job Search; Newly Observed Domain*	*Global Religion; Government and Legal Organization; Illegal or Unethical; Other Adult Materials; Political Organizations*
6 - 6	*Advertising; Brokerage and Trading*[7]*; Freeware and Software Downloads; Malicious Websites; Meaningless Content; News groups and Message boards*	*Finance and Banking* [8]*; Global Religion; Government and Legal Organization; Health and Wellness; Illegal or Unethical; Political Organizations*
7 - 7	*Domain Parking; Education; Entertainment; File sharing and Storage; Freeware and Software Downloads; Game; Job Search*	*Dating; Finance and Banking; Gambling;Global Religion; Government and Legal Organization; Health and Wellness; Other Adult Materials*
8 - 8	*Advertising; Brokerage and Trading; Business; Domain Parking; Education; Entertainment; File sharing and Storage; Freeware and Software Downloads*	*Finance and Banking;Global Religion; Government and Legal Organization; Health and Wellness; Illegal or Unethical; Other Adult Materials; Phishing; Political Organizations*
9 - 9	*Game; Information Technology; Internet Radio and TV; Job Search; Malicious Websites; Meaningless Content; Newly Observed Domain; Personal Vehicles*	*Dating; Finance and Banking; Global Religion; Government and Legal Organization; Health and Wellness; Illegal or Unethical; Other Adult Materials; Phishing; Political Organizations*

mainly followed by the past literature [3, 53, 68]. In particular, we calculate the maximum, minimum, median, mean, standard deviation, variance, coefficient variation, and deciles [9] for each feature that related to packet number, packet length, and inter-arrival time. We list broad categories of traffic features as follows (all features listed in Appendix A):

Packet count: The number of incoming and outgoing packets. While we access the websites, the homepage includes different resources like images, JavaScript files, advertisements, and videos. This will be resulting different numbers of DNS packets. For example, if we visit youtube.com, it requests many DNS resolutions. By contrast, one finance or government website may request a few DNS because the page content just includes bank information or government announcements.

Query and response length: The transfer length (bytes) of DNS query and response packets. Original DNS messages consist of the packet header and resource records. Encrypted DNS protocols have different additional encryption overhead. For example, DoQ encrypts the traffic with QUIC, and the packet length includes the QUIC header, padding, and other metadata for encryption.

Inter-arrival time: The features from a consecutive pair of query and response packets, for example, query-to-query, query-to-response (see more details in Appendix A.2), response-to-response. It includes the maximum, minimum, median, mean, standard deviation, variance, coefficient variation, and deciles of the inter-arrival value.

Cumulative bytes: The cumulative bytes of query and response packets. We also consider the rate of bytes being sent and received in a trace.

Entropy value: Shannon entropy is commonly used as a measure of uncertainty or disorder in a dataset or the distribution of class labels. Given a discrete random variable X, for a probability distribution p(X = x_i), the Shannon entropy (base 2 gives the unit of bits) of X is defined as:

$H(X) = -\sum_{i=1}^{n} p(x_i) log_b p(x_i)$

We consider Shannon's normalized entropy [10] of inter-arrival time and packet length.

Throughput: The average in packets and bytes.

Duration: The total transmission time in the trace.

[9] Deciles are often used as a way to divide a dataset into ten equal parts based on the values of a particular variable.

[10] the normalized of Shannon entropy, that range in [0, 1.0]

Time to receive first *N* bytes: As in [68], the time that is received from the DNS resolver could reflect whether the resolver cached. We set the value of N to 3000 and 5000.

Queries per Second: The total transmission time in the trace.

In the final step, we remove the features with zero variance.

3.2.3 Modeling

3.2.3.1 Classifiers Selection

We split the datasets into training and testing to evaluate the classification performance. Then, we use the dataset to train the supervised machine learning models we prefer and evaluate the model with F1 scores.

We randomly split 80% of the dataset as a training set and 20% as a testing set. We also adopt the stratified 10-fold cross-validation [76] method and repeat it ten times for each test to evaluate the performance of our experiments. We train all binary classifiers and tune parameters using the RandomizedSearchCV [77] function from the scikit-learn library [78]. The RandomizedSearchCV function can help us to find the best estimator for each classifier.

We obtain the mean F1 score to evaluate the performance of each classifier. A higher value of the F1 score (close to one) means better, and a lower (close to zero) indicates a worse classification performance. In our context, a lower F1 score is desirable because the attacker cannot infer the website categories precisely. We use well-known four supervised classification algorithms; Random Forest [79], AdaBoost [80], XGBoost [81], and LightGBM [82]. We list these four supervised machine learning and F1-measure as follows:

Random Forest Classifier Random Forest algorithm constructs a "forest", and multiple decision trees and merges them to get a more accurate and stable prediction.

AdaBoost Classifier AdaBoost is one of the boosting ensemble algorithms. The key idea is to train different classifiers (weak classifiers) for the same training data and then assemble these classifiers to build a strong learner. While building the classifier,

the one most important parameter of this algorithm is the base_estimator. In our experiment, we select the decision tree and random forest as the base estimator.

XGBoostClassifier XGBoost (Extreme Gradient Boosting) is another popular open-source gradient boosting ensemble algorithm (boosting performs better than bagging on average), which is an advanced version of the gradient boosting algorithms. Like LightGBM, it is based on the decision tree algorithm.

LightGBM Classifier LightGBM (Light Gradient Boosting Machine) is open-source, uses the GOSS technique to perform gradient-based sampling of instances during tree construction, and is based on decision tree algorithms to build ensemble models.

3.2.3.2 Performance Measures

Performance metrics depend on the model's goals in machine learning. We use the F1 score and confusion matrix to measure the classification performance in the experiment. We also examine the important features by calculating the gini impurity with the Random Forest classifier.

F1 score It combines precision [11] and recall [12] into a single value, providing a balanced measure of a model's performance. We calculate the micro-average of the F1 score by considering the total number of true positives, false negatives, and false positives across all classes. It treats all classes equally and is useful when weighing each instance or prediction equally, regardless of class imbalance.

The range of the F1 value is between 0 and 1.

It is usually defined as:

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

[11] Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It is calculated as the number of true positives divided by the sum of true positives and false positives.

[12] Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. It is calculated as the number of true positives divided by the sum of true positives and false negatives

In general, higher (close to 1) F1 scores are better because they indicate better overall performance of the classification model. However, the higher F1 means the attacker obtains a better performance for inferring the victim's private information in our work. A lower (close to 0) value means better encryption to protect users. We hope to reduce the F1 score for mitigation in our experiment.

Confusion Matrix A confusion matrix is a table that summarizes the performance of a classification model by comparing predicted labels with true labels. It provides information about true positives, true negatives, false positives, and false negatives, which can be used to calculate other metrics such as accuracy, precision, and recall. We also use it to measure our countermeasures.

Gini Impurity Gini impurity [83] is a metric used to generate a tree-based classification, particularly in binary classification. It provides more information about the distribution of data per node and less information about classification accuracy for reporting tree accuracy. If we have k classes, the Gini Impurity G is calculated as:

$G = 1 - \sum_{i=1}^{k} p_i^2$

p_i is the probability of elements belonging to class i. The range of gini impurity is from 0 to 0.5. When all p_i are 0, the gini impurity will be 0. When the values of p_i are all equal, the gini impurity is maximized at 0.5. A gini impurity of 0 indicates that is pure, all elements belong to the same class. A gini impurity of 0.5 indicates maximum impurity, the elements are evenly distributed among all classes. In our work, we measure the top-10 important features by calculating the gini impurity. Then, we show countermeasures to mitigate the attack with the important features that affect the performance significantly.

4

Classification Performance

In this chapter, we first evaluate the binary classification performance with three encrypted DNS protocols: DoQ, DoT, and DoH on three DNS caching resolvers: Bind, NextDNS, and Google. As the baseline analysis, we examine the DoQ traffic with two DNS recursive resolvers: Bind and NextDNS in §4.1.1. Specially, we show the top-10 important features and compare the value with these two DNS resolvers in §4.1.2. Then, we demonstrate the effect of cache including caching order and with/without cache with Bind in §4.2. We show the classification performance of DoQ with another public DNS resolver: Google in §4.1.3. We also investigate the binary classification performance of other encrypted DNS protocols: DoT and DoH in §4.1.4. Finally, we investigate the multi-classification performance for DoQ traffic on Bind with two configurations in §4.2.1 and discuss the effect of crawling numbers in §4.2.2.

4.1 Binary Classification Performance

Here, we evaluate baseline classification performance for DoQ traffic with balanced and imbalanced datasets on Bind and NextDNS caching resolvers. For the Bind resolver, we crawl the encrypted traffic without cleaning the cache in the Bind. Specifically, we discuss the F1 score of the classification and the resulting top 10 important features. Note that a lower F1 score is better for mitigating information leakage in our analysis.

4.1.1 Baseline Classification Performance

As shown in Figure 4.1 (left), overall the binary classification performances of Bind are very high for the four classifiers, in the balanced and imbalanced dataset. The performance of the imbalanced dataset is slightly higher than the balanced dataset, 5-7% more. This result indicates that the SMOTE technique works well. Thus, these classifiers are good enough to infer the website for the local resolver.

Next, we select NextDNS as the DNS resolver. Figure 4.1 (right) shows the results of NextDNS. We again find a high F1 value; the F1 scores are all around 0.90 with imbalanced and balanced data. The F1 score of the imbalanced dataset is still slightly higher than the balanced dataset, with only a 2-3% difference, except for the XGBoost classifier.

Comparison between Bind and NextDNS, we also notice that the classification performances of Bind are slightly higher than NextDNS because the NextDNS server is publicly available. Thus, we demonstrate that the classification performance is high enough for the attacker to infer the website regarding DoQ, whether on NextDNS or Bind resolver.

4.1.2 Feature Importance

We intend to understand what features have the most influence on the results for discussing possible countermeasures. Understanding feature importance helps us to identify the relative significance or contribution features, and potentially improve the model's performance. We measure the feature importance by calculating the mean degradation of the gini impurity with the Random Forest algorithm. To investigate the discriminative powers of the feature, we repeat ten times to obtain the average value

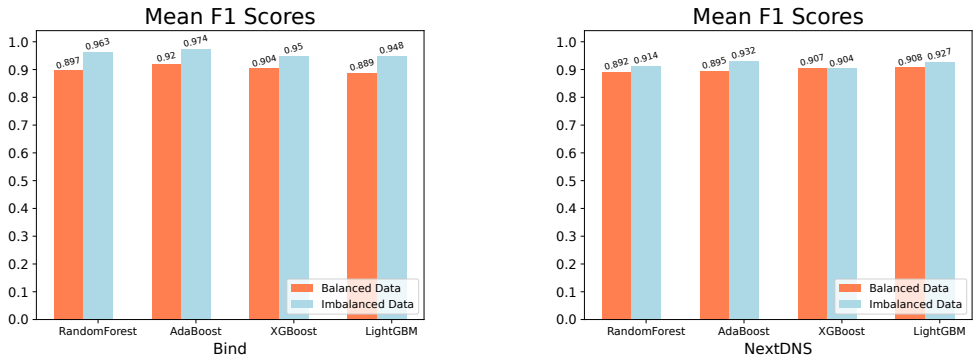


Figure 4.1: Classification performance (Bind and NextDNS)

for each feature. The range of features important is 0 to 1, the higher value means more influence on the classification.

We show the top-10 discriminative features of balanced (Table 4.1) and imbalanced (Table 4.2) datasets with Bind and NextDNS resolvers. We find that significant features are mainly related to inter-arrival time, regardless of Bind and NextDNS. We notice another feature also in the top-10 list, related to packet length. We find the mean value feature importance of NextDNS is higher than Bind, regardless of the balanced and imbalanced dataset. We also observe that the important features are different between these two DNS caching resolves. This is likely because of the different geographical locations between the client and the resolver. NextDNS is a public DNS resolver that can be located closer to the Internet core, though Bind is the local resolver in the home network. Also, NextDNS is publicly available and shared with other users, thus more random processing/queueing delays could be added. Thus, regardless of the DNS software, we should decrease the classification performance (F1 score) by controlling these effective features, to protect users' privacy.

4.1.3 Effect of cache on Bind

Here, we intend to understand whether the cache affects the classification performance. Since we cannot control the DNS cache in the public resolver, we select the Bind in this experiment. The DNS cache is a temporary storage that contains the information (DNS lookups) of recent user visits.

Table 4.1: Top-10 discriminative features （Balanced Dataset）

Rank	Bind		NextDNS	
	Feature	**Importance**	**Feature**	**Importance**
1	EntropyQRIntervalTime	0.022	QueryIntervalTimeDeciles1	0.086
2	EntropyRRIntervalTime	0.018	QueryIntervalTimeMedian	0.082
3	EntropyQQIntervalTime	0.016	ResponseIntervalTimeMin	0.070
4	QueryIntervalTime	0.015	QueryIntervalTimeMin	0.069
5	EntropyResponseLength	0.011	ResponseIntervalTimeMedian	0.068
6	EntropyDFIntervalTime	0.0103	QRIntervalTimeMean	0.060
7	RQIntervalTimeDeciles3	0.0103	QueryIntervalTimeMean	0.057
8	RQIntervalTimeDeciles4	0.0102	ResponseIntervalTimeMean	0.056
9	ResponseIntervalTimeMin	0.010	QRIntervalTimeMedian	0.043
10	QueryIntervalTimeMin	0.009	QRPktLength	0.038

4.1.3.1 Caching Order

First, we crawl the encrypted traffic data without cleaning the cache in the local recursive resolver (Bind). We examine whether the order of crawling affects the classification results in the local resolver. For example, if we first check youtube.com, then google.com, youtube.com information is cached in the resolver, and vice versa. In this experiment, we crawl the 30 categories twice. We compare the classification results with two different ordered Alexa lists; the Top list and the Randomized list. For the first time, we crawl the websites from the Alexa top-400 list per category by ascending order. For the second time, we randomize the order of websites in the Alexa list and crawl them.

As shown in Figure 4.2, we confirm that the order of caching does not affect the performance more for the imbalanced dataset. After the second crawling, we also notice there is no significant reduction compared to the first crawling, all higher than 0.93. Except for the random forest classifier, the degradation of F1 score is 6%. However, we note no significant change between these two cases in Bind. Thus, the caching order has less effect on mitigating the attack.

Table 4.2: Top-10 discriminative features （Imbalanced Dataset）

Rank	Bind		NextDNS	
	Feature	**Importance**	**Feature**	**Importance**
1	QRIntervalTimeMean	0.024	ResponseIntervalTimeMin	0.082
2	QRIntervalTimeMedian	0.018	QueryIntervalTimeMedian	0.078
3	EntropyDFIntervalTime	0.016	QRIntervalTimeCoefficientVariation	0.068
4	QRIntervalTimeMin	0.014	QueryIntervalTimeVariance	0.057
5	RQIntervalTimeDeciles4	0.012	ResponseLengthDeciles1	0.054
6	RQIntervalTimeDeciles3	0.0114	QRIntervalTimeMedian	0.053
7	QueryIntervalTimeStandard	0.0113	QRIntervalTimeMin	0.050
8	EntropyRRIntervalTime	0.0104	QRIntervalTimeMean	0.049
9	ResponseLengthMax	0.01001	ResponseIntervalTimeMedian	0.049
10	QueryLengthDeciles3	0.009	QueryIntervalTimeMax	0.047

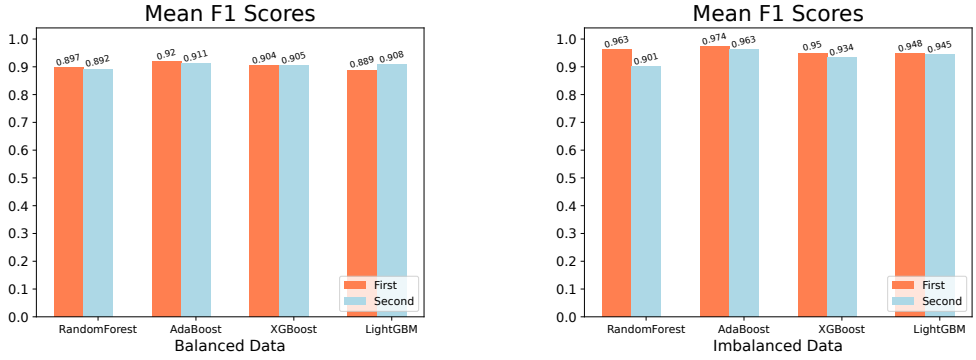


Figure 4.2: Classification performance (Effect of caching order with Bind)

4.1.3.2 Comparison between Caching and No-Caching

Here, we clean the cache after capturing the traffic per category on the resolver side to see the effect of the resolver's (Bind) cache.

Figure 4.3 shows the classification performance of cache and no-cache scenarios on bind. The classification performances are all around 0.9, regardless of cache and non-cache. Particularly, we notice that the classification performance of cache is slightly higher (2%) than the no-cache case. We find no significant differences between caching and no-caching cases on bind. Thus, the results indicate no significant influence on whether the local resolver is cached.

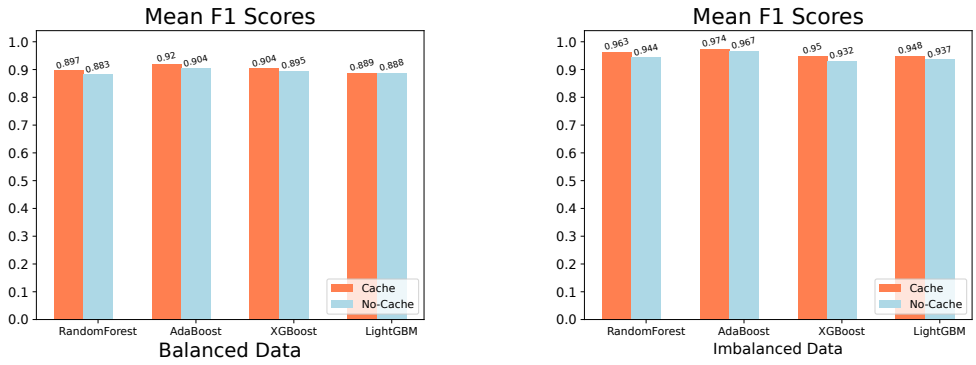


Figure 4.3: The classification performance of Cache and No-Cache

4.1.4 DoQ with another Public (Google) Resolver

Here, we evaluate the classification performance of DoQ with another DNS resolver; Google, using the first configuration (Figure 2.2 (a)).

As shown in Figure 4.4, we find that for the Google resolver, no matter which classifier is used or the dataset, the accuracies are all higher than 0.90. Then, we compare the classification performance with that using NextDNS and Bind. We notice that the performances are slightly higher than the other two DNS resolvers. For example, about the balanced dataset, the mean F1 score of NextDNS with Adaboost is 0.895, compared to 0.92 for Bind and 0.947 for Google resolver. We also observe no significant difference between the NextDNS, Bind, and Google DoQ protocol on Firefox. Thus, regardless of different DNS resolvers, users' privacy is likely to be exposed.

4.1.5 DoT and DoH with Public (Google) Resolver

To measure the impact of DoT and DoH, we select AdGuard proxy to connect public (Google) resolver to be consistent with the DoQ measurements.

Figure 4.5 shows F1 scores are still around 0.9, regardless of the balanced and imbalanced dataset. The accuracies of imbalanced datasets are also higher than balanced. In particular, we note that no big difference between these two protocols. Thus, the results demonstrate that user privacy leakage is still possible even in all the encrypted DNS regardless of the underlying protocols.

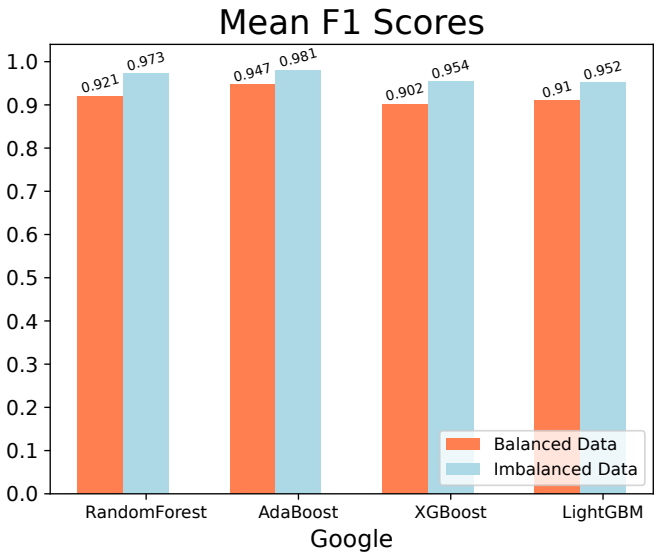


Figure 4.4: Classification Performance of Google Resolver with Binary Class

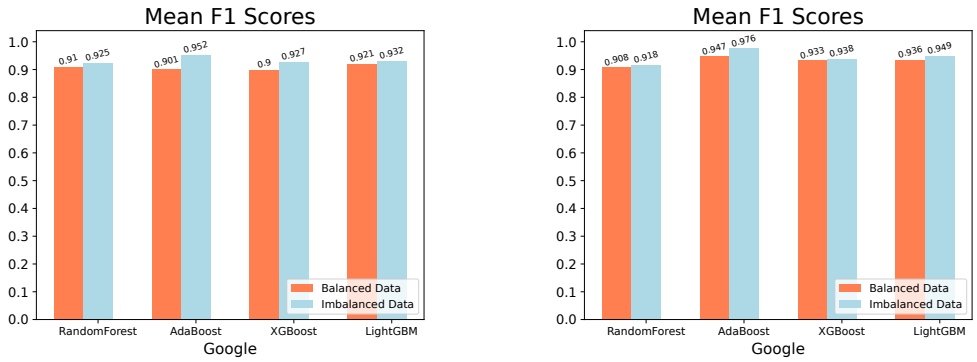
4.2 Multi-Classification Performance

The previous section shows that the binary classification performances are still high for the three encrypted DNS protocols. Here, we evaluate the performance of multi-classification and intend to understand the effect of the number of crawlings.

4.2.1 Baseline Classification Performance

We evaluate the classification performance of different combinations of 30 categories on Bind. We choose different combinations of categories (from 2/2 to 9/9) and select one combination randomly for each kind as described in Table 3.3. Then, we compare the classification performance with different combinations.

As shown in Figure 4.6, we notice that the F1 score decreases more as the number of categories increases. For the 9/9 combinations, the performance is lowest (≈ 0.4), especially 0.3 for the LightGBM classifier. Thus, we demonstrate that the multi-classification performance makes it hard for the attacker to infer the category of the website on DoQ.



(a) DoH (b) DoT

Figure 4.5: Classification Performance of DoT and DoH with Google

4.2.2 The Effect of Number of Crawls

We intend to understand the effect of the number of crawling on the Bind resolver. In the experiment, we select the 2 (*Sensitive*) - 2 (*Non-Sensitive*) combination and crawl the traffic of 30 categories five times. First, we randomly select two categories from the 'Sensitive' and 'Non-Sensitive' categories to form a combination, a total of ten combinations. Then, we calculate the average F1 score for each crawl as follows:

$$Average\ F1 = \frac{1}{n}\sum_{n=1}^{n} F1_i \tag{4.1}$$

(n: the number of combinations, n = 10)

Figure 4.7 shows the classification performance of five crawlings. We notice that LightGBM outperforms other classifiers. From the first to second crawling, we find an increase, of around 0.08. However, the classification performance slightly improves from the second to fifth crawling. Thus, the performances are not directly related to the number of crawls.

4.3 Summary

This chapter first evaluates the binary classification performance of encrypted DNS protocols: DoQ, DoT, and DoH. For the DoQ protocol, we also compare the results with

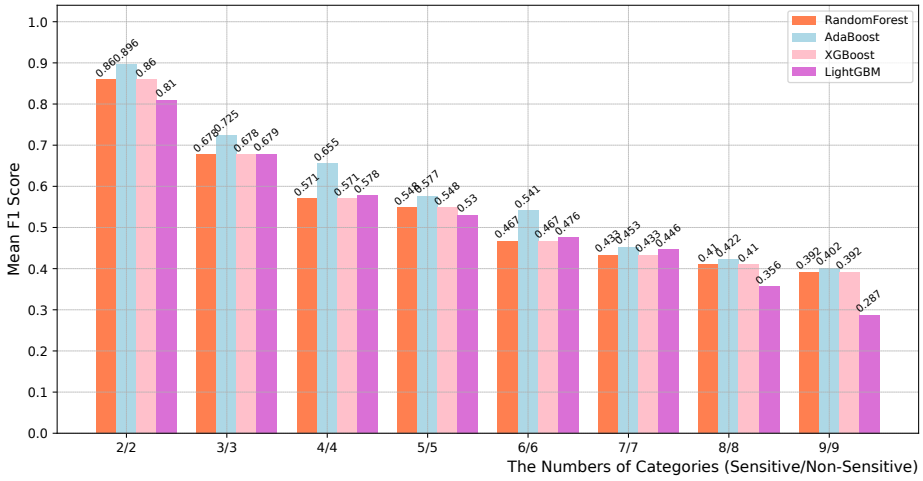


Figure 4.6: Multi-classification performance (Bind)

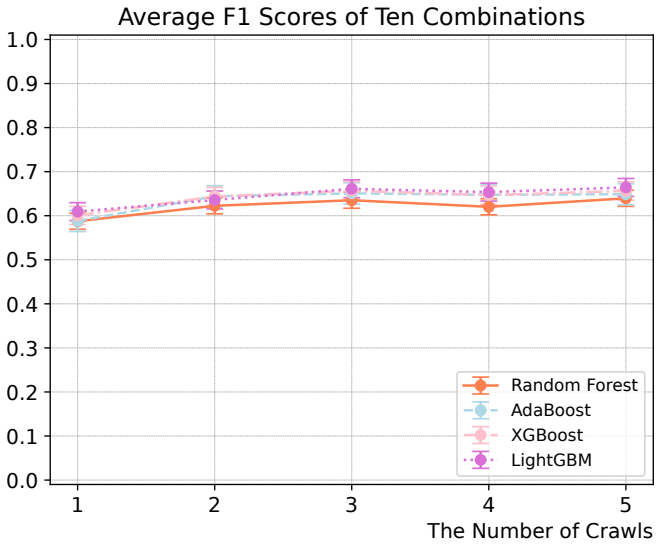


Figure 4.7: Multi-Classification Performance (Effect of the Number of Crawls)

different DNS resolvers: Bind, NextDNS, and Google; and show the top-10 important features. About the Bind resolver, we also consider the effect of caching on Bind: caching order and with/without cache. For Google resolver, we evaluate the binary classification performance of DoQ, DoT, and DoH.

The results show that the binary classification performances of Bind, NextDNS, and Google are very high for the four classifiers, regardless balanced and imbalanced dataset. Also, there is no big difference between these two datasets. We note that the performances of Bind are slightly higher than NextDNS. We investigate the discriminative powers of the feature, we notice that the top-10 important features are mainly related to inter-arrival time, then packet length. While we examine the effect of cache on Bind, we confirm that the caching order and with/without have less effect on mitigating the attack. then, we measure the impact of DoT and DoH, and the F1 measures are still higher than 0.9 regardless of balanced and imbalanced dataset. This result is consistent with DoQ on the Google resolver. Thus, we demonstrate that user privacy leakage is still possible even in all the encrypted DNS regardless of the underlying protocols and the DNS resolvers (Bind, NextDNS, and Google).

Then, we evaluate the multi-classification performance of DoQ on Bind. We randomly select different combinations from 30 categories and compare the performance. We find the performances decrease more as the number of categories increases. Also, we examine the effect of the number of crawls by crawling the 30 categories five times. We demonstrate that the performances are not directly related to the number of crawls.

5 Countermeasure

From the previous baseline results, a promising approach is to control the inter-arrival distribution and packet length for the mitigation. In this chapter, we discuss four possible countermeasures to mitigate the website fingerprinting attack on DoQ as shown in Figure 5.1. In our experiment, we measure the F1 score to evaluate the effectiveness of countermeasures. Compared to experiments without countermeasures, the strategy is in effect once the F1 decreases. We also consider the confusion matrix of classification performance.

We investigate three possible ways to control the inter-arrival time by using the AdBlocker extension in Firefox, disabling DNS prefetch, and adding random delay in the response process of DNS with binary classification in §6.1. To randomize the inter-arrival time, the easiest way is to add different random delays to the query or response. Also, as standardized in RFC 8467 [18], DNS payload padding is useful in controlling the packet length. For padding the DNS payload, we add the payload in the DNS query and response with Bind in §6.4. Then, we evaluate the performance as that combination of two countermeasures: adding random delay and padding DNS payload

with binary and multi-classification in §6.5.

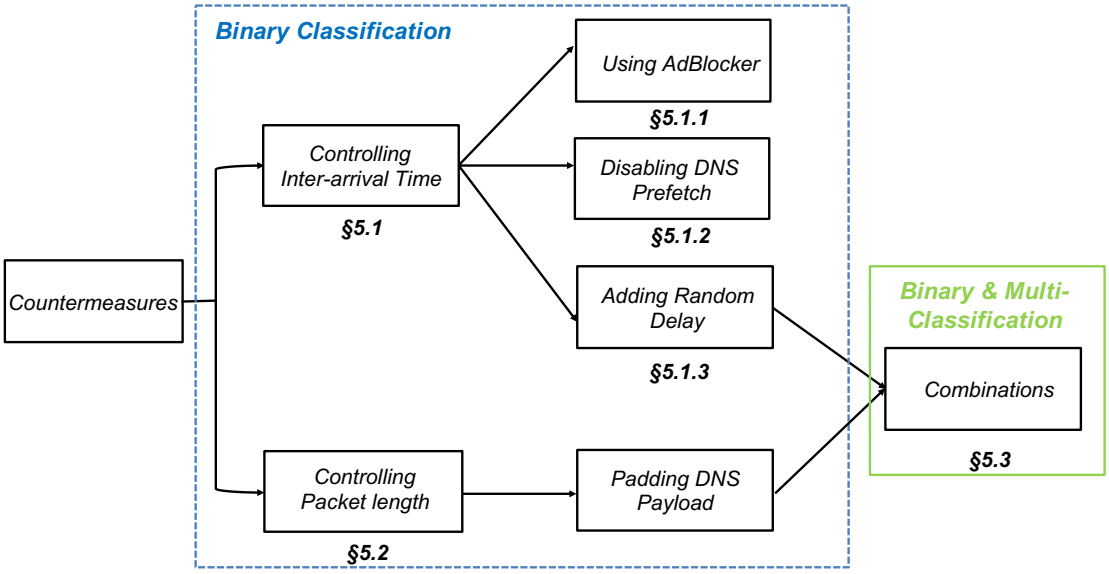


Figure 5.1: Possible Countermeasures to Mitigate the Website Fingerprinting Attack

5.1 Controlling Inter-arrival time

We discuss three possible ways to control inter-arrival time: using Adblocker extension, disabling DNS prefetch, and adding random delay.

5.1.1 Using AdBlocker

Here, we present the first countermeasure that might affect the classification performance, AdBlocker. The AdBlocker extension is often used to detect and block advertisements when users visit the websites. In this process, blocking these ads speeds up webpage loading time and might affect the inter-arrival time of the DNS process. We select the Ublock origin [84] extension (ver. 5.4.2) to install on the Firefox browser.

We measure the binary classification using the imbalanced dataset and compare the results with/without AdBlocker between NextDNS and Bind. As shown in Figure 5.2, we confirm the degradations of classification performance with AdBlocker are 6% on Bind and 5% on NextDNS. However, there is no significant difference between these

two cases, regardless of NextDNS and Bind. Thus, the results confirm that AdBlocker is less influential in mitigating the website fingerprinting attack.

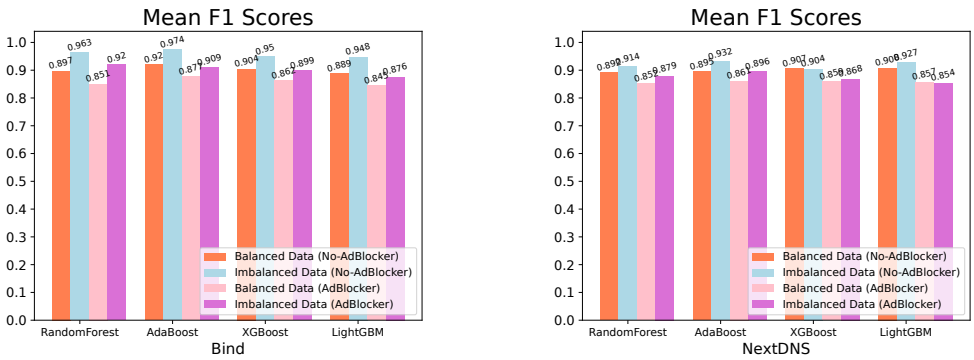


Figure 5.2: Classification performance of using AdBlocker

5.1.2 Disabling DNS Prefetching

Here, we describe the second countermeasure: DNS prefetching, which is the act of solving the domain names before web resources get requested (i.e., before the user clicks its link). Latency issues can be solved by DNS prefetch, so this function is enabled by the default setting in most browsers. This technique reduces the time (the process of looking up the DNS information) when the user accesses the webpage. Thus, this might affect the inter-arrival time, which in turn is classification performance. We conduct the experiment of disabling the DNS prefetch and then compare the classification performance with enabling (default setting) prefetch using the imbalanced dataset.

As shown in Figure 5.3, we find that the classification performance is slightly lower than enabling DNS prefetch for each classifier. The degradation of NextDNS (8%) is slightly higher than Bind (6%) while disabling DNS prefetch. However, there is no significant difference between these two cases, regardless of Bind and NextDNS. Thus, the results indicate disabling DNS prefetch can not help the user protect the private information more.

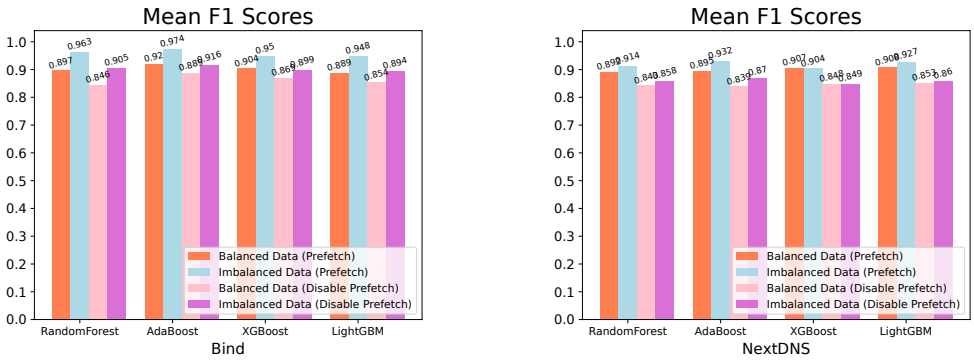


Figure 5.3: Classification performance of disabling DNS prefetch

5.1.3 Adding Random Delay

Here, we conduct an experiment to understand the random delay effect as the third countermeasure. We add a random delay on the returning path at the DNS proxy to the client, to ensure that the random delay is added to the encrypted traffic as shown in Figure 5.4. The random delay is followed by uniformly distributed random values. The range of random delay varies from 0-3 milliseconds to 0-300 milliseconds [1].

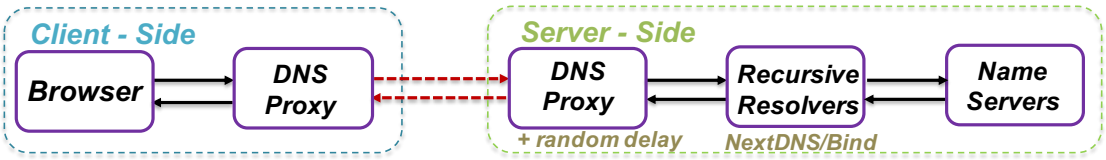


Figure 5.4: Adding random delays

As shown in Figure 5.5, we confirm the classification performance degradation (10% - 22%) by adding random delays. We also notice that the degradation of NextDNS (22%) is slightly higher than that of Bind (18%). This is likely because of the different geographical locations between the client and the resolver. NextDNS is a public DNS resolver that can be somewhere in the network, though Bind is the local resolver in the home network. Thus, in the more realistic situation where DoQ is provided by a public resolver, we expect better mitigation. The results demonstrate that adding random delays could decrease the performance. Also, the higher random delay might affect the

[1] 1 millisecond; 0-3 ms; 0-10 ms; 0-40 ms; 0-60 ms; 0-100 ms; 0-200 ms; 0-300 ms

user experience.

The distribution of inter-arrival

We also show the distribution of inter-arrival time after we add the random delay in the DNS response in Figure 5.6. While we added the 0-100 ms delay, the range of the mean inter-arrival time feature changed, from 0.0-1.0 to 0.0-2.0.

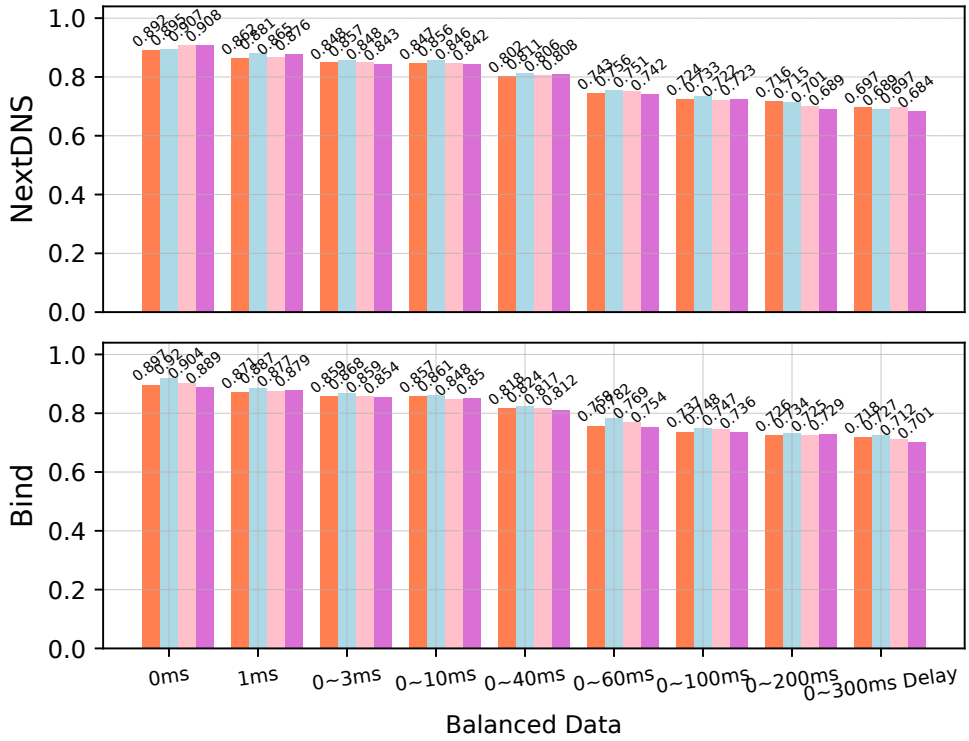
5.2 Controlling Packet Length

Here, we focus on DNS payload padding standardized in RFC 8467 [85]. DNS padding is a technique used to mitigate the attack by adding additional data to make them uniform in size. This technique is often employed as a privacy-enhancing measure to prevent attackers from inferring information about the content of DNS traffic based on packet sizes. This makes it more difficult for attackers to analyze the DNS traffic. Add EDNS(0) padding option to outgoing messages to increase the packet size. DNS padding has not been supported by all the DNS software. On the client side, we use the Adguard dnsproxy to support the DoQ. However, it does not have the ENDS(0) [86] option, we modify the configuration of routedns [87] to pad the DNS query. Also, we notice that a newer version (ver. 9.19) of Bind supports the EDNS(0) padding option, though NextDNS does not. We modify the configuration of Bind to support pad response, see more details in Appendix A.3.

Thus, we take two evaluations with Bind as shown in Figure 5.7: 1) Only padding at the outgoing direction from the client to the resolver. We set the block size is 128 bytes, after padding, the padded query length becomes a multiple of 128 bytes, such as 128, 256, and 512 bytes. and 2) While the Bind receives the query with the EDNS (0) option, the response will be padded to a multiple of 468 bytes. The former corresponds to the case that the resolver does not support the padding, and the latter is the case that both the client and the resolver are aware of the padding.

5.2.1 Padding DNS Payload

As shown in Figure 5.8, we confirm that the padding affects the classification performance. We obtain the lowest results for padding the DNS message on the client



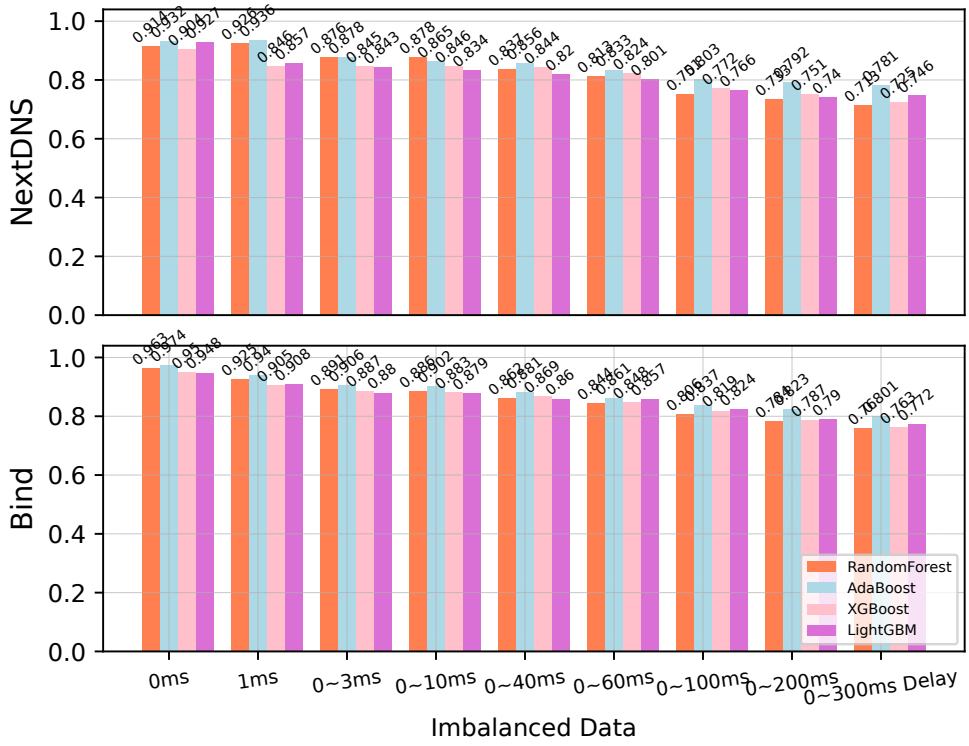
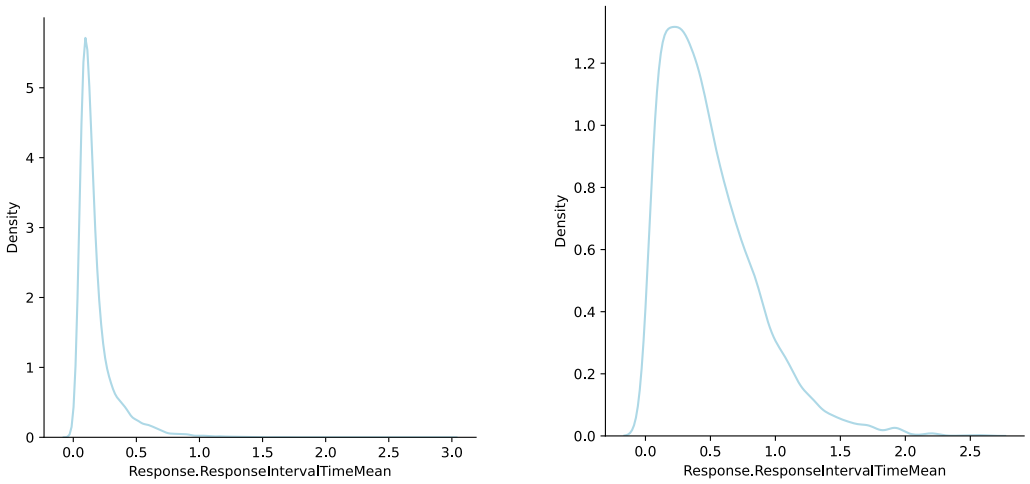


Figure 5.5: Classification performance of adding random delays (Bind and NextDNS)



(a) No delay (b) Add 0-100ms delay

Figure 5.6: The distribution of Adding random delays (i.e., the mean value of response inter-arrival time)

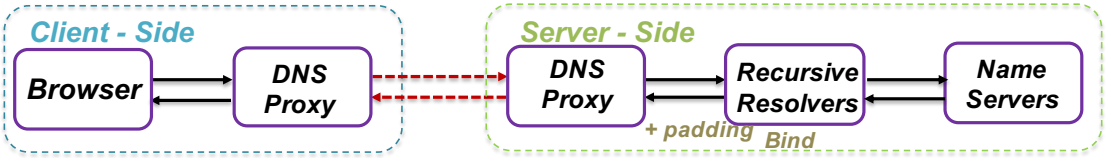


Figure 5.7: Padding DNS payload

and server; the performances decrease from 0.97 to 0.86 with padding query and response. However, the performance degradation is limited to the case of outgoing padding only. Thus, padding the DNS message for both directions is effective, though the wide deployment of padding is required for the caching resolvers.

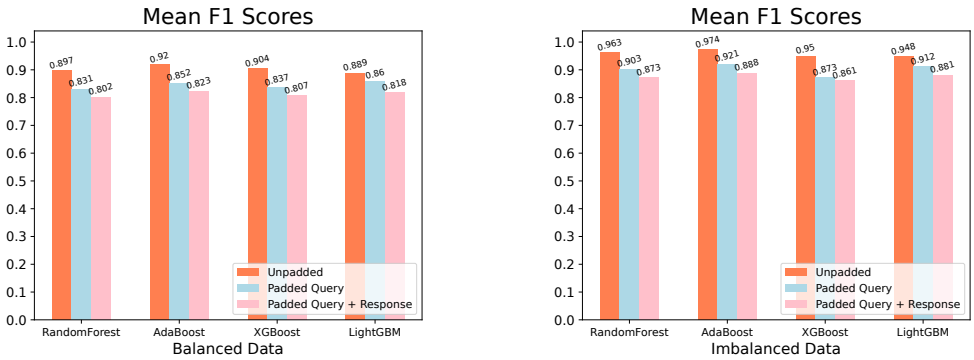


Figure 5.8: Classification Performance of Padding DNS payload (Bind)

5.3 Combination of Two Countermeasures

From the results of adding random delay and payload padding, we find they have some positive effects on performance degradation. Here, we add the random delay (0-60ms and 0-100ms) and pad the query/response on Bind. Then, we show the performance of binary and multi-classification. For binary classification, we also show the confusion matrix.

5.3.1 Binary Classification Performance

As shown in Figure 5.9, we find a significant change with padding and adding 100ms delay. The average reduction is about 27% with adding 100ms delays and padding query/response. The results conclude that adding some random delays and padding at the same time can protect users' information from attacks to a certain extent though the delay affects the user experience.

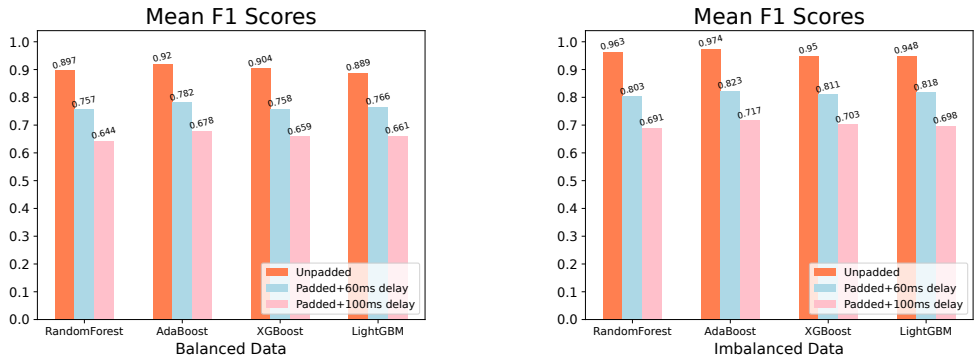


Figure 5.9: Classification performance for random delays and padding (Bind)

5.3.1.1 Feature Importance

We also show the top-10 discriminative features in Table 5.1. We notice that the effective features change, not only related to inter-arrival time and packet length. Other features such as 'QueryPktCount' and 'RPerSecondMax' related to packet count also appear in the list. We also observe that the mean value of the top-10 important features decreases compared to the Bind (as shown in Table 4.1 and Table 4.2) without the countermeasures. Thus, our approach to mitigating the inference ability works as expected.

Table 5.1: Top-10 discriminative features (random delays and padding)

Rank	Balanced Data		Imbalanced Data	
	Feature	**Mean**	**Feature**	**Mean**
1	QueryIntervalTimeMax	0.015	QueryLengthDeciles8	0.015
2	QueryLengthDeciles7	0.013	QueryIntervalTimeMax	0.015
3	FlowSentRate	0.012	ResponseLengthMin	0.013
4	QRIntervalTimeStandard	0.011	ResponseLengthDeciles3	0.012
5	QueryLengthMode	0.011	QueryPktCount	0.010
6	QRIntervalTimeCoefficientVariation	0.010	ResponseLengthDeciles5	0.009
7	QueryIntervalTimeStandard	0.009	Duration	0.009
8	QueryPktCount	0.009	RPerSecondMax	0.009
9	QRIntervalTimeMax	0.009	QueryLengthMax	0.009
10	ResponseIntervalTimeDeciles5	0.008	DFIntervalTimeDeciles2	0.008

5.3.1.2 Confusion Matrix

Finally, we show the confusion matrix [88] of Bind without countermeasures in Figure 5.10 (left) and adding 0-100ms delay and padding query/ response DNS payload in Figure 5.10 (right) for the imbalanced dataset. We find the value of the false positive increases from 61 to 449, and the false negative also increases from 72 to 318. Both the values of true positive and negative decrease. The results support our conclusion that adding random delays and padding could mitigate the website fingerprinting attack again.

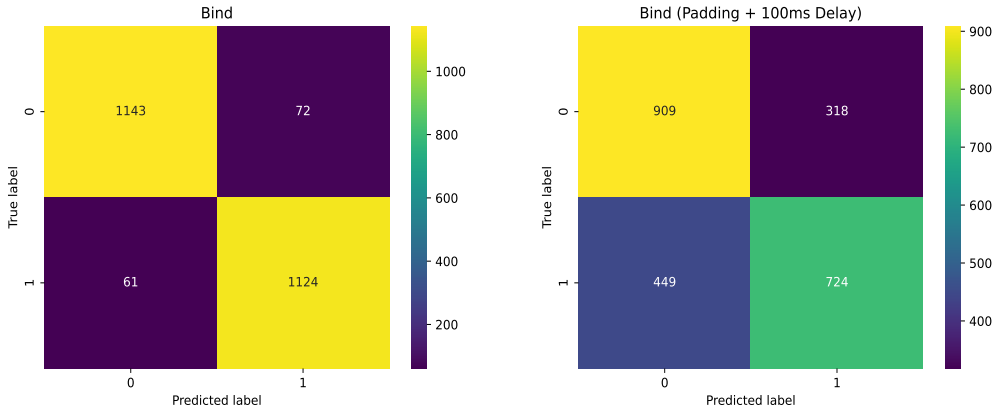


Figure 5.10: Confusion matrix of Bind with/without delay and padding

5.3.2 Multi-Classification Performance

Here, we evaluate the multi-classification performance of these two countermeasures. Figure 5.11 shows a reduction compared to the multi-classification performance of bind without any countermeasures (Figure 4.6). With 0-60ms delay and payload, we find the average reduction is about 12% of 2 - 2 and 5% - 12% of other combinations. Also, the degradation of 0 - 100 ms delay is about 22%. However, there is no significant change from 4 - 4 combinations. Thus, the results demonstrate that adding random delays and padding payload could decrease the multi-classification performance.

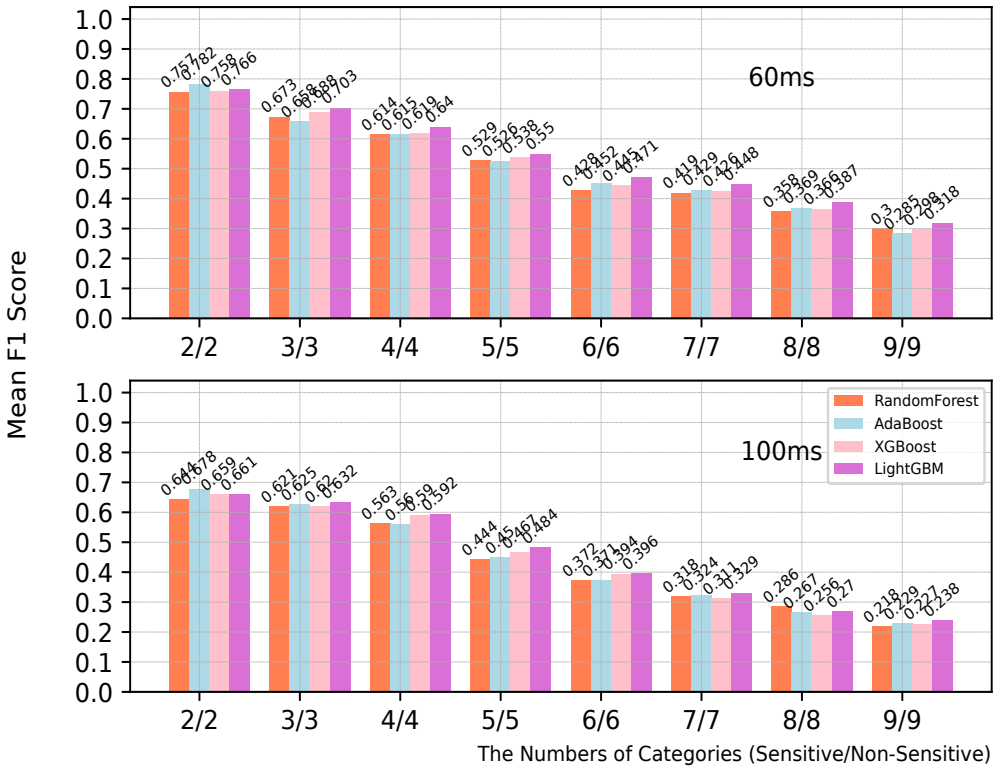


Figure 5.11: Multi-classification performance for adding random delay and padding payload (Bind)

5.4 Summary

In this chapter, we propose a promising approach to mitigate the website fingerprinting attack of DoQ by controlling the inter-arrival time and packet length.

We first describe three possible ways to control time: using AdBlocker, disabling DNS prefetch, and adding random delays in DNS response. We find that AdBlocker and disabling DNS prefetch are less influential in mitigating the attack. The degradations of classification performance with AdBlocker are 6% on Bind and 5% on NextDNS. While disabling DNS prefetch, the degradations of the performance of Bind is 6% and NextDNS is 8%. The third countermeasure is to add the random delay in the returning path. We confirm that adding random delay could mitigate the attack, regardless of the

balanced and imbalanced dataset. We also notice that the degradation of NextDNS (22%) is slightly higher than that of Bind (18%).

Then, we pad the DNS payload to control the packet length. We take two evaluations with Bind: 1) Only padding at the outgoing direction (DNS query), and 2) Padding at both outgoing and returning direction (DNS query and response). The classification performances decrease from 0.97 to 0.86 with padding query and response. The results demonstrate that padding the DNS message for both directions is effective.

Finally, based on previous results, we confirm adding random delay and padding the DNS payload have some positive effects on mitigating the attack. We both add random delay (0-60 ms and 0-100 ms) and padding DNS payload on query/response. The average reduction is about 27% with these two countermeasures. The results conclude that adding some random delays and padding at the same time can help users protect private information from attacks. We also notice that the top-10 features are not only related to inter-arrival time and packet length. Other features such as 'QueryPktCount' and 'RPerSecondMax' related to packet count also appear in the list. Thus, the results demonstrate that our approach: adding random delays and padding payload has a better mitigation.

6 Discussion

This dissertation analyzed the privacy leakage problem of three encrypted DNS: DoQ, DoT, and DoH with public and local DNS resolvers. We intended to understand whether the encryption could help users protect private information and how to defend against the website fingerprinting attack. Also, we discuss the limitations and future works.

6.1 Discussion

6.1.1 Key findings

- **Finding 1: Information leakage is still possible even in encrypted DNS regardless of the underlying protocol and DNS resolvers.**

  Encrypted DNS can enhance privacy and security at the application layer and encrypt DNS traffic to prevent eavesdropping and tampering. Firstly, we examined the classification performance of DoQ with three DNS resolvers using

two configurations (the DNS resolver supports the encrypted DNS or not). From the basis analysis, we observed that the F1 scores are higher, around 0.9. Secondly, we compare the results of three encrypted DNS protocols: DoQ, DoT, and DoH with Google resolver. Three DNS encryption methods encapsulate DNS in their encryption-enabled layer, i.e., QUIC, TLS, and TLS with HTTP (HTTPS). However, we observed these differences do not matter for our results. Depending on the baseline analysis, we demonstrate that user privacy leakage is still possible even in these three encrypted DNS regardless of the underlying protocols and DNS resolvers.

Recommendation for researcher: Develop evaluation frameworks and metrics for quantifying privacy leakage in encrypted DNS protocols. This includes defining metrics to measure the encrypted DNS traffic, the effectiveness of privacy-enhancing features, and the impact of different factors (i.e., attack scenarios) on user privacy.

Recommendation for users: Encrypted DNS protocols, such as DoH, DoT, and DoQ, are effective in helping users prevent third-party eavesdropping. Encrypted DNS ensures that the content of DNS queries and responses is hidden from network intermediaries, including ISPs and potentially malicious actors. The easiest way for users to deploy the encrypted DNS is by modifying a configuration file: select the DNS provider like 1.1.1.1 to support encrypted DNS in the setting of browsers. Browsers like Chrome and Firefox have already supported DoH.

Recommendation for developer: Implement privacy-preserving features in encrypted DNS protocols to enhance user privacy. This includes options for users to control data-sharing preferences. Also, developers should implement measures to minimize metadata exposure, such as randomization and padding.

- **Finding 2: The inter-arrival time and length-related features are key metrics for mitigation**

Controlling inter-arrival time and packet length is a promising approach to mitigate the website fingerprinting attack on DoQ. We conducted the experiments of DoQ with Bind and NextDNS resolver.

- For controlling inter-arrival time, we proposed three approaches: using AdBlocker, disabling DNS prefetch, and adding random delay. The easiest way is to add the random delay (0-300ms) in the returning path. For adding random delay, the degradation of NextDNS (22%) is slightly higher than that of Bind (18%). We confirm that adding random delay could reduce the F1 score regardless of Bind and NextDNS. However, more delays may affect users' experience. For example, the webpage may partially load, displaying some content but not all. Users may see a blank or incomplete page, with missing images, text, or other elements. Even if some content loads, interactive elements such as buttons, links, or forms may be slow to respond or unresponsive altogether.

  Also, we tested another two experiments: install the AdBlocker extension and disable DNS prefetch on the Firefox browser. However, the performance degradation is limited to the case of using AdBlocker and disabling DNS prefetch. We obtain the lowest results for padding the DNS message on the client and server, the F1 value decreases from 0.97 to 0.86.

Recommendation for researcher: While adding more delays, the website loading process is slow. The researchers may not capture all packets related to the webpage load. For example, our experiment gives each page 30 seconds to load. After adding random delays, the webpage may not fully load within this specified time. This can result in incomplete or missing data, making it challenging to analyze the entire communication process accurately.

Recommendation for users: Adding intentional delays to webpage loading can be a complex decision, as it directly impacts user experience. For users, we recommend selecting the DNS proxy that supports encrypted

DNS and adding random delays. Also, a better way for users is to install the AdBlock extension in the browser.

Recommendation for developers: Minimize the impact of delays on user experience by optimizing their duration and timing. The developer should ensure that delays are only added when necessary and that they do not significantly degrade the overall user experience. For the developer of the encrypted DNS proxy, we recommend giving the option message of adding different delays to the user.

How long are web users willing to wait? Ref. [89] suggested that the waiting time for the user to load the webpage is approximately 2 seconds. Also, as the report of some analysis tools [90], they select the top 1,000,000 websites and test the loading speed from Jan 1, 2020, to Apr 1, 2024. They show the median loading time is 2.1 seconds for the desktop. Studies have shown that users prefer web pages to load quickly, ideally within 2 to 3 seconds. Longer loading times can lead to increased bounce rates, where users abandon the page before it finishes loading. We suggest the range of adding random delay is 0-60 or 0-100 ms for response. The cost of one response is 0.05 - 0.1 second higher than the case of adding no delay. If one homepage triggers 40 response packets, the additional delay could be 2 - 4 seconds. We also observed the F1 score could decrease the performance by about 15% - 18%.

– For controlling packet length, we pad the DNS payload on the Bind. RFC 8467 [85] proposed the DNS padding strategies for encrypted DNS. Some studies from [15, 70] confirmed the mitigation is limited to padding on DoT and DoH. We took two evaluations on Bind: 1) only padding query, and 2) padding query and response. We found the performance degradation is about 11% with padding query and response. However, the degradation is limited to padding query, only 6%.

Recommendation researcher: The supported DoQ caching resolver does not support padding in EDNS(0) queries as NextDNS. There is no

way to check this case. We can only select Bind9 because the padding is supported by modifying the configuration file. We also need a proxy to support the EDNS(0) query.

Recommendation for users: Some local resolver software supports the encrypted DNS and padding, like Bind and Unbound. The user could configure the padding settings that involve specifying the padding length, padding algorithm, or enabling automatic padding according to their preferences.

Recommendation for developers: Incorporate padding support into your DNS client applications or libraries. Ensure that the developer's implementation adds appropriate padding to DNS queries before sending them to caching resolvers. Also, offer users configuration options within the application to enable or disable padding. Provide clear explanations of the privacy and security benefits of enabling padding.

Recommendation for network operators: Deploy DNS caching resolvers that support encrypted DNS with padding within the network infrastructure. Configure the DNS caching resolvers to add padding to DNS queries and handle padded queries from clients. While DNS query padding is typically applied at the client or resolver side, supporting padding at authoritative DNS servers is less common and has some nuances to consider. Adding padding at authoritative servers might not provide significant privacy benefits, as the primary goal of padding is to obscure DNS query patterns between clients and resolvers. However, supporting padded queries ensures compatibility with privacy-enhancing configurations implemented at the client or resolver side.

These results demonstrated that adding random delay and padding DNS payload are effective privacy strategies for DNS encryption.

6.1.2 The target F1 scores for binary and multi-classification

We consider the F1 score as the evaluation results of our experiment. Without any countermeasures, the mean F1 scores are all around 0.9 regardless of the datasets and DNS caching resolvers. We could not find any references about the universal target F1 score that applies to all machine learning models. Some blogs pointed out that the results of better models should be higher than 0.85. We intend to get more degradation of the F1 score for the countermeasures. For example, while we add the random delays in the DNS response, the degradation of NextDNS is 22% and Bind is 18% for binary classification.

Additionally, the binary and multi-classification should have different f1 score targets. To determine what target F1 value for users is more appropriate to consider the actual risks in the binary and multi-classification scenarios, we need to consider several criteria that are crucial for assessing risk, particularly when a higher F1 score implies a higher security risk for user:

In binary classification scenarios, the attacker's goal is typically to distinguish between two classes (e.g., *'Sensitive'* or *'Non-Sensitive'*). If the F1 score is high (typically above 0.8), it indicates effective discrimination between sensitive and non-sensitive categories. For the attacker, it is easy to get a higher F1 score than that of the multi-classification. For example, we got the results around 0.9 without countermeasure. In the binary classification, the attacker just knows whether it belongs to the sensitive or non-sensitive category, not the details of classes. Thus, the actual risk of binary classification is relatively low for user because it is binary. In our experiment, the F1 score decreases from 0.9 to 0.68 with two countermeasures (adding random delay and padding payload).

In the multi-classification scenario, the goal is to classify the website into multiple classes (e.g., *'Dating'*, *'Phishing'*, or *'Entertainment'*). This task is more complex, requiring distinguishing between more than two classes compared to binary classification. For the attacker, it is hard to get a higher F1 score, because each category may have a different F1 score target based on its sensitivity and importance. For example, the F1 score varied from 0.8 (2/2: four categories) to 0.4 (9/9: 18 categories) without any countermeasures in our experiment. More categories make it harder to get higher F1 scores. If the F1 score of multi-classification is relatively high, the attacker could

infer more details of the user's interest or habit. This is much more risky than binary classification for users. After we added random delay (0 - 100ms) and padding DNS payload, the results are 0.6 with 2/2 combinations and only 0.2 with 9/9 combinations.

Thus, the target F1 score for multi-classification should be lower than that for the binary to reduce the user's risks, and our measurement results support this consideration; Indeed, the risk of the multi-classification is lower for more categories in our controlled experiments.

6.1.3 Why category should be protected?

As discussed in a past literature [15], inferring categories from encrypted DNS data is a risk for users. For example, websites in the health category may indicate that users intend to access to obtain medical information. Such personal sensitive categories (e.g., health, dating, gambling, religion) should be protected from attackers.

6.1.4 Can the attacker evade the countermeasures?

For our countermeasures, we presented two effective countermeasures: adding random delay and padding DNS payload. If the attacker extracts features like packet size and timing and then uses machine learning techniques to analyze the encrypted DNS traffic, it' s hard to evade the countermeasure for the attacker. However, if the attacker can control the client or the DNS caching resolver directly, they might be able to decrypt the padded data.

6.2 Limitations

Here, we discuss the limitations of our experiments.

The dependency on site. The first limitation of our work is that we only consider the top-400 websites of each category in the Alexa dataset. We chose these websites due to their great influential power and often visited by users.

Physical location. The second limitation is that the physical measurement point of the experiment is close to the user. We conducted these experiments in the home

network, only one location (Tokyo, Japan). This location is close to the user in our threat model. For the general scenario, the attackers are located in the middle or near caching resolvers. we will consider more platforms and sites to measure classification performance in future work.

The cases of domains that provide many services. Another limitation of our work is that we do not consider the influence of CDN (Content Delivery Network). CDN is a geographically dispersed group of servers that speeds up the distribution of Web content by placing content (i.e., image, video, and social media) closer to the user's location. While the user visits the websites, the server may request additional contexts for CDNs that are near the user. In this process, The CDN may reduce the page loading time, then affect the inter-arrival time.

Padding limitation. Our padding experiments are only conducted with the bind resolver, because other public DNS resolvers (i.e., NextDNS, Google) do not support the padding, at the time of writing. Padding DNS payload can introduce additional costs in terms of network overhead. By padding DNS payload, the size of DNS messages increases. This can lead to larger packets being transmitted over the network, potentially affect the inter-arrival time, especially in environments where network resources are limited or congested.

Measurement and testing environment Finally, we do not consider the measurement and testing conditions, such as the CPU power of hosts and network bandwidth. 1) The CPU can affect the DNS server's transmission speed to handle incoming requests and generate responses. If the CPU is overloaded or the processing capability is insufficient, the processing time of DNS queries may be longer, and the arrival interval of DNS packets increases; 2) Network bandwidth refers to the capacity of the network connection to transmit data. Bandwidth affects the transmission delay of packets, which in turn impacts the inter-arrival time. Higher bandwidth allows for faster transmission of DNS packets between clients and DNS servers, leading to shorter inter-arrival times if the network is not congested.

6.3 Future Work

In the following, we identify possible directions for future work:

1. Consider adding an appropriate delay or only add the delays in some response packets: Adding random delays could reduce the classification performance. However, more delay may affect the user experience while loading the webpage. For example, if one homepage triggers 40 response packets, the additional delay is 2 - 4 seconds. We consider two ways to add random delays: 1) We will build a script to find the "perfect" delay against the attack; 2) We can add the delay in the random response, not all response packets. For example, we add the delay in some responses with a response probability.

2. Building a tool to pad DNS queries or responses in the DNS resolver: The DoQ cannot support the EDNS(0) in the client. We plan to build an automatic tool to encrypt query and pad DNS payload as a proxy. While the resolver receives the DNS message, the proxy pads the response with different octets. Then, send the encrypted and padded response to the client. We also intend to figure out the "perfect" (which size is the most effective for the F1 score) padding size.

3. Consider different network conditions: In our study, we only experimented with the home network. We will consider more network conditions like 4G, 5G, campus networks, mobile, and different kinds of ISP networks. The network may affect the performance.

4. Compare the classification performance of sensitive category: In this thesis, we compare the performance of *"Sensitive"* and *"Non-Sensitive"*. We want to examine the performances of different *"Sensitive"* categories (i.e., *"Dating"* and *"Health"*) and find the key metrics for each category.

7 Conclusion

Privacy leakage on the web is a serious problem with the current Internet. Various encrypted DNS protocols have been standardized to protect users from website attacks, such as the website fingerprinting attack. In this work, we intended to understand whether the encrypted DNS protocols (DoQ, DoT, and DoH) could protect user privacy while visiting the webpage.

At first, we investigated the information leakage by performing the binary classification analysis of DoQ traffic on Bind, NextDNS, and Google resolvers. We confirmed that the classification performances of the websites are very high in these resolvers to infer which category of websites the user visited. Specifically, we pointed out that discriminative features are mainly related to the inter-arrival time of packets and the packet length. Then, we measure the impact of DoT and DoH with Google resolver. We noticed that the results are still high to be consistent with DoQ.

Depending on the baseline analysis, a promising way to mitigate the attack is by controlling inter-arrival time and packet length. We proposed four countermeasures: using the AdBlocker extension, disabling DNS prefetch, adding the random delay

to control inter-arrival time, and padding DNS payload to control packet length. Using AdBlocker and disabling DNS prefetch had less effect in mitigating the website fingerprinting attack. Adding the random delay decreased the classification performance by 18% with Bind and 22% with NextDNS. While we padded the DNS payload (query/response) with Bind, the performances decreased by 10% (from 0.97 to 0.86). In the end, we confirmed reasonable degradation: approx 27% of binary class and 22% of multi-class, while padding and adding a 100ms delay.

Our analysis clarified four possible countermeasures: using Adblocker and disabling DNS prefetch have less effect in mitigating the attack, adding random delay and padding DNS payload are promising to mitigate privacy harm from users, having a 0.27 reduction in the F1 score with Bind. However, adding a delay is a tradeoff between the privacy and user experience. Also, DNS padding has not yet been widely deployed in public DNS. For the more realistic situation (i.e., NextDNS), we expect more degradation of the classification performance.

Publications

List of the publications related to this PhD dissertation.

Journal Papers

1. **GUANNAN HU**, and Kensuke Fukuda, "Investigate the Countermeasures to Mitigate the Privacy Leakage in DNS over QUIC", Journal of Information Processing, IPSJ, 2024. (submitted)

2. **GUANNAN HU**, and Kensuke Fukuda, "Characterizing Privacy Leakage in Encrypted DNS traffic", IEICE transactions on Commmunications, Vol.E106-B, No.2, pp.156-165, IEICE, 2023.

International conference papers (reviewed)

1. **GUANNAN HU**, and Kensuke Fukuda, "An analysis of privacy leakage in DoQ traffic", CoNEXT Student Workshop 2021 (CoNEXT-SW'21), pp.7-8, Munich, Germany (Virtual Conference), 2021. ACM.

2. **GUANNAN HU**, and Kensuke Fukuda, "Privacy Leakage of DNS over QUIC: Analysis and Countermeasure", International Conference on Artificial Intelligence in Information and Communication (ICAIIC'24), pp. 518-213, Osaka, Japan, Feb 2024. IEEE.

Bibliography

[1] Paul Mockaptris. Domain names - implementation and specification. https://www.rfc-editor.org/info/rfc1035, 2004.

[2] E. Rescorla. Http over tls. https://www.rfc-editor.org/info/rfc2818.

[3] Jean-Pierre Smith, Prateek Mittal, and Adrian Perrig. Website Fingerprinting in the age of QUIC. *Computer Networks*, 200:48 – 69, 2021.

[4] Pengwei Zhan, Liming Wang, and Yi Tang. Website fingerprinting on early quic traffic. *Computer Networks*, 200:108538, December 2021.

[5] S. Farrell. Pervasive Monitoring Is an Attack. https://www.rfc-editor.org/rfc/rfc7258, 2014.

[6] P. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). https://www.rfc-editor.org/info/rfc8484, 2018.

[7] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for DNS over Transport Layer Security (TLS). https://www.rfc-editor.org/info/rfc7858, 2016.

[8] C. Huitema. Specification of DNS over Dedicated QUIC Connections. https://datatracker.ietf.org/doc/rfc9250/, 2022.

[9] Dnscrypt. http://dnscrypt.org/, 2019.

[10] Google. https://developers.google.com/speed/public-dns/.

[11] Cloudflare dns. https://cloudflare-dns.com/.

[12] Quad9. https://www.quad9.net/.

[13] Adguard Software Ltd.˙ Adguard. https://adguard.com.

[14] Romain Cointepas and Olivier Poitrey. Nextdns. https://nextdns.io/.

[15] Rebekah Houser, Zhou Li, Chase Cotton, and Haining Wang. An Investigation on Information leakage of DNS over TLS. In *ACM CoNEXT'19*, pages 123 – 137, Orlando, Florida, USA, 2019.

[16] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. Encrypted DNS ⇒ Privacy? A Traffic Analysis Perspective. In *NDSS'20*, San Diego, CA, USA, 2020.

[17] Sandra Deepthy Siby, Marc Juárez, Narseo Vallina-Rodriguez, and Carmela Troncoso. DNS Privacy not so private: the traffic analysis perspective. In *HotPETs'18*, 2018.

[18] A. Mayrhofer. Padding Policies for Extension Mechanisms for DNS (EDNS(0). https://datatracker.ietf.org/doc/rfc8467, 2022.

[19] Jana Iyengar and Martin Thomson. Quic: A udp-based multiplexed and secure transport. https://www.rfc-editor.org/info/rfc9000, May 2021.

[20] Chris Duckett. Google public dns gets dns-over-tls treatment. https://www.zdnet.com/article/google-public-dns-gets-dns-over-tls-treatment/, 2019.

[21] Public resolvers. https://dnsprivacy.org/public_resolvers.

[22] Publicly available servers. https://github.com/curl/wiki/DNS-over-HTTPS.

[23] ISC. Bind 9. https://www.isc.org/bind.

[24] NLNET Labs. Unbound. https://www.nlnetlabs.nl/projects/unbound/about/.

[25] About stubby. https://github.com/getdnsapi/stubby, 2018.

[26] Knot. https://www.knot-dns.cz/, 2018.

[27] Firefox browser. https://www:mozilla:org/en-US/exp/firefox/.

[28] Chrome browser. https://www:google:com/chrome/.

[29] Opera browser. https://brave:com/.

[30] Safari browser. https://www.apple.com/safari/.

[31] Edge browser. https://www.microsoft.com/en-us/edge/.

[32] Brave browser. https://brave:com/.

[33] Andrey Meshkov. Adguard dnsproxy. https://github.com/AdguardTeam/dnsproxy, 2021.

[34] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website Fingerprinting at Internet Scale. In *NDSS'16*, San Diego, CA, USA, 2016.

[35] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *ACM CCS'12*, pages 605 – 616, Raleigh, NC, USA, 2012.

[36] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *ACM CCS'14*, pages 227 – 238, Scottsdale, AZ, USA, 2014.

[37] Jamie Hayes and George Danezis. K-fingerprinting: a Robust Scalable Website Fingerprinting Technique. In *USENIX Security'16*, pages 1187 – 1203, Austin, TX, 2016.

[38] Shawn Shan, Arjun Nitin Bhagoji, Haitao Zheng, and Ben Y. Zhao. A real-time defense against website fingerprinting attacks. *ArXiv*, abs/2102.04291, 2021.

[39] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. CCS '06, page 255–263, New York, NY, USA, 2006. Association for Computing Machinery.

[40] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of*

the 10th Annual ACM Workshop on Privacy in the Electronic Society, WPES '11, page 103–114, New York, NY, USA, 2011. Association for Computing Machinery.

[41] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE Symposium on Security and Privacy*, pages 332–346, 2012.

[42] Tao Wang and Ian Goldberg. Improved website fingerprinting on tor. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, page 201–212, New York, NY, USA, 2013. Association for Computing Machinery.

[43] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, San Diego, CA, Aug 2014. USENIX Association.

[44] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website Fingerprinting: attacking popular privacy enhancing technologies with multinomial naïve-bayes classifier. In *ACM Workshop on Cloud Computing Security (CCSW'09)*, pages 31 – 42, Chicago, Illinois, USA, 2009.

[45] Charles V. Wright, Fabian Monrose, and Gerald M. Masson. On Inferring Application Protocol Behaviors in Encrypted Network Traffic. *Journal of Machine Learning Research*, 7:2745 – 2769, 2006.

[46] Laurent Bernaille and Renata Teixeira. Early Recognition of Encrypted Applications. In *PAM'07*, pages 165 – 175, Louvain-la-neuve, Belgium, 2007.

[47] Guang-Lu Sun, Yibo Xue, Yingfei Dong, Dongsheng Wang, and Chenglong Li. An Novel Hybrid Method for Effectively Classifying Encrypted Traffic. In *IEEE GLOBECOM'10*, pages 1 – 5, Miami, FL, USA, 2010.

[48] Gianluca Maiolini, Andrea Baiocchi, Alfonso Iacovazzi, and Antonello Rizzi. Real Time Identification of SSH Encrypted Application Flows by Using Cluster Analysis Techniques. In *NETWORKING'09*, pages 182 – 194, Berlin, Heidelberg, 2009.

[49] Mohamad Jaber, Roberto G. Cascella, and Chadi Barakat. Can we trust the inter-packet time for traffic classification? In *IEEE ICC'11*, pages 1 – 5, Kyoto, Japan, 2011.

[50] Chaoyi Lu, Baojun Liu, Zhou Li, Shuang Hao, Haixin Duan, Mingming Zhang, Chunying Leng, Ying Liu, Zaifeng Zhang, and Jianping Wu. An End-to-End, Large-Scale Measurement of DNS-over-Encryption. In *ACM IMC'19*, pages 22 – 35, Amsterdam, Netherlands, 2019.

[51] Sebastián García, Karel Hynek, Dmtrii Vekshin, Tomáš Čejka, and Armin Wasicek. Large Scale Measurement on the Adoption of Encrypted DNS. In *arXiv 2107.04436*, 2021.

[52] Casey Deccio and Jacob Davis. DNS Privacy in Practice and Preparation. In *ACM CoNEXT'19*, pages 138 – 143, Orlando, FL, USA, 2019.

[53] Minzhao Lyu, Hassan Habibi Gharakheili, and Vijay Sivaraman. A survey on dns encryption: Current development, malware misuse, and inference techniques. *ACM Computing Surveys*, 55(8):1–28, December 2022.

[54] Dnscrypt. http://dnscrypt.org/, 2019.

[55] Dmitrii Vekshin, Karel Hynek, and Tomas Cejka. DoH Insight: Detecting DNS over HTTPS by Machine Learning. In *ARES'20*, pages 1 – 8, New York, USA, 2020.

[56] Sebastián García, Joaquín Bogado, Karel Hynek, Dmitrii Vekshin, Tomáš Čejka, and Armin Wasicek. Large scale analysis of doh deployment on the internet. page 145–165, Berlin, Heidelberg, 2022. Springer-Verlag.

[57] L. Csikor, H. Singh, M. Kang, and D. Divakaran. Privacy of dns-over-https: Requiem for a dream? In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 252–271, Los Alamitos, CA, USA, sep 2021. IEEE Computer Society.

[58] Pratyush Dikshit, Jayasree Sengupta, and Vaibhav Bajpai. Recent trends on privacy-preserving technologies under standardization at the ietf. *SIGCOMM Comput. Commun. Rev.*, 53(2):22–30, jul 2023.

[59] Haya Shulman. Pretty bad privacy: Pitfalls of dns encryption. WPES '14, page 191–200, New York, NY, USA, 2014. Association for Computing Machinery.

[60] Dmitrii Vekshin, Karel Hynek, and Tomas Cejka. Doh insight: detecting dns over https by machine learning. page 9, 08 2020.

[61] Austin Hounsel, Kevin Borgolte, Paul Schmitt, Jordan Holland, and Nick Feamster. Analyzing the costs (and benefits) of dns, dot, and doh for the modern web. In *Proceedings of the Applied Networking Research Workshop*, ANRW '19, page 20–22, New York, NY, USA, 2019. Association for Computing Machinery.

[62] Patrick McManus. Firefox nightly secure dns experimental results. 2018.

[63] Nguyen Phong Hoang, Michalis Polychronakis, and Phillipa Gill. Measuring the accessibility of domain name encryption and its impact on internet filtering, 2022. arXiv:2202.00663.

[64] Nguyen Phong Hoang, Arian Akhavan Niaki, Phillipa Gill, and Michalis Polychronakis. Domain name encryption is not enough: Privacy leakage via ip-based website fingerprinting, 2021. arXiv:2102.08332.

[65] Martino Trevisan, Francesca Soro, Marco Mellia, Idilio Drago, and Ricardo Morla. Attacking doh and ech: Does server name encryption protect users' privacy? *ACM Trans. Internet Technol.*, 23(1), Feb 2023.

[66] Sean Rivera, Vijay K. Gurbani, Sofiane Lagraa, Antonio Ken Iannillo, and Radu State. Leveraging ebpf to preserve user privacy for dns, dot, and doh queries. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA, 2020. Association for Computing Machinery.

[67] Lin Jin, Shuai Hao, Haining Wang, and Chase Cotton. Understanding the impact of encrypted dns on internet censorship. In *Proceedings of the Web Conference 2021*, WWW '21, page 484–495, New York, NY, USA, 2021. Association for Computing Machinery.

[68] Kwan Carmen, Janiszewski Paul, Qiu Shela, Wang Cathy, and Bocovich Cecylia. Exploring Simple Detection Techniques for DNS-over-HTTPS Tunnels. In *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet (FOCI'21)*, number 6 in FOCI'21, pages 37 – 42, New York, NY, USA, 2021. Association for Computing Machinery.

[69] Karel Hynek and Tomas Cejka. Privacy illusion: Beware of unpadded doh. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0621–0628, 2020.

[70] Jonas Bushart and Christian Rossow. Padding ain't enough: Assessing the privacy guarantees of encrypted dns. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI'20)*, 2020.

[71] Mozilla Manifesto. Firefox. https://www.mozilla.org.

[72] Fortiguard Labs. Fortiguard web filtering. https://fortiguard.com/webfilter.

[73] Alexa. Alexa top websites. https://support.alexa.com/hc/en-us/articles/4410503838999, 2020.

[74] Baiju Muthukadan. Selenium with python. https://selenium-python.readthedocs.io/, 2018.

[75] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL: http://jmlr.org/papers/v18/16-365.html.

[76] scikit-learn developers. Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html.

[77] scikit learn. Randomizedsearchcv. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[78] David Cournapeau. Scikit learn library. https://scikit-learn.org, 2007.

[79] Leo Breiman. Random forests, 2001.

[80] J. Zhu, H. Zou, S. Rosset, and T. Hastie. Multi-class adaboost, 2009.

[81] Xgboost classifier. https://xgboost.readthedocs.io/en/stable/.

[82] Microsoft. Lgbmclassifier. https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html.

[83] Gini impurity measure – a simple explanation using python. https://towardsdatascience.com/gini-impurity-measure-dbd3878ead33.

[84] R. Hill. Ublock. https://github:com/gorhill/uBlock.

[85] A. Mayrhofer. The edns(0) padding option. https://www.rfc-editor.org/rfc/rfc7830, 2016.

[86] M. Graff. Extension Mechanisms for DNS (EDNS(0)). https://www.rfc-editor.org/rfc/rfc6891/, 2022.

[87] Frank Olbricht. Routedns. https://github.com/folbricht/routedns.

[88] Confusion matrix. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.

[89] Fiona Nah. A study on tolerable waiting time: How long are web users willing to wait? volume 23, page 285, Tampa, FL, USA, Aug 4-6 2003. 9th Americas Conference on Information Systems, AMCIS 2003.

[90] Http archive. https://archive.org/.

A Appendix

A.1 Features List

Table A.1 shows features for the identification.

Table A.1: Flow Feature Selection

ID	Abbreviation	The flow discriminator
1	QueryPktCount	The number of outgoing packets
2	QueryIntervalTime	Query inter-packet arrival time
3	QueryIntervalTimeMax	Max query inter-packet arrival time
4	QueryIntervalTimeMin	Min query inter-packet arrival time
5	QueryIntervalTimeMean	Average query inter-packet arrival time
6	QueryIntervalTimeMedian	Median inter query arrival time
7	QueryIntervalTimeStandard	Standard deviation of inter-query arrival time
8	QueryIntervalTimeVariance	The variance in query inter-packet arrival time

9	QueryIntervalTimeCoefficientVariation	The coefficient of variance in query inter-packet arrival time
10	QueryIntervalTimeKurtosis	The "tailedness" of the distribution of query inter-packet arrival time
11-19	QueryIntervalTimeDeciles1-9	The deciles query-query intervals time
20	QueryLengthMax	Max number of bytes in outgoing packets
21	QueryLengthMin	Min number of bytes in outgoing packets
22	QueryLengthMean	Average number of bytes in outgoing packets
23	QueryLengthMedian	Median number of bytes in outgoing packets
24	QueryLengthKurtosis	The "tailedness" of the distribution of number of bytes in outgoing packets
25	QueryLengthStandard	Standard number of bytes in outgoing packets
26	QueryLengthVariance	The variance number of bytes in outgoing packets
27	QueryLengthCoefficientVariation	The coefficient of variance number of bytes in outgoing packets
28-36	QueryLengthDeciles1-9	The deciles bytes in outgoing packets
37	EntropyQueryLength	The normalized entropy of outgoing packet length
38	RPerSecondMax	Max number of response in one second
39	RPerSecondMean	Average number of query in one second
40	RPerSecondMedian	Standard number of bytes in outgoing packets
41	RPerSecondMin	Standard number of bytes in outgoing packets
42	RPerSecondStandard	Standard number of bytes in outgoing packets
43	RPerSecondVariance	The variance number of bytes in outgoing packets
44	PerSecondKurtosis	The "tailedness" of the distribution number of bytes in outgoing packets
45	RPerSecondCoefficientVariation	Max number of response in one second
46-54	RPerSecondDeciles1-9	The deciles number of response in one second
55	FlowBytesSent	The amount bytes sent
56	FlowSentRate	The rate of the bytes being sent
57	ResponsePktCount	The number of incoming packets
58	ResponseIntervalTime	Response inter-packet arrival time
59	ResponseIntervalTimeMax	Max response inter-packet arrival time
60	ResponseIntervalTimeMin	Min response inter-packet arrival time
61	ResponseIntervalTimeMean	Average response inter-packet arrival time
62	ResponseIntervalTimeMedian	Median response inter-packet arrival time
63	ResponseIntervalTimeStandard	Standard deviation of response inter-packet arrival time
64	ResponseIntervalTimeVariance	The variance in response inter-packet arrival time

65	ResponseIntervalTimeCoefficientVariation	The coefficient of variance in response inter-packet arrival time
66	ResponseIntervalTimeKurtosis	The "tailedness" of the distribution in response inter-packet arrival time
67-75	ResponseIntervalTimeDeciles1-9	The deciles of response inter-packet arrival time
76	ResponseLengthMax	Max number of bytes in incoming packets
77	ResponseLengthMin	Min number of bytes in incoming packets
78	ResponseLengthMean	Average number of bytes in incoming packets
79	ResponseLengthMedian	Median number of bytes in incoming packets
80	ResponseLengthStandard	Standard number of bytes in incoming packets
81	ResponseLengthVariance	The variance number of bytes in incoming packets
82	ResponseLengthCoefficientVariation	The coefficient of variance number of bytes in incoming packets
83	ResponseLengthKurtosis	The "tailedness" of the distribution number of bytes in incoming packets
84-92	ResponseLengthDeciles1-9	The deciles of bytes in incoming packets
93	Duration	Inter-packet arrival time
94	QRPktCount	The total number of packets
95	QRPktLength	The total length of packets
96	PacketThroughput	The throughput of packet length
97	ByteThroughput	The throughput of packet bytes
98	EntropyResponseLength	The normalized entropy of incoming packet length
99	FlowBytesReceived	The amount bytes received
100	FlowReceivedRate	The rate of the bytes being received
101	RQIntervalTimeMax	Max intervals between pair of response-query
102	RQIntervalTimeMin	Min intervals between pair of response-query
103	RQIntervalTimeMean	Average intervals between pair of response-query
104	RQIntervalTimeMedian	Median intervals between pair of response-query
105	RQIntervalTimeStandard	Standard intervals between pair of response-query
106	RQIntervalTimeKurtosis	The "tailedness" of the distribution of intervals between pair of response-query
107	RQIntervalTimeVariance	The variance intervals between pair of response-query
108	RQIntervalTimeCoefficientVariation	The coefficient of variance intervals between pair of response-querys
109-117	RQIntervalTimeDeciles1-9	Deciles of the query-response intervals time
118	QRIntervalTimeMax	Max intervals between pair of query and response packets

119	QRIntervalTimeMin	Min intervals between pair of query and response packets
120	QRIntervalTimeMean	Average intervals between pair of query and response packets
121	QRIntervalTimeMedian	Median intervals between pair of query and response packets
122	QRIntervalTimeStandard	Standard intervals between pair of query and response packets
123	QRIntervalTimeVariance	The variance intervals between pair of query and response packets
124	QRIntervalTimeKurtosis	The "tailedness" of the distribution of query-response interval time
125	QRIntervalTimeCoefficientVariation	The coefficient of variance intervals between pair of query and response packets
126	DFIntervalTimeMax	Max intervals between adjacent pair of query-response
127	DFIntervalTimeMean	Average intervals between adjacent pair of query-response
128	DFIntervalTimeMedian	Median intervals between adjacent pair of query-response
129	DFIntervalTimeMin	Min intervals between adjacent pair of query-response
130	DFIntervalTimeStandard	Standard intervals between adjacent pair of query-response
131	DFIntervalTimeVariance	The variance intervals between adjacent pair of query-response
132	DFIntervalTimeKurtosis	The "tailedness" of the distribution of intervals time
133	QPerSecondCoefficientVariation	The coefficient of variance of query number in one second
134	DFIntervalTimeCoefficientVariation	The coefficient of variance intervals between adjacent pair of query and response packets
135-143	DFIntervalTimeDeciles1-9	deciles of adjacent pair query-response intervals time
144	QPerSecondMean	Average the number of response in one second
145	QPerSecondMax	Max the number of response in one second
146	QPerSecondMedian	Median the number of response in one second
147	QPerSecondKurtosis	The "tailedness" of the distribution of the number of responses in one second
148-156	QPerSecondDeciles1-9	The deciles of query number in one second
157-165	QRIntervalTimeDeciles1-9	The deciles of query-response interval-time
166	QPerSecondMin	Min number of response in one second
167	QPerSecondStandard	Standard number of response in one second
168	QPerSecondVariance	The variance number of response in one second
169	EntropyRRIntervalTime	The normalized entropy of two consecutive responses
170	EntropyQRIntervalTime	The normalized entropy of pair of query-responses
171	EntropyRQIntervalTime	The normalized entropy of pair of query and responses
172	EntropyDFIntervalTime	The normalized entropy of pair of responses-query

173	EntropyQQIntervalTime	The normalized entropy of pair of query-query
174	bytes_ts3	The time to receive 3000 bytes response
175	bytes_ts5	The time to receive 5000 bytes response

A.2 The Feature of Query-Response

About the features related to query and response DNS packets, we consider three cases.

case 1 In the first case as shown in Figure A.1, the inter-arrival feature may include the QQ, RR, QR, and RQ. It causes some features will be overlapped. We extract all of these features.

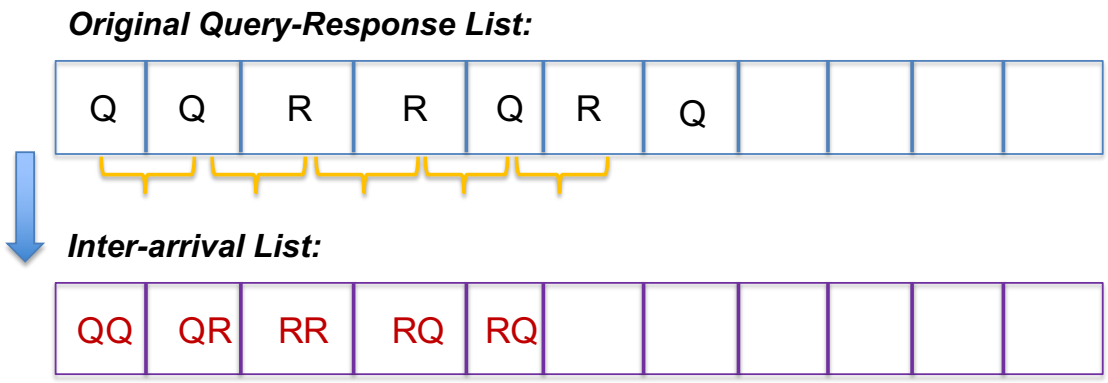


Figure A.1: First Case of Query-Response

case 2 We consider the intervals between an adjacent pair of a query and response (and vice versa):

case 3 As shown in :

QR inter-arrival[0] = Response[0] − Query[0]

QR inter-arrival[1] = Response[1] − Query[1]

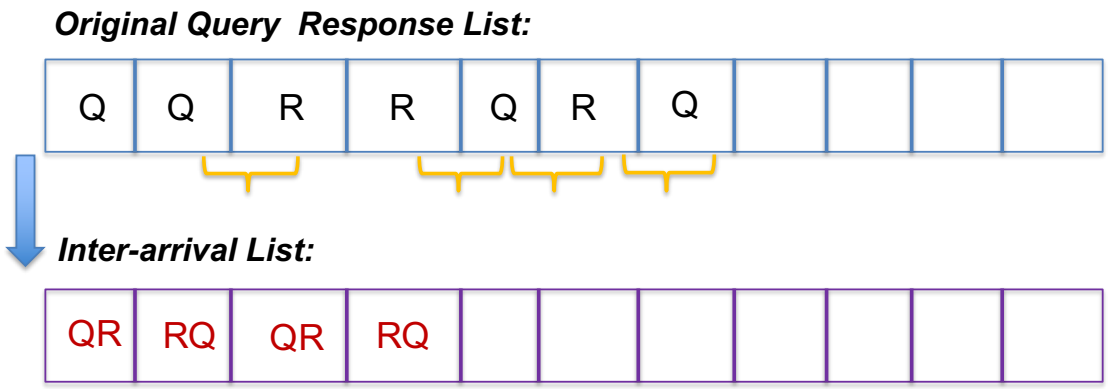


Figure A.2: Second Case of Query-Response

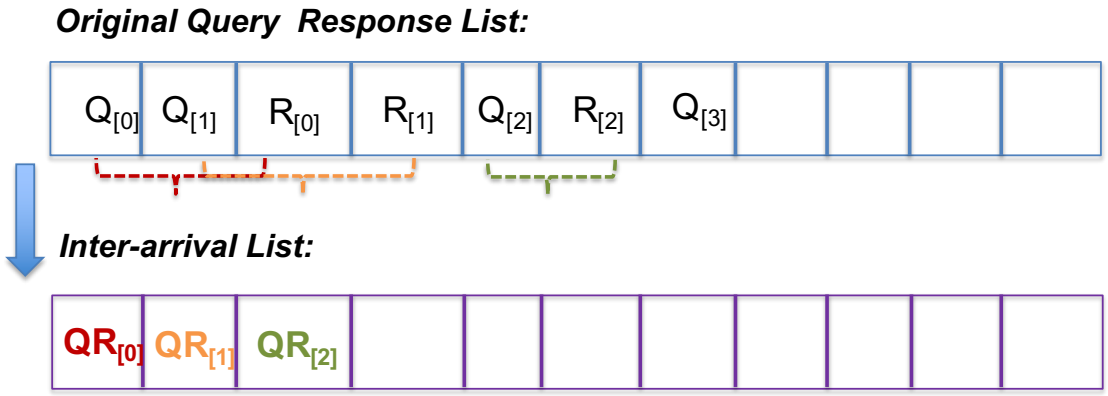


Figure A.3: Third Case of Query-Response

QR inter-arrival[2] = Response[2] − Query[2]...

In this case, the time of response may be smaller than the query (maybe one query will cause some responses, like A, CNAME.)

A.3 Padding Configuration of Bind9

Bind9 could support the pad in response by modifying the configuration.

In etc/bind/named.conf.options:

response-padding any;

block-size 128;