Analysing The Effect Of The Transport Protocol In The Success Of Website Fingerprinting

Andrew Ellison

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2024

Abstract

In 2022, the IETF published RFC9114 detailing HTTP/3, a new HTTP stack which uses a different transport protocol to both HTTP1 and HTTP2, called QUIC. With the introduction and widespread adoption of HTTP/3, the original Google transport protocol QUIC has now been established as a core element of Internet communication. TCP traffic has been shown to be vulnerable to website fingerprinting, a traffic analysis attack that seeks to breach the browsing privacy of a targeted user by observing patterns in their web traffic. This attack can be similarly employed against QUIC. In theory, the specific transport protocol used should have little effect on website fingerprinting, as this is just the method of delivery rather than the content, so it is not site-specific. This conclusion is generally held in peer-reviewed literature, with one notable exception, in which Zhan et al. identify a weakness in QUIC in website fingerprinting on early traffic.

This dissertation provides evidence that this weakness is no longer present, critiques of the methodology, and results that show the difference between the fingerprinting accuracy for both protocols is less significant than found by Zhan et al. This dissertation also uses hyperparameter tuning and AutoML techniques to explore website fingerprinting on each protocol, finding that the results for both protocols are similar. This suggests that in the finished product of the IETF's HTTP/3, there is no underlying flaw that is significant enough to give an attacker a significant advantage in fingerprinting HTTP/3 traffic to HTTP/2 traffic.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Andrew Ellison*)

Acknowledgements

Firstly, I would like to give a huge thanks to my supervisor, Marc Juarez, for always being ready to answer my questions and give feedback on my project. Thanks for the ongoing support throughout the year and for helping me grow my curiosity in website fingerprinting. I enjoyed our meetings greatly, so thanks for always making time for them.

Thanks to my friends and family for keeping me motivated and for the moral support, and a particular thanks to my Dad for proofreading the less technical bits.

Table of Contents

Chapter 1

Introduction

Website Fingerprinting is a traffic analysis attack that breaches a user's privacy on a network. It does this by training a learning algorithm to recognise the destination of a traffic trace based on features extracted from a user's traffic. Privacy is vital to protect online, as different parties can leverage it in many ways. Breaching a person's privacy online could be for less malicious reasons like targeted advertising but also more malicious ones like a government spying on members of its population for signs of dissent. Website fingerprinting involves "sniffing" the traffic of a network and recording the metadata of packets as they pass by. Because the attacker does not interfere with the traffic in any way, this attack is essentially untraceable. Furthermore, as it is powered by machine learning, it is likely that, as hardware and learning algorithms improve, this attack will become more potent. Therefore, research into website fingerprinting and ways to defend against it as it evolves is critical.

The transport protocols analysed in this paper are QUIC and TCP within the HTTP3 and HTTP2 stacks, respectively. These are the website delivery protocols most commonly used on the Internet. QUIC is a relatively new protocol but is becoming popular fast due to the performance benefits it brings to internet communication; therefore, it needs to be studied. It is essential to know that HTTP3 is as secure as HTTP2, and this paper will focus on the privacy protection of these protocols and how well they resist website fingerprinting attacks. This project is not the first paper in the field, although this area has yet to be researched in depth. Among these papers, the general conclusion is that HTTP2 and HTTP3 are as good as each other; however, in 2021, Zhan et al. [59] published a paper regarding a discovered weakness in the early traffic of QUIC when targeted with website fingerprinting attacks called "Website Fingerprinting on Early QUIC Traffic".

This conclusion is surprising, as intuitively, the transport protocol used to communicate across a network should have an insignificant effect on website fingerprinting. To understand this, picture the analogy of a postman delivering parcels and a group of "porch pirates" who try to determine if a package is worth stealing. When choosing a parcel to steal, the pirates will look at the size and shape of the package; for example, when seeing a large rectangular parcel that seems heavy to lift, a pirate can infer that this parcel is a new TV and an ideal target for stealing. What is not useful to the

pirates is the delivery service used to deliver the TV, as this does not tell them about the package because the delivery process would be the same regardless of the parcel being delivered. Even if a different delivery service is used, the purpose and general process are the same; given the same package, different delivery companies may have slightly different delivery methods, but the package delivered is still the same. In this analogy, the parcel is a person's web traffic for a specific website, so a collection of individual packets and the metadata of these packets would provide an attacker with information to analyse. In a website fingerprinting attack, rather than looking for lucrative parcels to steal, an attacker is trying to work out the actions of a network user, for example, what websites or website sections they are visiting. However, the point remains that the delivery service, or the transport protocol in a website fingerprinting context, should not have much effect on determining where the traffic is visiting.

This leads to this paper's hypothesis that the transport protocol should have an insignificant effect on the success of website fingerprinting.

The project's objective is to explore if there is a fundamental weakness in QUIC traffic that makes it more susceptible to website fingerprinting than TCP-based traffic. In an attempt to achieve this objective, this project makes several contributions. Firstly, this paper critically analyses the website fingerprinting on early QUIC traffic paper to determine the strength of their conclusions. This project also contributes a more rigorous approach to cleaning traffic data than the original paper to ensure that our conclusions are strong. Next, this project reproduces Zhan et al.'s experiments and finds results that contradict those in the original paper, leading to opposing conclusions about the weakness of early QUIC traffic to website fingerprinting. This paper shows that the results found in the Early QUIC paper are limited to experimental versions of QUIC and that following QUIC's standardisation in HTTP3, the patterns shown in the paper are no longer present. Unlike Zhan et al., this paper includes standard deviations, where available, so the significance of the results is verifiable.

Furthermore, unlike the Website Fingerprinting on Early QUIC Traffic paper, this paper also provides all source code and docker files to facilitate the reproducibility of the results. This paper also builds from the original paper to explore differences in fingerprintability by tuning the models used to check if the hyperparameters that make an effective fingerprinting model for HTTP3 are the same or similar to the hyperparameters for HTTP2. Again, this is taken one step further by using AutoML. AutoML refers to systems to automate machine learning tasks such as selecting models and training hyperparameters. This paper uses AutoML to better understand whether the problem of classifying HTTP3 traffic is the same or similar to classifying HTTP2 traffic.

Chapter 2

Background

2.1 Definition of Key Terms and Ideas

Attackers can intercept and view the contents of packets that are routed through a router they have access to. This is known as **sniffing** but given that internet protocol packet payloads are standardly encrypted, they can usually only access the packet header, which should not include any sensitive information of the sender.

The **TCP/IP** stack is a layered model that defines how to connect two devices across the internet (and other similar networked systems). Each layer is a grouping of protocols by purpose. The general model has four layers: the Application layer, the Transport layer, the Internet layer and the Link layer. This paper focuses on the transport layer, but it is also useful to understand the application layer. The **application layer** is the domain of applications and processes that wish to communicate with other applications on the network. This includes most user protocols for network communication such as the **Hypertext Transfer Protocol**, which is used for browsing websites and is foundational to the World Wide Web. The **transport layer** is responsible for establishing a host-to-host connection to pass messages across. Typically, a connection-based protocol called **TCP** will be used, but there are also connectionless protocols like **UDP**, which can transfer data without establishing a connection between hosts.

2.2 Website Fingerprinting

2.2.1 The birth of Website Fingerprinting

When the Internet was conceived, it was never intended to be anonymous; this meant that for every packet sent over the Internet, the source and destination IP addresses and website domains were visible [14][13]. In these scenarios it is considered that the content of the packets is encrypted (and cannot be decrypted) as is internet standard [5]. In the modern day, most people access the Internet from local area networks with multiple users, so there is some inherent anonymity [31][25] but this is not an invincible protection particularly considering an attacker who has visibility over this Local Network such as an Internet Service Provider (ISP). The IP addresses can be

used to deanonymise users, as from any router in the packet's path, we can identify where the packet has come from and where it is going, making it straightforward to track an internet user's behaviour. As a response to this problem (and others [26]), The Onion Routing (Tor) browser used its protocol of onion routing to dissociate the origin and the destination IP address from routers on a packet's path. The implementation of onion routing means a local adversary sniffing packets can only see the origin or the destination, but not both at the same time, helping to keep users' browsing history anonymous.

However, as machine learning capabilities developed, a new kind of attack was introduced. Rather than simply observing the packets' addresses as they pass by, an attacker uses a machine learning model to analyse the traffic pattern and create a 'fingerprint' corresponding to a specific website. After being trained to identify specific website traffic, this machine learning model can be used to deanonymise a user's behaviour. With reasonable accuracy (over 90% in some scenarios) [56][29][43] an attacker can now identify the websites a target visits. However, papers tend to make assumptions about settings and scale that may not hold in the real world [34][39], this means that this type of attack may not be as effective in realistic open world, large scale scenarios where traffic patterns are not quite so revealing. Because of the nature of this attack, we no longer need to see the IP addresses of the packets as features of the traffic are sufficient. As well we should also consider that while website fingerprinting may be limited at scale as of today, as hardware and software evolve this may not always hold true.

Many papers involving website fingerprinting will use Tor [57] [30], VPNs [35] [44], or both [32] in their setup as these tools are designed to make internet traffic more anonymous. Hence, an attack to deanonymise users is a more critical problem than it would be for regular web traffic. However, as most internet users do not use these tools [12] [10], it is helpful to consider the scenario in which neither of these frameworks is employed.

2.2.2 What is it?

The aim of an attacker in a website fingerprinting attack is to discover a user's activity over the Internet based on network traffic header information. Over Tor, this attack is performed through the systematic collection of traffic information to train a classifier to recognise network traffic features to identify a website given a website traffic trace. The collection of features of a traffic trace is known as a "fingerprint", so the practice of creating these fingerprints is known as website fingerprinting. An attacker can fingerprint from anywhere in the packet's path, as the condition for fingerprinting is that an attacker can read a packet's unencrypted metadata. Some features have been demonstrated to be more effective for website fingerprinting on Tor than others; features that have proven to be effective are unique packet size [38], packet size count [32], packet order [22], and burst size [53] [56] which has shown lots of promise recently. Website fingerprinting is passive, this means that the attacker does not modify or interfere with the traffic in anyway making it effectively undetectable on a network. Attackers are also generally considered to be local, so they can only observe traces

passing through their local area, such as a single router in a system, rather than having access and the ability to monitor the whole network.

Website fingerprinting is generally approached as a supervised learning problem. This means that the input data used will be labelled with a desired output, in the case of website fingerprinting, this will be a domain or different sections of a domain. The first step of supervised learning is to identify the problem and collect relevant data, which will be used during training and testing. For website fingerprinting this will be traces of visits to a selection of websites. The next step is feature selection: features are chosen from this collected networked traffic to distinguish these traces. In website fingerprinting, an obvious feature would be the size of the packets being sent in a trace. Next, a learning algorithm is required such as a Random Forest, which will be used to learn the structure of our collected training data and make predictions when provided with new data. The No-Free-Lunch theorem implies that there is no learning algorithm which is better than any other algorithm when considered on all types of problems, so the most appropriate or best performing model will change from problem to problem. Finally, the model is trained using the data, and then evaluated using the collected test data (which was kept separate from the training data). The resulting model can then be used to predict where a user is visiting from our original list of locations.

Website fingerprinting is considered relevant to any adversary who seeks to exploit private user browsing data. The Adversary Model in Chapter 3 provides a more in-depth discussion of an attacker in a website fingerprinting context.

2.2.3 Measuring Website Fingerprintability

The main measurement used in Website fingerprinting is classifier accuracy, this is the measure of how often the classifier gets the correct result in a provided data set, so whether the predicted domain is equal to the actual domain from which the traffic was collected.

Website fingerprinting attack and defence papers tend to use accuracy since it is widely used in the general field of supervised learning classifiers, but this may not tell the whole story regarding the performance of a fingerprinter. Another possible aspect is the information leakage of these attacks and defences in combination with the accuracy of the models, as a low accuracy website fingerprinter does not necessarily mean low levels of information leakage, thus there is space for improvement in the attack [37]. A key measurement we use in evaluating information leakage is Entropy. Entropy is a measure of the average levels of unpredictability of a system, this is useful as there is a fundamental connection between entropy and lack of information [40]. This measure of information was first pioneered by Shannon in 1948 & 1949 and has been used widely in website fingerprinting research, but convincing arguments have been made for more refined measures such as Renyi entropy [23].

2.2.4 Moving away from Tor

Although Website Fingerprinting (WFP) was developed to attack Tor, it has also been shown to be applicable to general internet traffic. Website fingerprinting on general

HTTPS traffic provides additional information about a client's activity online. Although now the header of a packet can be checked to find the source and destination of the packet. Still, with website fingerprinting, we can use just information available in the packet metadata to identify a specific area of a website (and sometimes actions taken on the website). This is useful as more specific data about a client's activities on a website is now available.

WFP on non-tor domains can further compromise a user's privacy as it provides a more comprehensive model of their behaviours on the web. While this data can be used maliciously it is also useful to research website fingerprinting to assist experts in producing stronger defences and countermeasures, as well as to raise the awareness of this kind of attack.

2.2.5 Content Distribution Networks

Another important factor to consider in modern web traffic is the wide usage of Content Distribution Networks (CDNs). In brief, a CDN is a group of servers that caches content close to end users; they have grown massively in popularity and in the modern-day, most content is served through CDNs, including from major sites like Netflix, Facebook, and Amazon [2]. The use of CDN caching technology hinders traditional fingerprinting feature extraction and attacks because it is an intra-domain WFP problem rather than inter-domain, which means distinguishing data within the same domain rather than across domains. Intra-domain fingerprinting is generally more difficult as the content on a CDN or within a domain will usually follow a similar pattern, i.e. on a typical social media site, parts of the page are generated using the same template, thus exhibiting similar traffic [55]. Therefore, new fingerprinting methods that use the DNS resolution sequence in a CDN fingerprint have been developed, which can effectively identify HTTPS websites in a CDN cache [54] [55]. Furthermore, centralisation of resources in this way has been shown to be favourable for attackers utilising website fingerprinting, particularly for Google's CDNs [46]. However, as a response to this, Cloudflare, one of the largest providers of CDN services, has introduced Encrypted Client Hello (ECH), which is designed to encrypt the Server Name Indication (SNI) that is used to negotiate a TLS handshake [1]. This will help protect handshake metadata from observers, such as those wishing to perform a website fingerprinting attack.

2.2.6 Communication Layers and Fingerprinting

A website fingerprinting attack physically takes place at the link layer, through the physical connection between two endpoints (Fig. 2.1) but through the communication process, different layers have different effects on the fingerprintability of traffic.

The application and network layers are the most critical in their effect on WFP. The network layer is responsible for packet routing, so the traffic patterns are ultimately produced at this layer. This is why many WFP defences target this layer to try and alter the traffic somehow to make it less distinguishable. The application layer has the largest role in website fingerprinting; although attackers do not have direct access to this layer due to encryption, the webpage-identifying features are introduced at this layer, such as

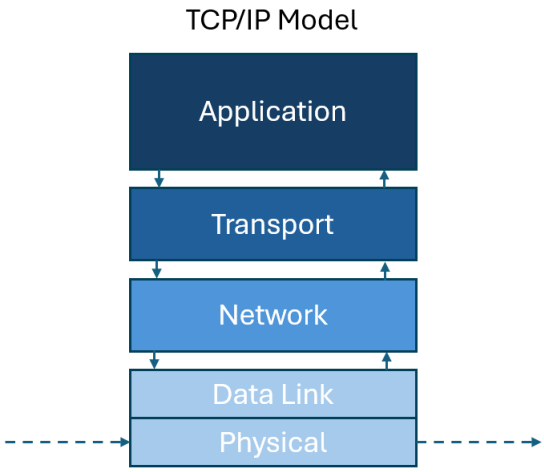

Figure 2.1: TCP model

the total webpage size in HTTPS, which could be leaked through the web traffic traces [20].

2.2.7 Fingerprinting the Transport Layer

In between the application and the network layers is the transport layer; this layer is responsible for organising how packets of data from the application layer are sent from a source to a destination. In theory, this layer is not helpful for website fingerprinting as almost all traffic uses the same transport protocols, so no website-unique information should be revealed at this layer. However, with the widespread introduction of HTTP/3 (HTTP over QUIC), QUIC transport is becoming increasingly ubiquitous over TCP which has been standard since HTTP/1. Although the mixture of these two schemes has been shown to have a positive effect on protecting against WFP [49], the principle remains that if most network traffic uses these transport protocols, there should be little effect on distinguishing an individual's web traffic.

2.2.8 Defending against WFP

While we cannot perceive the real-world usage of website fingerprinting as it is undetectable, we do know that similar traffic analysis attacks were revealed in Snowden's 2013 NSA leaks [42]. Therefore, it's likely that intelligence agencies may use website fingerprinting on specific targets. Given this as well as its specific threat to more anonymity-focused browsers like Tor, several other defensive techniques have been created to defend against WFP specifically. Many WFP defences are often designed at the network level, but these defences have been shown to require substantial and arguably unrealistic changes in infrastructure [20].

WFP defences are a trade-off between security and overhead, such as packet injection increasing latency by using up bandwidth with extra packets of data. This results in the question posed by Cai et al [18]: How efficient can a defence be while offering a given level of security?

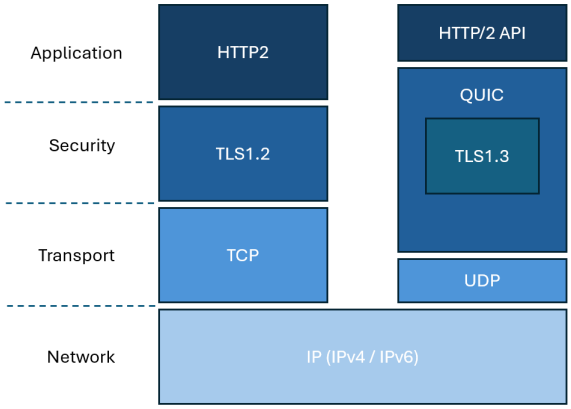

Figure 2.2: The HTTP/2 vs. HTTP/3 stack

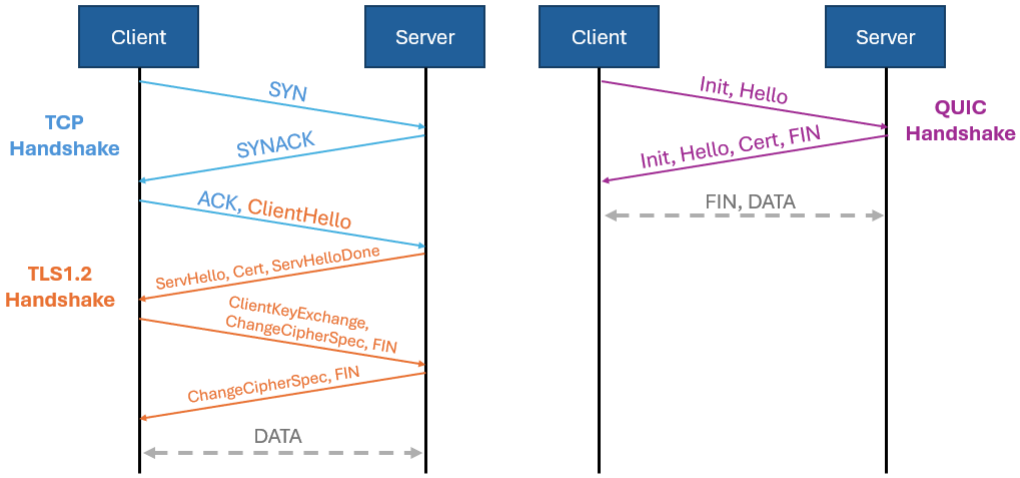

Figure 2.3: The HTTP/2 vs. HTTP/3 handshake

Many defences exist [17][18][27][38][32][58] but this is something that won't be explored in this project, as the focus here is on just the transport protocols and their effects on website fingerprinting.

2.3 QUIC

This paper discusses the effect of transport protocols on website fingerprinting vulnerability. Specifically, the transport protocols discussed are TCP and QUIC. An overview of QUIC is provided, along with some specific features of QUIC to provide insight into the general mechanism, and previous work discussing previous analysis of website fingerprinting in a QUIC context.

In 2012, Jim Roskind at Google designed, then with assistance developed and deployed a new transport protocol, designed to improve performance for HTTPS traffic and allow quicker and easier deployment of new transport mechanisms [9] [36]. This protocol was

publicly announced later in 2013 as QUIC (Quick UDP Internet Connections). QUIC was designed to solve key issues with its predecessor TCP, namely the head-of-line (HOL) blocking problem. In recent years, QUIC has been adopted by the IETF, resulting in the publication and widespread use of HTTP/3 (HTTP over QUIC). This is important in the history of the Internet as it was the first major update to the Hyper Text Transfer Protocol (HTTP) since 2015, so this change in transport protocol brings whole new opportunities, both positive and negative, for the wider Internet.

2.3.1 What does QUIC replace in the communication stack?

QUIC spans multiple existing layers (Fig. 2.2). We can see that QUIC spans the application layer due to its user-space implementation (as opposed to previous HTTPS implementations implemented in kernel-space). In the application layer, we still rely on HTTP/2 for HTTP protocol parsing - but rely on QUIC to fulfil tasks traditionally done by HTTP/2, such as multiplexing and link management [60]. As for the Security layer, RFC9001 section 4 [52] details QUIC's encapsulation of TLS1.3 and "how TLS acts as the security component of QUIC". Unlike the traditional HTTPS stack, TLS is not decoupled from the other protocols and uses the QUIC 0-RTT handshake. Finally, QUIC is also situated in the transport layer over UDP, where it builds on top of UDP, which a more bare-bones transport protocol than the common TCP. Over UDP they implemented congestion control and several other features [9].

2.3.2 HTTP/3

Both HTTP/1 and HTTP/2 use TLS and TCP over IP (Fig. 2.3), while as of RFC9114 [16] IETF have set out their design for HTTP/3 that uses QUIC as the Proposed Standard for exchanging data on the World Wide Web. Following this, support for HTTP/3 / HTTP-over-QUIC was added to Chrome and Firefox in 2019, which became enabled by default in later years.

This process began with the IETF QUIC working group publishing a QUIC version 1 in RFC 9000 [33] where they standardise the protocol initially set out by Google (gQUIC). gQUIC was monolithic, meaning that the transport and cryptographic handshakes were coupled, it was also designed as a general-purpose transport protocol. gQUIC was then adapted for use with HTTP by Google and the general community and following this in 2016 an official IETF working group was established to build what would become HTTP/3.

QUIC's development by the IETF working group led to several redesigns as they sought to retain all the benefits of gQUIC while making the protocol suitable for widespread adoption [33] [52].

Because HTTP/3 uses TLS1.3 it means that a new feature was introduced called "zero roundtrip time connection resumption" or 0-RTT. This allows clients to start sending data without waiting for the TLS handshake to complete, reducing latency produced by establishing new connections [36] [6]. If the client and server have previously held a TLS connection, they can use data from that session to create a new one rather than establishing new connection parameters from scratch [3].

As HTTP/3 is being widely deployed over many user cases, it is very important that any issues with QUIC are discovered and resolved, as they will greatly impact the World Wide Web if left unchecked.

2.3.3 Key Features of QUIC

QUIC solves several problems associated with HTTPS (and its usage of TCP). To help understand the importance and design of QUIC, we will discuss some key features in comparison to TCP, although it's important to note from the previous section on the role of QUIC in the communication stack that while we are comparing it to TCP, that they do not fill the exact same space in the stack.

Head-of-line Blocking: One of the most prominent issues with TCP is the head-of-line (HOL) blocking problem. When a client sends packets over a network, they want to be sure all the packets of data sent successfully arrive at the provided location. TCP uses a single channel for communication; this means that if a packet is lost in the stream of traffic, all the packets behind the packet in the channel will have to wait for that packet to be sent and successfully received. This hold-up of all other packets is called the HOL blocking problem and is a big issue as it slows down data transmission. In QUIC, multiple channels are set up simultaneously between a source and destination, so if a packet is lost, only the packets within that single channel will be held up, and all other channels will transmit freely, thus providing faster transmission times. QUIC is not the first attempt to solve the HOL blocking problem, but *is* the most successful so far [8].

UDP: While previous versions of HTTP were implemented over TCP, QUIC is implemented on top of UDP. In short, TCP requires the setting up of channels to ensure the integrity of data transmission, however, this comes with an associated overhead, making the process of communication slower. UDP is more bare-bones: this means that it avoids this channel setup overhead as packets, as UDP does not require prior communication to set up channels or paths.

User Space: Previous versions of HTTPS were based in the operating system kernel, this means that it is difficult to evolve any transport protocol as it would require clients and servers to update their operating systems and deploy new versions, something that is notoriously difficult as it requires lots of collaboration between parties. With QUIC being implemented in the user space, this allows for much faster updates and iterations of the protocol, meaning that it will now be much easier to implement different implementations of QUIC, which could result in better performance or better tailoring of the protocol to individual needs.

Protocol Ossification: Middleboxes are devices set up in between two network endpoints, they are used to ensure that the Internet works as it is intended to. They perform many different functions and have thus as the Internet has grown, have required more and more about the networks and protocols they monitor. One large issue brought about by these middleboxes is Protocol Ossification. Because middleboxes are so widely used and when initially deployed were not designed to change their minds about what is permissible traffic. This means that any new behaviours introduced in attempts to solve newly found issues or to add new features to existing protocols could end up being

blocked by these middleboxes. This is especially true for TCP, where changes will often be blocked as the middleboxes have become so used to how TCP traffic normally looks, for example, TCP Fast Open was a protocol developed to provide speed benefits to TCP yet has shown to be difficult to deploy [41] [11]. Another feature of QUIC is that it encrypts packet transport headers, this effectively prevents ossification as middleboxes can no longer understand and thus make decisions on how to handle these packets [21].

Padding Frame: To help defend against traffic analysis attacks, the QUIC working group introduced a Padding frame into its official specification [45] [33]. This can be used to increase the size of a packet, making patterns in the traffic less distinct. This packet frame has also been used in the implementation of client-side frameworks to implement existing WFP defences [45] [48].

Conclusion: From this we now know that QUIC is a fast, secure, and more future proofed transport protocol to predecessors. However, these new features may result in weaknesses in privacy that may not yet be fully identified or understood.

2.3.4 Existing QUIC implementations

Because QUIC is implemented in the user-space (see Section 2.3.3) and has been standardised by the IETF this has allowed for easier creation of implementations of QUIC. To view these implementations, see the GitHub page for the QUIC working group [7]. Some important implementations to note are the Chromium implementation deployed on Chrome versions 85.0.4171.0 and later, and Neqo, the Firefox version of QUIC and HTTP/3.

2.3.5 Website Fingerprinting QUIC

As mentioned previously in Section 2.2.7, the transport layer should have little effect on website fingerprinting. Still, in the essence of comprehensive risk mitigation, research has been conducted into how well QUIC fairs against website fingerprinting. QUIC is generally considered on par with TLS/TCP in the normal traffic scenario; this is primarily due to Smith et al., who have studied it in comparison to TCP using a large open-world dataset using a large range of features [49]. Although there are many evolving attacks and defences, so it is hard to be sure this conclusion will remain long-term.

In contrast, Zhan et al. [59], find that QUIC could be vulnerable in the early stages of communication. This defies intuition, as the transport layer should have no effect on fingerprinting effectiveness. This result prompts further research into the paper's findings on whether the QUIC protocol somehow sabotages the network layer, making fingerprinting easier or if it is simply due to an immature codebase where issues will be ironed out given time. Research in this area is vital because, as stated, QUIC has been introduced with HTTP/3 and is growing in popularity. However, suppose this is a genuine weakness compared to previous transport protocols. In that case, it must be addressed as it could lead to a widespread decline in internet privacy. It should be noted that Zhan et al. do not explore the normal traffic scenario in depth; they focus on the early traffic, which they define as the first 40 packets of a trace.

Chapter 3

Adversary Model

This project will consider WFP (Website Fingerprinting) in a normal HTTP scenario; this means that privacy tools such as VPNs or Tor, which manipulate the addresses of packets, will not be used. However, it is assumed the body of the packets is encrypted, as is standard for modern web traffic, and that they cannot be decrypted without an unreasonable amount of effort by the attacker. Therefore, the attacker must use the packet metadata to perform an attack.

3.1 Adversary Goals

The aim of an attacker in a website fingerprinting attack where no privacy framework is used is to breach the privacy of a targeted individual. This is useful to parties interested in what an individual is viewing online. As no privacy framework is employed, the attack would generally be intra-domain, where the attack reveals information about the actions or specific areas of a domain being visited. This differs from the courser-grained inter-domain attack used on services like Tor, which reveals which domain an individual visits. While this paper discusses only regular traffic without using privacy-enhancing frameworks, it still focuses on **inter-domain fingerprinting**. This is because the Website Fingerprinting on Early QUIC Traffic paper uses inter-domain fingerprinting, and this paper seeks to recreate their results. Justification for this decision is included later in Section 4.2.2.

An example attacker for a website fingerprinting attack over HTTPS using no extra privacy tools could be an oppressive government monitoring a person of interest, visiting websites like WikiLeaks to see what areas of the page they are looking at, intending to find incriminating behaviour or evidence of dissent against the government.

3.2 Adversary Capabilities

This paper also considers a **passive** and **local** adversary. This means that an adversary can sit locally on a gateway and view the headers of packets as they pass by. They are passive, so they do not interfere with the packets in any way. They can then feed

features based on this collected header information into a pre-trained classifier, which will output the domain. Because of the passivity of the attacker, they are essentially undetectable during an attack. We will assume that the adversary has sufficient ability to collect traffic from a node, and to train a classifier.

3.3 Attack Environment

This paper uses the **closed-world** scenario like in the Website Fingerprinting on Early QUIC Traffic paper. The closed-world scenario assumes that an attacker has a list of websites for which they know the metadata information and that the victim can only visit these websites. This usually includes not sending the traffic over the internet but rather keeping it contained to a local network so that there is no interference from other traffic, as this is simpler to implement. However, a real attacker would conduct this attack on a large open network. Taken from [24]: "The closed world model is used when complete information is available. A closed-world model assumes that an attacker knows the metadata information for a select list of websites. The website visited by a victim is in the list known by the attacker. It is a strong assumption used to simplify the threat model, implement the experiment, and evaluate an attack's success." Most research is conducted using the closed-world model [17][18][19][32][38][51][50].

This paper also considers the early scenario for some experiments. This is defined by Zhan et al. as the first 40 packets [59], which means there is a slight change to the adversary. Initially, this would not make sense in an undetectable attack, as there is no more risk of collecting 200 packets than there is to collecting 40. However, this scenario is not as unlikely as it would first seem if scalability is accounted for. While deep learning on lots of traffic produces the most effective fingerprinters [47] even on smaller traces of traffic [15], it also comes with much larger computation and storage overheads. This is important when scaling the attack up and when attackers have limited resources. Tian et al. produced a paper regarding this [53], proposing a new attack with lower associated costs, thus designed for scalability. Zhan et al. also consider this, using only four size categories and directions when fingerprinting for 40 or fewer packets. A fingerprinter that uses only the first 40 packets would indeed be easier to scale and use, even with limited resources.

Chapter 4

Critical Review

This chapter discusses some of the key design decisions performed in the *Website Finger Printing on Early QUIC Traffic* (WFP on Early QUIC) paper, highlighting those that are sound, and potential ambiguities where assumptions have been made about their methods. Differences between their methods and this paper are described in the Methodology chapter, where changes from their original methodology are justified using the discussion and critiques from this chapter. This chapter is split into three sections, all focusing on the WFP on Early QUIC paper:

- How did they collect data?
- How did they handle that data?
- How did they evaluate the results of experiments using this data?

Only the key contentious design decisions are included in this chapter, there are other design decisions that could be discussed but they are less relevant.

4.1 Collection Aims

The Website Finger Printing on Early QUIC Traffic paper (WFP on Early QUIC henceforth) explores the vulnerability of GQUIC, IQUIC and HTTPS and thus collects encrypted traffic that simulates the visits of a real user to real websites. The three stated conditions they want their data to meet in their paper are to (1.) be large enough to be indicative of patterns in the real world, (2.) include sets of traffic for GQUIC, IQUIC and HTTPS from the same domains and (3.) to be collected how a real adversary would.

Aims 1 and 3 are logically sound and ensure realistic results, showing that the conclusions of the paper should be reflective of real-life scenarios. However, the choice to collect GQUIC, IQUIC and HTTPS is a little outdated and most likely due to the paper being published in the early years of QUIC. By modern standards, the HTTPS they refer to would be HTTP2 and IQUIC has now become standardised as HTTP3, throughout this chapter, IQUIC will be referred to as HTTP3 or (HTTP over) QUIC and to their HTTPS as HTTP2 or (HTTP over) TCP. Next, their choice to collect GQUIC traffic makes less sense as this edition of QUIC is not used over any public networks (although

it may still be used in Google internal systems), so it would never become the target of a fingerprinting attack.

4.2 Testbed Design Choices

The WFP on Early QUIC paper makes several design decisions they use when collecting their testbed of data. This section contains some discussion of some of the overarching design decisions, where finer details will be discussed in the later chapter regarding my implementation of their experiments. This section focuses on the key choices in their design that are the most contentious or where other strong alternatives are available.

4.2.1 Closed World

The WFP on Early QUIC paper collects their data under a closed-world environment, as opposed to sniffing traffic directly from the internet (*open world*). They provide two justifications for this: (1.) to avoid interference and (2.) as many sites do not support all three of their chosen protocols. Initially, this first justification is not particularly strong given their aims, why choose to avoid realistic traffic when a stated aim is to collect realistic traffic? In their justification they give the specific example of wanting to avoid RST packets in TCP (for HTTP2), this provides more insight as this paper centres on website fingerprint in the *early* traffic scenario. This includes creating a fingerprint when given just the first 5 packets of a trace. Given this their justification is better as they don't want random RST packets diluting their small collection of data, this would also be easy to implement in a real attack. Their second point is sound, particularly at the time of writing that paper in 2019 before HTTP3 became much more standard and is justification enough to not collect real internet traffic. Although this is much more feasible in the modern era this will not be explored in my report (however is explored more in depth in other papers [49][46][28]).

4.2.2 Inter-domain

In the WFP on Early QUIC paper, they select the landing pages of the top 100 universities in the 2019 Times World Rankings. Although not directly addressed, this means the paper explores inter-domain WFP over intra-domain WFP. The choice here is interesting as they provide no justification for it, which is not abnormal given that this is the standard for website fingerprinting attacks in the literature. However, the reason this is the standard is that website fingerprinting was conceived to perform interdomain fingerprinting on Tor, while this paper does not use Tor and instead performs fingerprinting on regular traffic. As source and destination IP addresses are both accessible in normal traffic sniffing, this would remove the need to fingerprint as we can already see the destination of every packet. As mentioned in the adversary model 3, fingerprinting on normal traffic has a different aim use case, which is to see the sections of a domain a user visits or interacts with, such as what they interact with on a social media platform which could reveal personal information about themselves such as their personal habits, or even their political leanings [55].

The implicit assumption here is that any discovered weakness in inter-domain fingerprinting in HTTP3 will transfer to intra-domain fingerprinting. This is reasonable, as the weaknesses making QUIC more fingerprintable will not disappear even in a different scenario, however, it would still be more appropriate to measure intra-domain fingerprinting in this context if the aim is to mimic the actions of a real attacker.

4.2.3 Secure Gateway

In the WFP on Early QUIC paper they use a secure gateway for two stated reasons (1.) to act as a reverse proxy and (2.) to update the local forwarding policy. The latter serves the expressed purpose of "block access from a client to a specific website" "based on the results of WFP attacks" [59] (section 4.1). This is ambiguous, as they don't explain *why* they would want to do this; this is interpreted to mean that while loading their *offline* website, additional resources may be requested from *online* websites, which they have stated they want to avoid. This is well-grounded as communication with these sites would most likely not use the transport protocol they want to capture in isolation. The former reason they justify the use of a reverse proxy is that it prevents adversaries from getting information about the web server from the packets. Therefore, this is posed as a decision to ensure realistic adversary conditions. Using a reverse proxy won't affect fingerprintability, as the attack discovers this information anyway. Although not included in the paper, this decision is more likely for convenience, as they split data across three web servers, they use a reverse proxy to manage connections, which should speed up the collection of their results. This is a rational decision but is misleading to pose as a realism choice, when this choice is tangential to fingerprinting and when the collection is already being performed on a local private network.

4.3 Data Handling

4.3.1 Cleaning the Handshake

When tailoring their traffic, because QUIC has a shorter handshake than TCP (1-RTT compared to 3-RTT) in the WFP on Early QUIC paper, they remove the handshake from the beginning of all traffic. They don't explicitly state why, but it is easy to assume that this is because the handshake is uniform across every domain and, therefore, is not useful when it comes to website fingerprinting, as it does not leak website-specific information. Therefore, they want a fair comparison of just the traffic that will leak information. However, as the handshake process is intrinsic to each protocol, it is important in a comparison between the protocols to include the handshake process. Because the paper considers the early traffic scenario where only a small number of packets are collected to WFP, it seems counterintuitive not to include the handshake, as a longer handshake should make a protocol more WFP resistant, given an attacker can only see the first n packets. Another sensible option would be to remove the handshake but only consider the first $n - |handshakepackets|$ when fingerprinting the early traffic.

Ultimately, as the handshake process is a small section of the whole traffic collection, for normal scenarios, it should not make a significant difference to the results, but it is

more reasonable, given the adversary model, to include it when comparing transport protocol.

4.3.2 General Cleaning

In the original paper, they mention that they delete invalid and error pages, but do not go into detail about how this is done. This is relevant as in the versions of Caddy (webserver) and Chrome (browser) they use, QUIC is still an experimental feature, so we could expect more errors and perhaps less obvious errors than a more mature protocol like TCP.

4.3.3 Poorly Defined Features

In the WFP of Early QUIC paper, the definitions for the features they use are ambiguous and poorly defined. For example, they never reference whether only the first n packets are used when extracting a feature. It is assumed that they do use this in their definitions otherwise it would defeat the purpose of their paper, but it is not clear. As well, they sometimes use strange terminology such as *cumulative sum*, when it would be more accurate to use *sum,* this is important as the cumulative sum would be n-dimensional and the sum is 1-dimensional, so must be inferred from their definition, which is already vague.

4.4 Evaluation

4.4.1 Top-k accuracy

In the original paper, they call this top-a accuracy, here it will be referred to as top-n accuracy as this is the standard name used but it can also be known as top-k accuracy (as n is used elsewhere to describe the number of packets, it will be referred to as top-k in this paper). Also, they call their fingerprint attack that uses top-k a top-a attack; this also is not standard as, in reality, top-k would never be used in an attack.

The reason they use top-k in the WFP on Early QUIC paper, is that it exposes a weakness in QUIC when increasing the value of n, which is not present in HTTP2 (TCP).

The use of top-k could be criticised for making results appear more significant than they are, for example, a 99% (top-5) accuracy is much more significant than a 10% (top-1) accuracy. However, in a website fingerprinting context, 10% accuracy would mean our attack is weak, as it cannot guess the correct website being visited enough to be useful to an attacker. In a real-world scenario, top-k would not be used in website fingerprinting as its usage significantly reduces the information leaked from an attack, which would defeat the purpose of WFP.

However, in the paper, they justify the usage of top-k accuracy as it exposes a possible weakness in QUIC that would not be identifiable using accuracy alone. This is sound reasoning, as this suggests a possible fundamental weakness in QUIC that makes website fingerprinting easier to perform. Finding a higher top-k accuracy for QUIC

implies that the traffic is easier to fingerprint, as the other top-predictions are more likely to be correct than TCP traffic. Particularly considering they only used models with default parameters, if an attacker used tuned models or deep learning, would this weakness that appears minor become more severe?

4.4.2 Figures

The figures used in the WFP on Early QUIC paper are not always helpful and cast some doubt on the results of their experiments.

Firstly, they plot PCA and Feature Importance graphs for both protocols, these are both non-deterministic and could produce significantly different results each time they are used. Firstly, while feature importance is a valid metric, this varies from model to model, so it is not certain that across all models the same features would be important. Feature importance is still useful; however, where if the rankings of feature importance are the same across transport protocols, this shows that to the same ML model, they are similar problems which require similar solutions. Next, the PCA model does not make sense to plot; it does not show anything meaningful other than that as more packets are considered, the patterns become more distinct, which is a fairly obvious fact. In the original paper they used only 10 of the 92 websites to plot these figures, so not only would it not be representative of the protocol in general, but it also is not representative of the collected traffic in the paper itself.

Next, when comparing model accuracy across the range of 5 to 200 packets, for transfer features, we can see for GQUIC, that the accuracy of their models at 5 packets is 100%. This is highly unlikely, even when not considering results from outside the paper, when we can see for every other experiment, 5 packets produce low accuracy for all models. This raises doubts about the quality of results, for all experiments.

4.5 Summary

To summarise, in the Website Fingerprinting on Early QUIC Traffic paper, the aim of their data collection is to be realistic so that the conclusions made in the paper can be applied to open-world environments. However, there is a bit of conflict about how much realism is really sought, as the methods they chose do not prioritise realism. Because the focus is searching for weaknesses in QUIC, it could be argued that striving for realism is less important, as a fundamental flaw in the protocol should be visible in test environments as well as real ones.

They make several decisions in the paper that reduce the strength of their conclusions, such as the lack of cleaning and poorly defining the features used to fingerprint. This paper does not seek to rectify all these issues however, as the aim is to achieve the same results found in the paper, so straying too far from the original method would not make sense.

Chapter 5

Methodology

This chapter describes the methods and implementations of experiments performed in this paper. This builds on the previous chapter where the methods of the *Website Fingerprinting on Early QUIC Traffic* (WFP on Early QUIC) paper are critiqued, which will provide the reasoning behind many changes in design between papers. Furthermore, this chapter will reinforce any decisions made to differ from the original paper as well as discuss some of the alternatives that could have been made. This chapter is split into three sections, all focusing on the methodology of this paper:

- Data collection process
- Data handling and traffic tailoring
- Experiments performed

The methodology and implementation of this paper follows a similar approach to the WFP on Early QUIC paper. First sites are downloaded, stored on a webserver, and visited using a web crawler and a chosen protocol, capturing all traffic in each visit. The captured traffic is then tested to ensure the collected data is of good quality, and then the relevant parts of the capture are parsed from which features are then extracted to be used by a fingerprinting model. The models are trained using this feature data, where the website fingerprinting performance of captured HTTP2 and HTTP3 can be compared.

5.1 Data Collection

5.1.1 Downloading Sites

This paper uses the same list of university landing pages used in the WFP on Early QUIC paper and used the same website copier: *HTTrack*. This will clone the same origin resources of the list of websites, up to a chosen depth (i.e. traversing all files in the first and second levels of the root directory when depth=2). A copy depth of 2 is used like in paper as inter-domain fingerprinting is being measured, so greater depths would produce very large sites that are too big to host with current resources. However, if intra-domain fingerprinting was the aim, the focus would be a single website or CDN

copied with a large depth, so that different sections of the site can be distinguished during fingerprinting.

This paper uses the modern versions of the sites, rather than the 2021 versions that the WFP on Early QUIC paper would have used. Alternatively, the 2021 version of each site could have been downloaded using an internet archiving tool like the *Wayback Machine*. However, downloading sites from this archive is difficult and it is unclear whether the archiving process affects the site's structure and fingerprintability.

In total, this paper downloaded and used 94 sites, rather than the 92 used in their paper.

5.1.2 Single machine and Docker

Lacking access to multiple machines, the client-server setup from which to collect traffic traces was implemented on a single machine. To do this, two docker containers were created, one for the web server and another for the client. This will provide isolation on my machine so that they are more representative of a real client and server. Using Docker also allows for the reproducibility of the results in this paper as it is possible to clone my containers, and use the same environment used for this paper, independent of Operating System and other machine-specific details. This is a stark contrast to Zhan et al. who do not provide any code or containers, so their results are much more difficult to review.

The single machine that was available was Windows-based, so the containers were also necessary for running the crawler and using Linux command tools, which were a great help.

The whole crawl was completed offline, this was to prevent redirection from the copied sites to download other resources from other origins. This needs to be avoided as these will not come from the caddy web server so the protocol used may not match the set protocol, and the focus of this project is on the transport protocols, so it is important to isolate them.

5.1.3 Caddy Web Server

The webserver is implemented using the same software used in the WFP on Early QUIC paper: *Caddy*. A newer version of Caddy is used as this supports a more modern version of QUIC, where some of the initial quirks from early development should have been ironed out. This is because this project is concerned with whether the results found in the WFP on Early QUIC hold today with modern versions of these protocols, as these latest protocols are used in current systems. In the Early QUIC paper, they are using a version of Caddy where QUIC was still experimental, and we can see this from their using multiple versions of the software to alter the QUIC support and because of the time of publishing. Specifically, caddy server version 2.7.5 is used which allows the specification of supported protocols, allowing easy switching between HTTP2 support and HTTP3 support.

Other web-serving software is available, but Caddy was chosen to use the same one as in the paper to try to stay closer to the original implementation.

5.1.4 Selenium Web Crawler

To crawl the websites stored on my server, in this paper, a *selenium* web crawler inspired by Juarez's crawler [47] is containerised. The browser used by the crawler was Chrome, the same as in the WFP on Early Quic paper, and traffic was collected using *dumpcap*. This crawler had QUIC enabled and forced on when collecting HTTP3 data and avoided page caching by setting the disk cache and media cache sizes of the browser to a single byte. Also, a Chrome profile was initialised, which recorded my caddy server as a certificate authority, as HTTP3 through Chrome is strict when it comes to certificates, and if there is a suspicious one, it will fall back on using HTTP2 (or not load at all!).

The crawler is supplied with the domains of 100 target websites (although 6 of these resolve to nothing as they failed to download, like the WFP on Early QUIC paper which used 92 for the same reason). The reason these failed to download is most likely due to anti-bot measures on each of the sites, preventing access to the automated web copier.

The crawling process for a single site is:

1. Start a new instance of Chrome with Selenium, with all relevant options (i.e QUIC enabled) and a preset profile.
2. Wait 2 seconds.
3. Initialise a sniffer, with filters set to capture both TCP (used in HTTP2) and UDP (used in HTTP3) traffic.
4. Access the chosen site and request all resources (all resources from different origins will result in 404 errors as the crawler is offline, so only resources from the caddy server will be retrieved).
5. Wait for the landing page to render completely and return a success state to Selenium.
6. Save a screenshot of the loaded website for debugging purposes.
7. Then wait for an additional 5 seconds before closing the sniffer process, followed by the Chrome process.
8. Repeat for a total of 100 captured traces.

This is repeated for all sites and each protocol. As well as this, not every visit is a success, so each site is visited more than 100 times, to collect at least 100 *successful* visits. Especially, when crawling using HTTP3, the crawler will often capture a small amount of traffic, indicating the full site has not been loaded. For further information on how this was dealt with see Checking Collected Traffic Section 5.2.1.

An alternative here is to collect empty packets, as is common in other papers. The WFP on Early QUIC does not mention doing this, and it should not make a difference to the results as only the packet header is used in website fingerprinting. Here payloads were included, to follow the methods of the paper, and because this is a quality-of-life design choice rather than one that will affect the results.

5.2 Data Handling

5.2.1 Checking Collected Traffic

As there is an inherent randomness associated with network communication, the traffic for the same site might not look the same over multiple connections. When crawling, commonly, a site will not fully load and so the captured traffic file will be very small, compared to what you would expect of a site of that size. To get rid of these small but not empty files, as well as any files which did not seem to match the general size of other visits to that domain, outliers were removed from the captured traffic. This was done by checking if the size of the file was greater than Q3 + 1.5*IQR or less than Q1 – 1.5*IQR. It is reasonable to do this as if it results in greater accuracy, then an attacker would likely do it too.

This removed the bulk of outliers, but when plotting boxplots, there were still suspicious results for some domains, especially for collected HTTP3 traffic (Fig. A.3). These results contained many close-to-empty captures which distorted the outlier detection. Following this, the collected screenshots were checked for unsuccessfully loaded pages, deleting those which had not loaded successfully. To ensure there were 100 successful captures for each domain, some domains had to be recrawled to fill in the gaps of removed outlier, or invalid pages.

The distribution of the cleaned collected traffic can be seen in the Figures A.1 and A.2 in the Appendix.

5.2.2 Parsing Traffic

TShark was used to parse PCAP (packet capture) files into CSV files, with the relevant fields for extracting features (and some extras for debugging). There is a separate process for parsing HTTP2 and HTTP3 traffic, as there are some differences between the fields that require different parsing processes (i.e. TCP-specific fields). But the important fields that are used in feature extraction, such as packet size, are calculated using fields that are shared between both types of traffic, in case there are slight differences in how each is calculated (i.e. how much metadata does it include, if any).

The traffic parser is also containerised to allow for reproducibility of results and portability of the parser across machines.

5.2.3 Checking Parsed Results

Parsed results were also examined for suspicious or off-looking traces that may have seeped through the cracks of the earlier tests. In particular, the ratios of positive to negative packets were useful in finding a small number of invalid captures. These tests were not as rigorous as earlier as by this point the traffic was largely clean.

5.2.4 Extracting Features

Firstly, for all features an early traffic limiter was implemented, to prevent any packets being used in calculating features above a given threshold. The "early" traffic scenario in the WFP on Early QUIC paper is defined as the first 40 packets, so in this instance, only the first 40 packets were used to calculate **all** features. When extracting features, packets are filtered to only ones going to or from the port of the Caddy server. This is so that other traffic included in the captures is ignored, and only traffic from loading the site is used in extracting features.

Also, the handshake is not removed from the collected traffic. Given that the handshake is part of the transport protocol, excluding it from the data does not make sense in the early traffic scenario. As the handshake is the same across each domain, it should not affect fingerprintability. It will have an effect between protocols however as a shorter handshake for QUIC should mean there are more packets with useful information in the first n.

The same features used in the WFP on Early QUIC paper are used here as the features they used were sufficient in finding a difference between the protocols, which the paper aims to confirm so there is no reason to use extras. These features are split into two categories: simple and transfer features. The simple feature is an 8-dimensional feature, which records size and direction, with four size categories, for both positive (towards the webserver) and negative (from the webserver) directions. The four size categories are tiny (packet size $<$ 80 Bytes), small (80 $<=$ size $<$ 160 Bytes), medium (160 $<=$ packet size $<$ 1280) and large (1280 $<=$ packet size).

Next, the transfer features described in the paper were used, although the definitions were vague in parts and generally unclear, requiring some assumptions to be made. An example of this is that they never mention whether they limit the calculation of these features to the first number of packets n (called k in the paper). The definitions for the features used in this paper are as follows:

- **Unique Packet Size** (1460-dimension vector, where 1460 is the difference between the largest and smallest packet). This feature records the unique packet sizes present in each capture, for example if there is a packet of size 1234 bytes, the 1234$^{\text{th}}$ dimension of this feature is set to 1, while if there is no packet of size 999 bytes then this will be set to a 0. Specifically, $(1\ if\ |\{p \mid |p| = l\}_{p \in T_{[0:n]}}| > 0\ otherwise\ 0)_{l \in [0:1460]}$ where T is the ordered traffic (so p is a packet), and n is the number of packets to consider.
- **Packet Size Count** (1460-dimension vector). This feature records the frequency of different packets sizes in the captured traffic. For example, if 10 packets are of size 1234 bytes, then the 1234$^{\text{th}}$ element of this vector will be set to 10. Specifically, $(|\{p \mid |p| = l\}_{p \in T_{[0:n]}}|)_{l \in [0:1460]}$ where T is ordered traffic, and n is the number of packets to consider, and p is a packet.
- **Packet Order** (n-dimension vector). This feature vector records the packet lengths in order they appear in the traffic capture. Specifically, $(|p|)_{p \in T_{[0:n]}}$ where T is ordered traffic, and n is the number of packets to consider.

- **Interarrival Time** (n-dimension vector). This feature records the difference between the arrival times of adjacent packets in the captured traffic. Specifically, $(time(p_{i+1}) - time(p_i))_{p_i \in T_{[0:n]}}$ where $time(p)$ returns the arrival time of a packet.
- **Negative Packets** (1-dimensional). This feature records the number of packets in the negative direction (from the server to the client). Specifically, $\sum_{p \in T_{[0:n]}} negative(p)$ where $negative(p) = \begin{cases} 1 & if\ src = server \\ 0 & if\ src = client \end{cases}$ and src is the source of a packet.
- **'Cumulative Size' aka Total Size** (1-dimensional). This feature records the sum of the packet sizes. Specifically, $\sum_{p \in T_{[0:n]}} |p|$.
- **'Cumulative Size with Direction' aka Total Size with Direction** (1-dimensional). This feature records the sum of packet sizes, except that now values in a negative direction have a negative size. Specifically, $\sum_{p \in T_{[0:n]}} direction(p) \cdot |p|$ where $direction(p) = \begin{cases} -1 & if\ src = server \\ 1 & if\ src = client \end{cases}$.
- **Burst Features** (3-dimensional). Records the number of bursts, largest burst length, and mean burst length. For more information regarding what a burst is see section VI. C. of Dyer et al., Peek-a-boo [27] which provides a good description.
- **Total Transmission Time** (1-dimensional). This feature records the total transmission time of the traffic calculated by the sum of the interarrival times. Specifically, $\sum_{p_i \in T_{[0:n]}} (time(p_{i+1}) - time(p_i))$

The initial implementation of feature extraction was inefficient as each feature was calculated separately, and all parsed files were processed in sequential order. This led to the implementation of a faster feature extractor which calculates all features at the same time and is also multithreaded to extract features from different files simultaneously. This was necessary as features must be extracted for each different level of n packets, and the previous implementation would have taken an unreasonable amount of time for the range of n=($5 \rightarrow 200$), for each protocol.

5.3 Experiments

5.3.1 Website Fingerprinting Early Traffic

With these extracted features several of the experiments from the WFP on Early QUIC paper were recreated. In their paper they also explore vulnerability under a flawed network, but the most significant results from their paper are the ones showing a weakness in QUIC to WFP, so there was a focus on producing these ones.

The most interesting results from their paper are testing the first n packets, using different top-n accuracies, and a Random Forest model (with default parameters) to classify the traffic traces. They also perform this using only the simple features described earlier, as well they don't record their testing and training splits but do mention earlier in the paper using 10-fold cross validation, therefore there is an assumption here that they are still using cross validation even though not comparing different models.

5.3.2 Protocol Comparison

The experiment from the paper where for both simple and transfer features, and for each protocol, I record the accuracy of five different models (Random Forest, Extra Trees, Naïve Bayes, KNN, and SVM) as the value of n (number of packets) increases from 5 to 200, in steps of 5. For this I used 10-fold cross validation, like they used in the paper.

Another experiment from the paper is recreated where for both simple and transfer feature, and for each protocol, the accuracy of five different models is recorded as the value of n (number of packets) increases from 5 to 200, in steps of 5. For this 10-fold cross validation is used, as is done in the paper. This is where a dataset is shuffled then split into 10 equally size folds, where for each fold, that fold is used as a testing set, and the other folds combined are used as the training set. Accuracy is calculated for each split, and then the mean accuracy of all folds is taken. This is also helpful as it allows us to observe the standard deviation of all folds, to understand the consistency of the performance for each model.

5.3.3 Feature Importance

The feature importance experiments from the original paper are also recreated. In short, these were to examine the individual features used in fingerprinting for each different transport protocol as the number of packets considered increases. This follows the same method as the previous experiment, except instead of accuracy it outputs feature importance at each step. Specifically, Gini Importance is calculated from the sklearn Random Forest model. As mentioned in the methodology critique chapter, plotting these does not make total sense, but the ranking of the most important features is useful to compare across the papers.

5.3.4 Tuned Website Fingerprinting

In the WFP on Early Quic paper they perform all experiments using the models' default hyperparameters. It could be assumed that in a real attempt to website fingerprint, the attacker would want to make their website fingerprinter as effective as possible, so would likely try tuning the models they use. In addition, by tuning the hyperparameters, we might further expose the weakness of QUIC discovered in the paper. Furthermore, it should give us insight into how similar the problems are, where if there is a significant difference in chosen parameters, we can presume that the models are exploiting different aspects of the protocols, so they do influence fingerprinting. For these reasons, another experiment was run where the five models used in the paper were tuned using *exhaustive grid search*to find the combination that worked the best. Grid search is a straightforward algorithm where a list of values is supplied for each hyperparameter, and each combination of these is tested to find which values produce the most effective model.

As hyperparameter tuning takes a long time and tuning needs to be completed for HTTP2 and HTTP3, it was decided to only tune the models for simple features extracted from the first 40 and 200 packets, using 5-fold cross-validation. The hyperparameters for the first 200 packets are compared, while the tuned models for the first 40 packets are used

to recreate the early top-k experiment, this time without the fingerprint, like in the WFP on Early QUIC paper.

Note that for each model, there are different hyperparameters to tune, whereas the tree-based models have a larger number of hyperparameters. To see the exact hyperparameters and values used in the grid search, see the Grid Search Values section in the Appendix A.4.

5.3.5 AutoML Website Fingerprinting

Following on from the tuning, *AutoML* is used to see what the best-performing models were on the early website fingerprinting problem. This is similar to the tuning experiments, but using autogluon, different machine learning approaches will be automatically tried and combined to produce a high-quality model. Note that this also includes the tuning of different models. The thought process is that there would be some difference in the top-performing models for HTTP2 and HTTP3, which could reveal something about the structure or nature of website fingerprinting for these different types of traffic.

This was first attempted using *autosklearn* as it is a drop-in replacement for sklearn, so it would be consistent with the other experiments. However, this package has not been maintained, so it was decided to use AutoGluon which has proven results in a wfp setting [28]. Then, the leaderboard was used to visualise the best-performing models along with the outputted predictor. This will use the simple features for 200 packets, as it was when just the simple features were used that original differences were found in the protocols, so this is sufficient.

Ha et al. [28] focus on performance in their paper, while here, the aim is to use AutoML to understand if website fingerprinting each protocol is a unique problem requiring different strategies. If the results are significantly different, this should reveal that the transport protocol does have an effect on fingerprintability.

Chapter 6

Results & Discussion

This chapter will provide the results of the experiments noted in the Methodology chapter, discuss what these results show and why they differ from the original results found in the *Website Fingerprinting on Early QUIC Traffic* (WFP on Early QUIC) paper. This chapter first explores the recreated experiments from the WFP on Early QUIC paper and then moves on to this paper's original extension experiments. It will then discuss reasons that the results differ between papers, as well as show where the results are unexpected and provide possible explanations for them. This is different to what was previously discussed in the methodology, where decisions that could have an effect were identified; here, the aim is to be more specific, referring to the actual results and proposing likely culprits of certain effects. This will include some of those previous points, which are now contextualised.

6.1 Recreated Experiments

6.1.1 WFP on Early Traffic using Top-k

In Table 6.1 and Table 6.2, HTTP3 appears to perform slightly better in resisting fingerprinting, but overall the protocols perform relatively similarly. There is no significant difference between the performance of the protocols in this scenario, as was found in the WFP on Early QUIC paper. In these results, no accuracy is anywhere near sufficient to breach a user's privacy, even when using top-k.

6.1.2 WFP Resistance Comparison

In the simple feature figures (Fig. 6.1a and Fig. 6.1b), we can see that the trends across both protocols are very similar. Again, no model reaches sufficient accuracy to pose any real-world threat. However, we see that the same models that work well on HTTP2 also work well on HTTP3. Comparing the Random Forest results for both protocols in Fig. 6.1e, it can be seen that the early performance for HTTP3 is marginally better than HTTP2 but that as more packets are considered, it becomes marginally less resistant to fingerprinting.

Table 6.1: WFP HTTP2 in the Early Traffic Scenario using Simple Features and Random Forest

	Mean Top-K Accuracy Across Folds (%)				
n	**top-1**	**top-2**	**top-3**	**top-4**	**top-5**
5	5.277±0.63	8.347±1.10	10.789±0.99	13.294±1.03	15.309±1.15
10	6.439±0.54	10.490±0.59	13.603±0.94	16.493±1.01	16.493±0.74
15	7.154±0.98	11.055±1.15	14.200±1.20	16.962±1.12	19.307±1.39
20	8.038±1.05	11.962±1.13	14.925±1.22	17.271±1.02	20.043±1.35
25	8.348±0.98	11.930±0.98	15.362±0.92	18.006±1.53	20.917±1.44
30	9.030±0.80	12.569±0.68	15.256±0.91	18.081±0.65	20.554±1.28
35	9.851±0.92	13.571±1.11	16.375±1.26	18.742±1.33	21.365±1.11
40	10.053±1.03	14.414±1.06	17.644±0.95	20.480±1.04	22.889±0.76

Table 6.2: WFP HTTP3 in the Early Traffic Scenario using Simple Features and Random Forest

	Mean Top-K Accuracy Across Folds (%)				
n	**top-1**	**top-2**	**top-3**	**top-4**	**top-5**
5	3.092±0.32	5.389±0.47	7.750±0.80	10.090±0.73	12.483±0.52
10	4.097±0.51	6.946±0.86	9.370±1.02	11.710±1.01	13.827±1.02
15	4.436±0.37	7.295±0.72	10.101±0.68	12.567±0.74	14.611±0.86
20	4.605±0.44	7.528±0.81	10.323±0.85	13.023±1.34	16.389±1.35
25	4.669±0.60	7.581±0.81	10.651±0.85	13.097±1.33	15.839±1.35
30	5.442±0.64	8.713±0.93	11.805±1.06	14.759±1.06	17.109±1.04
35	6.003±0.58	9.317±0.93	12.705±0.88	15.818±1.09	18.348±1.19
40	7.316±0.98	10.799±0.92	14.113±1.56	17.459±1.41	20.095±1.45

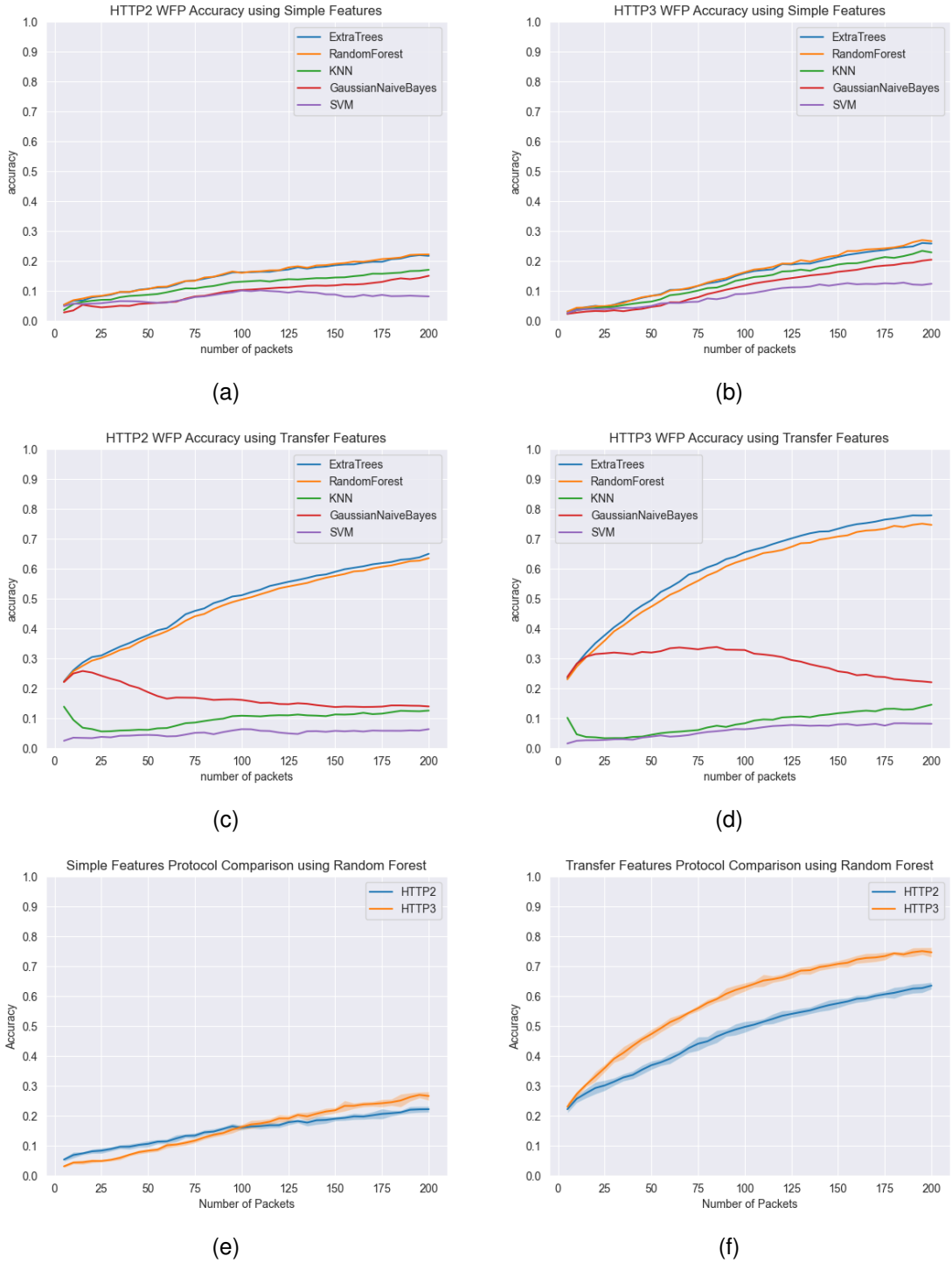

Figure 6.1: WFP performance comparison across protocols

In the transfer features figures (Fig. 6.1c and Fig. 6.1d), we can compare the performance pattern of all models using transfer features. These figures again show a similar pattern of model performance across protocols, where the trend for every model is very similar to its other protocol counterpart. The possible exception here is Naïve Bayes, which seems to perform better on the midrange of HTTP3 than on the midrange of HTTP2; however, it still performs very similarly across protocols for earlier and later traffic.

Next, looking at the direct comparison of Random Forest using transfer features in Fig. 6.1f, there is a greater accuracy when fingerprinting HTTP3. For the early traffic, the protocols perform the same but begin to diverge until 75 packets, where a constant difference of about 10% remains between the protocols. It is difficult to compare patterns here to the original paper's experiment as their accuracies are so inflated that they reach 100% midway through the experiment, preventing any meaningful comparison between protocols. Note that in Fig. 6.1e and Fig. 6.1f, the standard deviations are visible and relatively small, meaning this is statistically significant.

6.1.3 Feature Importance Comparison

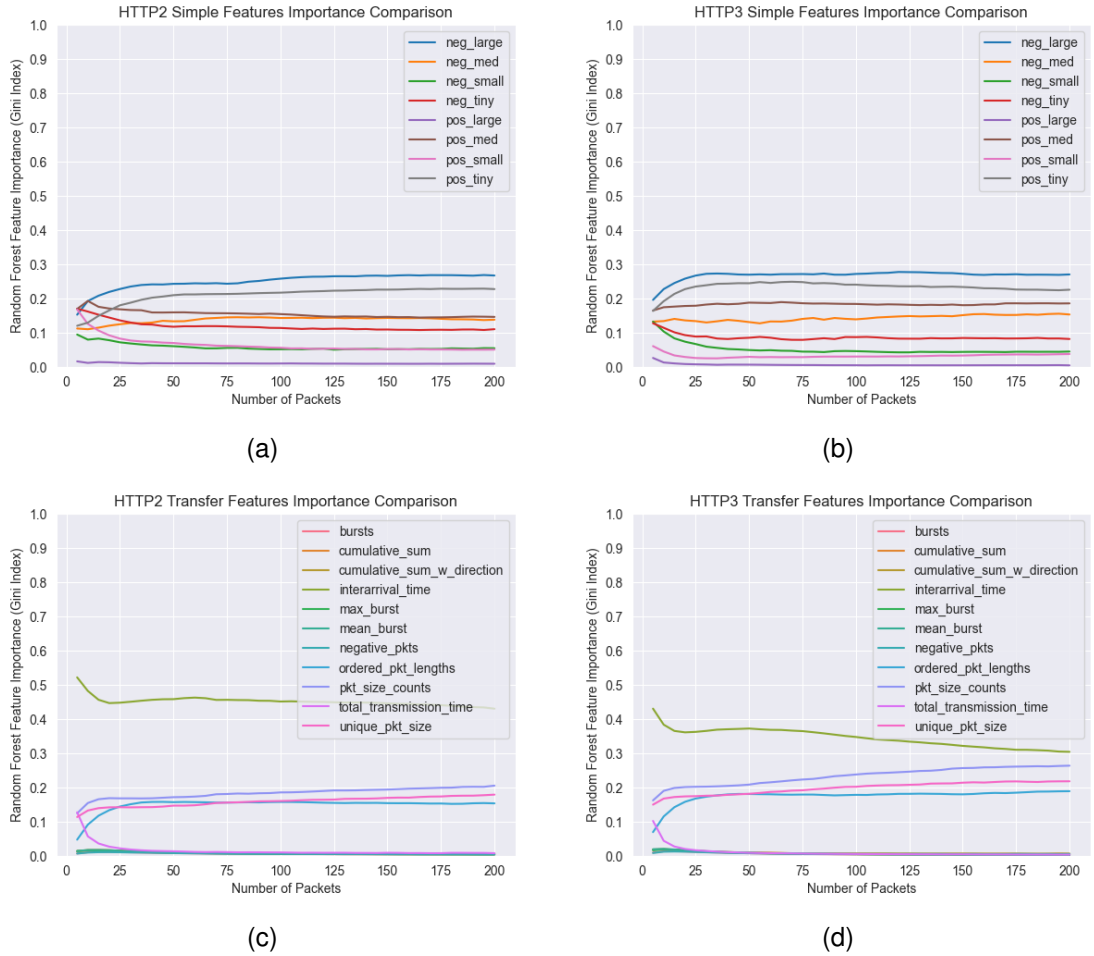

Figure 6.2: Random Forest WFP feature importance comparison across protocols

As mentioned in the Methodology Critique Chapter 4, plotting the feature importances can be slightly misleading, as this is a different metric depending on the model being used. Therefore, only the feature importance of Random Forest is compared. Also, rather than focusing on the exact values of feature importance, this paper is more concerned with the ranking of feature importances, i.e. what feature is the most important rather than the exact values of feature importance.

Figure 6.2 shows that packet size count, unique packet size, interarrival time and packet order are the most significant. When comparing the feature importances we see for

simple features, the ranking of features is the same except at the beginning of the traffic, where HTTP2 features a different feature ranking. For the transfer feature importances, we see the same ranking of features across both protocols.

6.2 Extension Experiments

6.2.1 Tuned Hyperparameters

The hyperparameters found through grid search for 200 packets are as follows:

6.2.1.1 HTTP2

- **Extra Trees** = { 'max_depth': 20, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 200}
- **Random Forest** = {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 200}
- **K-Nearest Neighbours** = {'n_neighbours':17}
- **Gaussian Naïve Bayes** = {'var_smoothing'=3.51e-5}
- **Support Vector Classifier** = {'C', 1000, 'gamma':0.0001}

6.2.1.2 HTTP3

- **Extra Trees** = {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 500}
- **Random Forest** = {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 300}
- **K-Nearest Neighbours** = {'n_neighbors': 13}
- **Gaussian Naïve Bayes** = {'var_smoothing': 0.0002310129700083158}
- **Support Vector Classifier** = {'C': 100, 'gamma': 0.001}

6.2.2 Tuned WFP on Early Traffic using Top-k

In Table 6.3 and Table 6.4, we can see that now, with the handshakes removed and using tuned random forest that these results do not match the previous ones. We see that HTTP3 is significantly harder to fingerprint than HTTP2. It should be noted that by using tuned models, the fingerprinter accuracy is expected to increase, further exposing the difference between the two protocols.

The effect of the handshake has been discussed previously, and it should be noted that the handshake will have the greatest effect on the early traffic, where it could take up a significant portion of the early packets, reducing fingerprint accuracy. The significant increase in accuracy on HTTP2 is due to a combination of tuning and

Table 6.3: Tuned WFP on HTTP2 in the Early Traffic Scenario using Simple Features and Random Forest

	Mean Top-K Accuracy Across Folds (%)				
n	**top-1**	**top-2**	**top-3**	**top-4**	**top-5**
5	7.260±0.87	11.599±1.32	15.448±1.54	18.475±1.82	21.716±1.85
10	11.471±1.31	16.994±1.51	20.640±1.64	23.625±1.66	26.205±1.59
15	12.260±1.26	17.143±1.27	21.226±1.34	23.923±1.54	27.154±1.33
20	12.047±0.70	17.409±0.96	21.130±1.24	24.051±1.60	26.823±1.70
25	12.601±0.71	17.687±1.08	21.269±1.22	24.755±1.10	27.260±1.30
30	12.697±0.67	18.742±1.29	23.188±0.90	26.439±0.89	29.270±0.81
35	13.422±0.48	19.200±0.87	23.252±0.82	26.503±1.06	29.232±1.13
40	14.158±0.86	20.394±1.17	24.638±0.99	27.569±1.19	30.586±1.09

Table 6.4: Tuned WFP on HTTP3 in the Early Traffic Scenario using Simple Features and Random Forest

	Mean Top-K Accuracy across Folds (%)				
n	**top-1**	**top-2**	**top-3**	**top-4**	**top-5**
5	3.618±0.50	6.201±0.89	8.443±0.70	10.854±0.61	13.224±0.79
10	4.632±0.66	7.482±0.87	9.788±0.97	12.669±1.16	15.829±1.07
15	4.686±0.66	7.322±0.78	9.926±0.78	12.381±0.93	15.103±0.84
20	5.187±0.47	8.123±0.74	11.036±0.80	13.619±1.14	16.640±1.23
25	5.081±0.46	8.485±0.63	11.303±0.88	13.801±0.80	16.661±1.04
30	5.646±0.65	9.233±0.92	12.232±1.10	15.412±1.20	17.995±1.21
35	5.550±0.63	9.126±0.87	12.360±1.08	15.807±0.76	18.519±0.83
40	6.799±1.09	10.919±1.17	14.302±1.28	17.073±1.49	21.058±1.80

removing handshakes, where removing the handshake creates a difference, and tuning then makes this difference more significant.

These results suggest that there is an effect from the transport protocol. However, it also provides evidence opposite the conclusion from WFP on Early QUIC, finding that HTTP3 is more resistant to fingerprinting. In terms of implications for website fingerprinting, if this is true, then this would mean that website fingerprinting would become harder in the future as more internet traffic uses HTTP3.

6.2.3 AutoML

Table 6.5 & Table 6.6 show the automl results for WFP HTTP2 and HTTP3, from these we can see that the final weighted L2 ensemble model's perform very similarly, this shows that there is little effect from the transport protocol in terms of WFP resistance. This experiment goes even further than the tuning experiment to tune the models used; it also combines models to produce a good predictive model. It is this model which gives us very similar accuracy, so this is strong evidence that the protocols perform the same. Note that the automl process is performed with handshakes removed and for 200 packets.

Table 6.5: AutoGluon Tabular Predictor Leaderboard for HTTP2 WFP

Model	Accuracy (%)
WeightedEnsemble_L2	30.8
CatBoost	28.1
XGBoost	26.2
NeuralNetTorch	26.2
NeuralNetFastAI	25.5
RandomForestEntropy	25.5
RandomForestGini	25.4
LightGBMXT	25.3
ExtraTreesGini	24.8
LightGBMLarge	24.6
ExtraTreesEntr	24.5
LightGBM	24.5

Table 6.6: AutoGluon Tabular Predictor Leaderboard for HTTP3 WFP

Model	Accuracy (%)
WeightedEnsemble_L2	31.0
CatBoost	27.9
ExtraTreesGini	27.7
RandomForestGini	27.5
RandomForestEntropy	27.3
NeuralNetFastAI	27.0
NeuralNetFastTorch	26.8
ExtraTreesEntropy	26.6
LightGBMXT	26.5
LightGBM	25.9
LightGBMLarge	25.5
XGBoost	24.3

Although looking at the results, it can be seen that there is a different ranking of models across protocols. This could imply that there is some difference between the protocols. However, the accuracy difference between models on the same protocol is fairly insignificant ($< 4\%$), so this is not particularly strong evidence.

6.3 Similarities Across Results

Firstly, in the website fingerprinting performance comparison Fig. 6.1, the rankings of models in this paper's results are the same as the ranking of models in the original paper. The tree-based models perform the best across both protocols in both papers, followed by KNN, Naive Bayes and SVM.

Next, regarding the feature importance Fig. 6.2 for transfer features, both papers observe that for early traffic, interarrival time is the most important feature, and for general traffic packet size counts and unique packet sizes are important features.

It is possible that the trend of resistance to fingerprinting using transfer features as packets increases matches across papers, but because the accuracy in the original paper is inflated, no patterns can be seen as the models all perform at close to or 100% accuracy as more packets are considered.

6.4 Difference Between Results

The most obvious is that the accuracies found in this paper are much lower than those found in the original paper. This is important as the values found for HTTP2 should be similar across papers. This protocol has been standardised for a long time, so no significant changes to the protocol have occurred between the publication of their paper and the writing of this one.

Arguably, the most important results from the original paper were those showing a weakness in early QUIC traffic. They find a significant difference between the performance of protocols in their experiments while Table 6.1 and Table 6.2 do not show this same weakness. Not only is there a much less significant difference between protocol fingerprinting resistance in this scenario, but our results also show much less variance, with the lower values of n having a slightly higher accuracy and the higher values of n having a much lower accuracy than the original paper.

Next, when comparing fingerprinting resistance for 5 to 200 packets, we not only see much higher resistance but also, the patterns between the papers do not match. Using just the simple features in our paper, we observe a more linear pattern than the original paper.

Finally, comparing feature importance results across papers; a notable difference can be seen for the simple feature importance. When comparing simple feature importances between protocols, the original paper sees a completely different ranking between IQUIC and TCP, while in this paper the results show that the ranking is very similar across protocols. For transfer features in this paper's results, interarrival time remains the most important feature for the duration, unlike the original paper, where it is quickly overtaken by packet size count and unique packet size. In this paper, these features are important but never become the most important.

6.5 Reasons for these Differences

6.5.1 Latest Versions

A large change between their methodology and this paper's is the usage of the latest versions of all software and packages at the time of writing (i.e. Chrome, Caddy). Firstly, using the latest version of Caddy would significantly affect the results, as the version of Caddy used in the original paper implemented QUIC as an experimental feature rather than as standard as it is in the latest version. This could mean that an immature version of QUIC was used that, although it had begun to be standardised by IETF, had not been finalised in Caddy. It is not just plausible but very likely that as HTTP3 was developed, the protocol became more standard in its function across different sites. This is deeply meaningful to website fingerprinting, as it means that site-specific information is not leaked due to the protocol acting in a certain manner for different sites. Using the modern standardised version of QUIC is the most likely reason why, in this paper's results, there is no significant difference in the vulnerability to website fingerprinting under top-a accuracy in the early scenario.

This applies to using the most modern version of Chrome too, as it would be expected that with HTTP3 becoming standard, the way Chrome handles it will have been refined over time, so that it is no longer experimental but standard. This would affect WFP vulnerability for the same reasons mentioned earlier.

However, this does not explain the large decrease in accuracy across all experiments, as we would expect the results for HTTP2 to stay roughly the same between the publication of their paper and the writing of this one.

6.5.2 Cleaning the Handshake

As discussed in the Methodology Chapter in Section 5.2.4, in this paper's implementation, it was chosen not to remove the handshake (for the recreated experiments), as this arguably does not make sense in the early scenario when you consider an attacker who can only access the first n packets of a trace. Intuitively, this should not have a large impact on this paper's results as the handshakes are relatively small (2-6 packets) compared to the number of packets used by the fingerprinting models. It should be noted, however, that because QUIC has a shorter handshake, this should make it marginally easier to fingerprint than TCP / HTTP2. This would also explain why the observed difference between protocols remains constant rather than continuing to diverge, as the effect of the handshake is expected to remain constant even as the number of packets increases.

To confirm this, when extracting features from parsed traffic, all packets involved in the handshake were removed, as is done in the paper, and the TCP handshake and IQUIC handshake in the *PCAP* files were examined. It was found that, particularly for HTTP2, it was difficult for some crawls to establish an initial connection to the server, resulting in a large preamble before any site-specific data was sent. This will significantly impact this paper's accuracy, which could be the reason for lower accuracies across all experiments. This could explain why HTTP2 seems more resistant to website fingerprinting for simple and transfer features as for larger numbers of packets Fig. 6.1eFig. 6.1f. To confirm this a random forest model was trained for 200 packets, with the handshakes removed for both protocols. It was found that for HTTP2, the accuracy increased to 67.6%, and the QUIC accuracy stayed relatively similar at 72.3%, a smaller difference than was observed initially, but this does not remove the whole difference between results, and this does not remove the generally lower accuracies found.

6.5.3 The Caddy QUIC Implementation

One significant concern when it comes to analysing these results is the effect of Caddy. Caddy was never designed to be used in the ways it is used in the WFP on Early QUIC paper, and that extends to this paper too. In particular, certain workarounds are necessary to get Caddy working locally with QUIC, which suggests there could be other inner workings that are less obvious and will affect the results of the experiments. For example, HTTP3 avoids reestablishing connections by remembering previous sessions (Section 2.3.2). This should be disabled by the crawler by removing the cache, but the Caddy server may still remember these connections and thus will not need any TLS negotiation. It is very difficult to know the exact effects of Caddy as this use case is far from its intended, so it is unlikely there has been much debugging for how it is used in this paper.

There is evidence of this in the collected traffic. When observing the capture files themselves to investigate the protocol handshakes, most QUIC *PCAP*s have a strange handshake structure, where they seem to lack an initial connection packet or maybe multiple initial packets. This could be related to server-side caching, but also could be that Caddy QUIC connections do not elegantly close, which could cause interference between visits. Ultimately, it is outside the scope of this paper to debug Caddy's

implementation of QUIC in this unordinary setup, so it is difficult to provide a specific aspect of Caddy that is affecting the results.

It is likely that the strange preamble (or, more accurately, the lack of preamble) to the Caddy QUIC traffic makes it more difficult to fingerprint compared to HTTP2, particularly when the handshakes are cleaned and the models are tuned. In the past, internal Caddy optimisations have caused QUIC trouble [4], so it is not unreasonable to assume that it may still cause strange behaviour, particularly in this paper's nonstandard use case. It is also possible that this was present in the original paper along with other issues related to the experimental implementation of QUIC in Caddy used in the original paper.

This is particularly relevant to the tuned WFP experiments that show HTTP3 being more resistant to website fingerprinting than HTTP3. This would be less obvious in the initial experiments that included the handshake, which will also make HTTP2 harder to fingerprint.

6.5.4 Traffic Tailoring

As discussed in the methodology critique Section 4.3.2, while they mention traffic tailoring, they do not go into depth about their process for removing invalid and error pages. This paper's implementation thoroughly cleaned the collected data, removing outliers and invalid pages. During the cleaning process, it was also noted that invalid pages were difficult to spot, as they still produced some traffic, and without using screenshots and more rigorous approaches, they would have been unnoticed in this paper's data. It is possible that they may include some erroneous traces in their traffic tailoring and cleaning process. This may affect the results, but it is difficult to say whether it will result in higher or lower accuracy.

The HTTP2 and HTTP3 datasets may have effected differently by the cleaning process, creating some differences in fingerprintability. Observing the figures in the Methodology chapter, we can see that the bar plots for the same domain are not always the same across protocols; for example, Sorbonne-Universite has a shorter box for HTTP3 (A.2c) compared to HTTP2 (A.1c), which is longer. This would affect accuracy as the pattern for this shorter box is more distinctive than the pattern for the longer box, so should be easier to fingerprint. Observing the collections, we can see that HTTP3 has more of these short boxes, which suggests that the dataset is easier to fingerprint based on distribution. These patterns are likely not a result of the protocol but the cleaning process, which could explain why the results do not match the expectation of performing similarly.

It is not impossible, however, that this different distribution is because of the transport protocol; however, in combination with strange patterns in initial QUIC traffic, a likely explanation is that the QUIC captures are missing initial packets, resulting in generally smaller *PCAP* files for QUIC, effecting the distribution and the cleaning process.

6.5.5 Discussion of Extension Experiments

From the tuning results it can be seen that many of the hyperparameters found for HTTP2 and HTTP3 are similar, although not the same. When tuning using an exhaustive grid search, a collection of values to try for each hyperparameter needs to be provided. For most of this paper's results, the values found for HTTP2 and HTTP3 are adjacent in the provided lists of values. There are exceptions to this: firstly, the number of estimators for HTTP3 seemed to be larger for both tree-based models. Also, the values for var_smoothing for Gaussian Naïve Bayes appear significantly different.

This similarity between results across protocols shows us that not only are the best-performing models the same across both protocols but also that, generally, the same selection of hyperparameters does, too. This again reinforces the idea that the effect of the transport protocol is small.

However, the two protocols still perform differently when using tuned models and removing handshakes. This could imply that there is indeed an effect from the transport protocol, or this could also be due to the Caddy QUIC implementation, which is performing some optimisations that affect the repeated local connections.

6.6 Summary

To summarise, the recreated experiments show that website fingerprinting HTTP2 and HTTP3 have similar patterns. This proves they are similar problems and provides evidence that the transport layer has little effect on website fingerprinting. This paper finds that the weakness discovered in the WFP on Early QUIC paper is absent in the results. Also, this paper's results show weak accuracies, which would never be sufficient to perform an attack. Finally, while this paper shows a difference in fingerprintability between protocols when more packets are used, this is mostly due to the inclusion of handshakes and the rigorous cleaning of the data. When accounting for the handshake, this difference is smaller but still noticeable, so this could suggest that HTTP3 is marginally weaker in fingerprinting, but this is not supported by the tuned WFP experiments. However, previous research has found that in these normal scenarios, both protocols perform about the same [49], so this is more likely to result from reasons independent of the protocols, most likely something in the methodology, such as the cleaning process, or something to do with the way Caddy serves QUIC traffic.

The extension experiments provide further evidence that the problem of website fingerprinting each protocol is very similar. However, it also provides some evidence that QUIC is harder to fingerprint, this casts some doubt on the methodology, as this is unexpected for the same reason that QUIC performing worse is unexpected. This could be for several reasons, but the most likely would be due to the Caddy server performing internal optimisations (which have caused issues for QUIC in the past) that will affect the initial packets of a trace in this paper's setup.

Chapter 7

Conclusions

7.1 Effect of the Transport Protocol

Website fingerprinting is a privacy breaching attack designed initially for use on Tor but has also been extended to non-Tor traffic, where both source and destination IP addresses are still associated with each packet. The widespread introduction of a new transport protocol – QUIC – on the internet within the HTTP/3 stack prompts research into whether this new transport protocol influences website fingerprinting. Intuitively, the transport protocol should have little to no effect, yet Zhan et al. discovered an apparent weakness in the QUIC protocol when website fingerprinting on early traffic using top-n metrics. This prompted further investigation in this paper, which performed similar experiments, aiming to discover the same weakness and further experiments to work towards an explanation of where the weakness is in QUIC or what feature of QUIC makes it more vulnerable to fingerprinting. The results of this paper dispute those of the original paper, finding that using the standard version of QUIC in HTTP3 and TCP in HTTP2, this weakness no longer exists. This paper demonstrates that the exposed weakness has been resolved through the standardisation of the protocol.

Furthermore, this paper extends Zhan et al.'s paper by individually tuning each model for HTTP2 and HTTP3 separately. This experiment provides further evidence that what makes each model effective at fingerprinting is independent of the transport protocol used. This reinforces that the transport protocol has a negligible effect on website fingerprinting.

7.2 Limitations of Study

The results show around a 10% difference in accuracy when website fingerprinting both protocols (including the protocol handshakes) in the normal scenario using Random Forest, this difference is also statistically significant as the standard deviations are relatively small. This defies the hypothesis and previous studies [49]. This paper explained that the difference is due to the traffic tailoring and cleaning process. However, this undermines the hypothesis that the protocols perform relatively equally as the

results do not clearly show as such, limiting the conclusion's strength. Moreover, when removing the handshakes from the traffic and tuning the models, the results show that HTTP3 is more resistant to website fingerprinting than HTTP2. This does not back up the hypothesis, and while an explanation is given as to why this is not the result of the transport protocol, this result still provides evidence against the hypothesis and reduces the strength of the conclusions.

In the discussion of the results, several reasons are given as to why the results do not match the hypothesis, unrelated to the transport protocols themselves. For the most part, these are speculative, so would need further research to confirm or refute.

7.3 Future Work

A possible extension to this paper would be to run similar experiments with intra-domain fingerprinting. This paper chooses to analyse inter-domain fingerprinting as this was used by Zhan et al., yet in this case, it would make more sense for Zhan et al. to research intra-domain fingerprinting, as this is what a real attacker would use. However, the results would likely match those found in this paper as the transport protocol should have little effect whether inter or intra-domain fingerprinting. Another extension along similar lines would be to perform the experiments in the open world like Smith et al. [49]. This would also avoid the use of Caddy, which is suspected to be interfering with the results and would ensure the conclusions apply to more realistic and general scenarios.

Another useful extension would be to recreate the Zhan et al. experiments using the exact system and versions used in the paper. This was not possible in this paper due to inaccessibility to multiple machines and a focus on finding a present weakness rather than a past one. Clearly, this weakness has been resolved inadvertently between the publication of the Zhan et al. paper and writing this one. Therefore, if the paper was recreated exactly as described, it could be iterated upon to find exactly when this problem was solved, and this could provide a basis to discover why it seemed weaker at the time. This is useful as the problem may not have been related to QUIC itself but instead to the system used, in which case it is important to learn what element of the system compromised the protocol for website fingerprinting attacks.

In this paper, an assumption made is that a victim is visiting one website at a time and completes their visit before moving on to another domain. This paper chooses to do this as this assumption was made in the Zhan et al. paper. However, Cui et al. [24] bring up a valid point that this is not necessarily realistic and that users often visit one website after another continuously or may visit another website while loading the current one. This prompts future work to ensure that the conclusion that the transport protocol has a negligible effect is robust by comparing different versions of HTTP under these circumstances.

Finally, this paper could be extended to analyse entropy and information leakage from each protocol. Accuracy does not tell the whole story when it comes to website fingerprintability (Section 2.2.3) so it could be useful to verify further there is no weakness in the QUIC protocol using entropy metrics.

Bibliography

[1] *Cloudflare: Announcing Encrypted Client Hello*. https://blog.cloudflare.com/announcing-encrypted-client-hello/ [Accessed: 11/03/24].

[2] *Cloudflare: What is a CDN?* https://www.cloudflare.com/learning/cdn/what-is-a-cdn/ [Accessed: 11/03/24].

[3] *Even faster connection establishment with quic 0 rtt resumption*. https://blog.cloudflare.com/even-faster-connection-establishment-with-quic-0-rtt-resumption/ [Accessed: 11/03/24].

[4] *Fixing Caddy's HTTP3 blog post*. https://moebuta.org/posts/fixing-caddys-http3/ [Accessed: 2/04/24].

[5] *IETF: Security & Privacy*. https://www.ietf.org/topics/security/ [Accessed: 11/03/24].

[6] *QUIC 0-RTT explained*. https://http3-explained.haxx.se/en/quic/quic-0rtt [Accessed: 11/03/24].

[7] *QUIC Working Group GitHub: Implementations*. https://github.com/quicwg/base-drafts/wiki/Implementations [Accessed: 11/03/24].

[8] *Robin Marx: Head-Of-Line Blocking in QUIC and HTTP3: the details*. https://calendar.perfplanet.com/2020/head-of-line-blocking-in-quic-and-http-3-the-details [Accessed: 2/04/24].

[9] *Roskind, QUIC: Design Document and Specification Rationale*. https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/edit [Accessed: 11/03/24].

[10] *Surfshark VPN User Statistics*. https://surfshark.com/blog/vpn-users [Accessed: 11/03/24].

[11] *TCPM updates on TCP fast open deployment IETF 101 meeting slides*. https://datatracker.ietf.org/doc/slides-101-tcpm-updates-on-tcp-fast-open-deployments/01/ [Accessed: 11/03/24].

[12] *Tor Country Relay Statistics*. https://metrics.torproject.org/userstats-relay-country.html [Accessed: 11/03/24].

[13] *What Is My IP Address?* https://whatismyipaddress.com/ [Accessed: 11/03/24].

[14] S. Ansari, S.G. Rajeev, and H.S. Chandrashekar. Packet sniffing: A brief introduction. *Potentials, IEEE*, 21:17 – 19, 02 2003.

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

[16] Mike Bishop. HTTP/3. RFC 9114, June 2022.

[17] Xiang Cai, Rishab Nithyanand, and Rob Johnson. Cs-buflo: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, page 121–130, New York, NY, USA, 2014. Association for Computing Machinery.

[18] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 227–238, New York, NY, USA, 2014. Association for Computing Machinery.

[19] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 605–616, New York, NY, USA, 2012. Association for Computing Machinery.

[20] Giovanni Cherubin, Jamie Hayes, and Marc Juarez. Website fingerprinting defenses at the application layer. *Proceedings on Privacy Enhancing Technologies*, 2017(2):186–203, 2017.

[21] Jonathan Corbet. Quic as a solution to protocol ossification. 2018. https://lwn.net/Articles/745590/ [Accessed: 11/03/24].

[22] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Traffic classification through simple statistical fingerprinting. *SIGCOMM Comput. Commun. Rev.*, 37(1):5–16, jan 2007.

[23] Weiqi Cui, Tao Chen, and Eric Chan-Tin. More realistic website fingerprinting using deep learning. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 333–343, 2020.

[24] Weiqi Cui, Tao Chen, Christian Fields, Julianna Chen, Anthony Sierra, and Eric Chan-Tin. Revisiting assumptions for website fingerprinting attacks. In *Proceedings of the 2019 ACM asia conference on computer and communications security*, pages 328–339, 2019.

[25] Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability

and the network effect. In *Workshop on the Economics of Information Security*, 2006.

[26] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, 2004.

[27] 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.

[28] Joonseo Ha and Heejun Roh. Quic website fingerprinting based on automated machine learning. *ICT Express*, 2023.

[29] Jamie Hayes and George Danezis. k-fingerprinting: a robust scalable website fingerprinting technique, 2016.

[30] Gaofeng He, Ming Yang, Xiaodan Gu, Junzhou Luo, and Yuanyuan Ma. A novel active website fingerprinting attack against tor anonymous system. In *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 112–117, 2014.

[31] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. Census and survey of the visible internet. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, IMC '08, page 169–182, New York, NY, USA, 2008. Association for Computing Machinery.

[32] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, page 31–42, New York, NY, USA, 2009. Association for Computing Machinery.

[33] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021.

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

[35] Khaleque Md Aashiq Kamal and Sultan Almuhammadi. Vulnerability of virtual private networks to web fingerprinting attack. In Kevin Daimi, Hamid R. Arabnia, Leonidas Deligiannidis, Min-Shiang Hwang, and Fernando G. Tinetti, editors, *Advances in Security, Networks, and Internet of Things*, pages 147–165, Cham, 2021. Springer International Publishing.

[36] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. The quic transport protocol: Design and internet-scale deployment. In *Proceedings*

of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17, page 183–196, New York, NY, USA, 2017. Association for Computing Machinery.

[37] Shuai Li, Huajun Guo, and Nicholas Hopper. Measuring information leakage in website fingerprinting attacks and defenses, 2019.

[38] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 255–263, New York, NY, USA, 2006. Association for Computing Machinery.

[39] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *Network and Distributed System Security Symposium*, 2016.

[40] R.K. Pathria and Paul D. Beale. 3 - the canonical ensemble. In R.K. Pathria and Paul D. Beale, editors, *Statistical Mechanics (Third Edition)*, page 51. Academic Press, Boston, third edition edition, 2011.

[41] Sivasankar Radhakrishnan, Yuchung Cheng, Hsiao-Keng Jerry Chu, Arvind Jain, and Barath Raghavan. Tcp fast open. In *Conference on Emerging Network Experiment and Technology*, 2011.

[42] Bruce Schneier. Attacking tor: how the nsa targets users' online anonymity. 2013.

[43] Meng Shen, Zhenbo Gao, Liehuang Zhu, and Ke Xu. Efficient fine-grained website fingerprinting via encrypted traffic analysis with deep learning. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10, 2021.

[44] Yan Shi and Subir Biswas. Website fingerprinting using traffic analysis of dynamic webpages. In *2014 IEEE Global Communications Conference*, pages 557–563, 2014.

[45] Sandra Siby, Ludovic Barman, Christopher Wood, Marwan Fayed, Nick Sullivan, and Carmela Troncoso. You get padding, everybody gets padding! you get privacy? evaluating practical quic website fingerprinting protections for the masses, 2022.

[46] Sandra Deepthy Siby, Ludovic Barman, Christopher A. Wood, Marwan M. Fayed, Nick Sullivan, and Carmela Troncoso. Evaluating practical quic website fingerprinting defenses for the masses. *Proc. Priv. Enhancing Technol.*, 2023:79–95, 2023.

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

[48] Jean-Pierre Smith, Luca Dolfi, Prateek Mittal, and Adrian Perrig. QCSD: A QUIC Client-Side Website-Fingerprinting defence framework. In *31st USENIX Security*

Symposium (USENIX Security 22), pages 771–789, Boston, MA, August 2022. USENIX Association.

[49] Jean-Pierre Smith, Prateek Mittal, and Adrian Perrig. Website fingerprinting in the age of quic. *Proceedings on Privacy Enhancing Technologies*, 2021:48 – 69, 2021.

[50] Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard. Exploiting data-usage statistics for website fingerprinting attacks on android. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '16, page 49–60, New York, NY, USA, 2016. Association for Computing Machinery.

[51] Qixiang Sun, D.R. Simon, Yi-Min Wang, W. Russell, V.N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 19–30, 2002.

[52] Martin Thomson and Sean Turner. Using TLS to Secure QUIC. RFC 9001, May 2021.

[53] Cong Tian, Dengpan Ye, and Chuanxi Chen. Tiny wfp: Lightweight and effective website fingerprinting via wavelet multi-resolution analysis. In Mehdi Tibouchi and XiaoFeng Wang, editors, *Applied Cryptography and Network Security*, pages 237–259, Cham, 2023. Springer Nature Switzerland.

[54] Kai Wang, Liyun Chen, and Xingkai Chen. *Website Fingerprinting Attack Method Based on DNS Resolution Sequence: Applications and Techniques in Cyber Security and Intelligence*, pages 1227–1233. 01 2019.

[55] Kailong Wang, Junzhe Zhang, Guangdong Bai, Ryan Ko, and Jin Song Dong. It's not just the site, it's the contents: Intra-domain fingerprinting social media websites through cdn bursts. In *Proceedings of the Web Conference 2021*, WWW '21, page 2142–2153, New York, NY, USA, 2021. Association for Computing Machinery.

[56] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, page 143–157, USA, 2014. USENIX Association.

[57] 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.

[58] Charles Wright, Scott Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. 01 2009.

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

[60] Jingjing Zhang, Xianming Gao, Lin Yang, Tao Feng, Dongyang Li, Qiang Wang,

and Ruhul Amin. A systematic approach to formal analysis of quic handshake protocol using symbolic model checking. *Sec. and Commun. Netw.*, 2021, jan 2021.

Appendix A

Appendix

A.1 GitHub Repository

Link to repository.

A.2 Docker Images

Caddy Server Docker Image

Caddy Crawler Docker Image

Run the caddy server using this command :

```
$ docker run --hostname=f508090b710f --env=PATH=/usr/local/sbin
   :/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin --env=
   CADDY_VERSION=v2.7.5 --env=XDG_CONFIG_HOME=/config --env=
   XDG_DATA_HOME=/data --workdir=/srv -p 2020:443 -p 2020:443/udp
    --restart=no --label='org.opencontainers.image.description=a␣
   powerful,␣enterprise-ready,␣open␣source␣web␣server␣with␣
   automatic␣HTTPS␣written␣in␣Go' --label='org.opencontainers.
   image.documentation=https://caddyserver.com/docs' --label='org.
   opencontainers.image.licenses=Apache-2.0' --label='org.
   opencontainers.image.source=https://github.com/caddyserver/
   caddy-docker' --label='org.opencontainers.image.title=Caddy'
   --label='org.opencontainers.image.url=https://caddyserver.com'
    --label='org.opencontainers.image.vendor=Light␣Code␣Labs' --
   label='org.opencontainers.image.version=v2.7.5' --runtime=runc
    -t -d andrewellison/my-caddy-server
```

Run the caddy crawler using this command :

```
$ docker run --hostname=25378218787c --user=root --env=PATH=/usr/
   local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin --
   env=DEBIAN_FRONTEND=noninteractive --env=
```

```
DEBCONF_NONINTERACTIVE_SEEN=true --env=TZ=UTC --env=SEL_USER
=seluser --env=SEL_UID=1200 --env=SEL_GID=1201 --env=HOME=/
home/seluser --env=SEL_DIR=/opt/selenium --env=EXTERNAL_JARS
=/external_jars --env=SE_DOWNLOAD_DIR=/home/seluser/Downloads
 --env=SE_BIND_HOST=false --env=LANG_WHICH=en --env=
LANG_WHERE=US --env=ENCODING=UTF-8 --env=LANGUAGE=en_US.UTF-8
 --env=LANG=en_US.UTF-8 --env=NOVNC_VERSION=1.4.0 --env=
WEBSOCKIFY_VERSION=0.11.0 --env=SE_VNC_PASSWORD=secret --env=
SE_SCREEN_WIDTH=1360 --env=SE_SCREEN_HEIGHT=1020 --env=
SE_SCREEN_DEPTH=24 --env=SE_SCREEN_DPI=96 --env=SE_START_XVFB
=true --env=SE_START_VNC=true --env=SE_START_NO_VNC=true --
env=SE_NO_VNC_PORT=7900 --env=SE_VNC_PORT=5900 --env=DISPLAY
=:99.0 --env=DISPLAY_NUM=99 --env=CONFIG_FILE=/opt/selenium/
config.toml --env=GENERATE_CONFIG=true --env=
SE_DRAIN_AFTER_SESSION_COUNT=0 --env=SE_OFFLINE=true --env=
SE_NODE_MAX_SESSIONS=1 --env=SE_NODE_SESSION_TIMEOUT=300 --
env=SE_NODE_OVERRIDE_MAX_SESSIONS=false --env=
DBUS_SESSION_BUS_ADDRESS=/dev/null --env=SE_RELAX_CHECKS=true
 --workdir=/usr/src/app -p 443:443 -p 443:443/udp -p 4444:4444
 -p 4444:4444/udp -p 7900:7900 -p 7900:7900/udp --restart=no
--label='authors=' --label='org.opencontainers.image.ref.name=
ubuntu' --label='org.opencontainers.image.version=22.04' --
runtime=runc -t -d andrewellison/my-caddy-crawler
```

A.3 Data Collection Graphs

See Figures A.1 and A.2 for the final cleaned traffic collection.

See Fig. A.3 for a suspect QUIC boxplot, after an initial clean.

A.4 Grid Search Values

See Figure A.4.

A.5 More Differences in Methods and their Effects on the Results

Number of Sites:

I collected and trained this paper's website fingerprinters using 94 websites rather than 92. This should make it harder for this paper's fingerprinter to predict the website being visited accurately. However, the effect of this should be minimal as two extra websites should not have a great effect since it is a small difference, but this will still have some effect.

Website Size Distribution:

I use newer versions of sites; generally, sites are getting larger as they become more modern with more content. This will most likely make website fingerprinting easier as if websites have more content, which stands to reason that their traffic will look more distinct from other websites. Although this should mean that this paper's sites are slightly easier to fingerprint rather than harder, it does not explain the discrepancy between the two sets.

Collection statistics:

While they provide some information regarding downloading websites, the WFP on Early QUIC paper does not provide information on the size or distribution of their traffic collection. This is arguably much more important than the download size of a website, as this is the data that will be used by the website fingerprinting models after processing and feature extraction. During the implementation of this paper, there were several less-than-obvious discrepancies in the collected data, but these were resolved by visualising the distributions and removing outliers. The original paper does not mention removing outliers or visualising the collected traffic in any way. This is frustrating as this is likely why our results do not match, but they have not provided a sufficient way to verify this. Also, as it was not mentioned, perhaps the original paper did include outliers in their collection. However, this would most likely result in a worse accuracy as there is a larger distribution of capture sizes for each domain. But again, there is no way to verify the extent to which their collection was different to mine due to their lack of information regarding it.

A very likely reason that our solution would not match is the results of the simple features extracted from the traffic for this paper's collection may be much less distinctive than those found in the paper. It is reasonable to think that the packets seen in this paper's collection, with newer websites and up-to-date versions of different software, are more uniform across sites than the ones used in their paper. For example, when observing the results of the early features for this paper's collection, I tended to see most packets being classified as large, if all collections follow a similar pattern, then the fingerprinting will be very weak. If the paper provided some insight into the distribution of the collected packet sizes, this would also help verify this.

Local Machine:

I performed this paper's experiments on a local machine and achieved isolation between client and server with docker containers. The use of multiple machines will affect the experiment results. Two of the extracted features are the interarrival time of packets and total transmission time, which will be greatly reduced in this paper's implementation. However, the early top-k fingerprinting experiment used only simple features, which is irrelevant.

Using a local machine setup, there is likely less noise than communicating across different machines, as there are fewer points of failure for each packet. Intuitively, this should make fingerprinting easier, as it should make patterns in the traffic more distinct. Therefore, this does not explain the decrease in accuracy across all tests.

However, perhaps this isn't the case. Because of other processes communicating across this paper's same local machine, there is much more interference than on an isolated network whose only purpose is to collect traffic traces. This interference could mean that many of this paper's packets get lost, missed, and resent, resulting in the same packet possibly appearing multiple times in this paper's collected traffic. This would lower accuracy as fewer of the collected packets are meaningful, so the traffic pattern for a specific website will be less distinct, making it harder to fingerprint.

Feature Definitions:

As stated in the Methodology section, the features they use are poorly defined. This leaves room for ambiguity and possible differences in interpreting the features. For example, they do not use the value of n (number of packets) in the feature definitions; one must assume that this value was used; however, it is not referenced in the paper. This does not apply to results that only rely on the simple features, though, but rather to the transfer features, so it would not explain the large difference in results when just simple features are used.

A.6 Tuned Hyperparameters

HTTP2, n=40:

- Extra Trees = {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 400}
- Random Forest = {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 500}
- K-Nearest Neighbours = {'n_neighbors': 9}
- Gaussian Naïve Bayes = {'var_smoothing': 0.0023101297000831605}
- Support Vector Classifier = {'C': 100, 'gamma': 0.01}

HTTP3, n=40:

- Extra Trees = {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 500}
- Random Forest = {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 500}
- K-Nearest Neighbours = {'n_neighbors': 13}
- Gaussian Naïve Bayes = {'var_smoothing': 0.0023101297000831605}
- Support Vector Classifier = {'C': 10, 'gamma': 0.01}

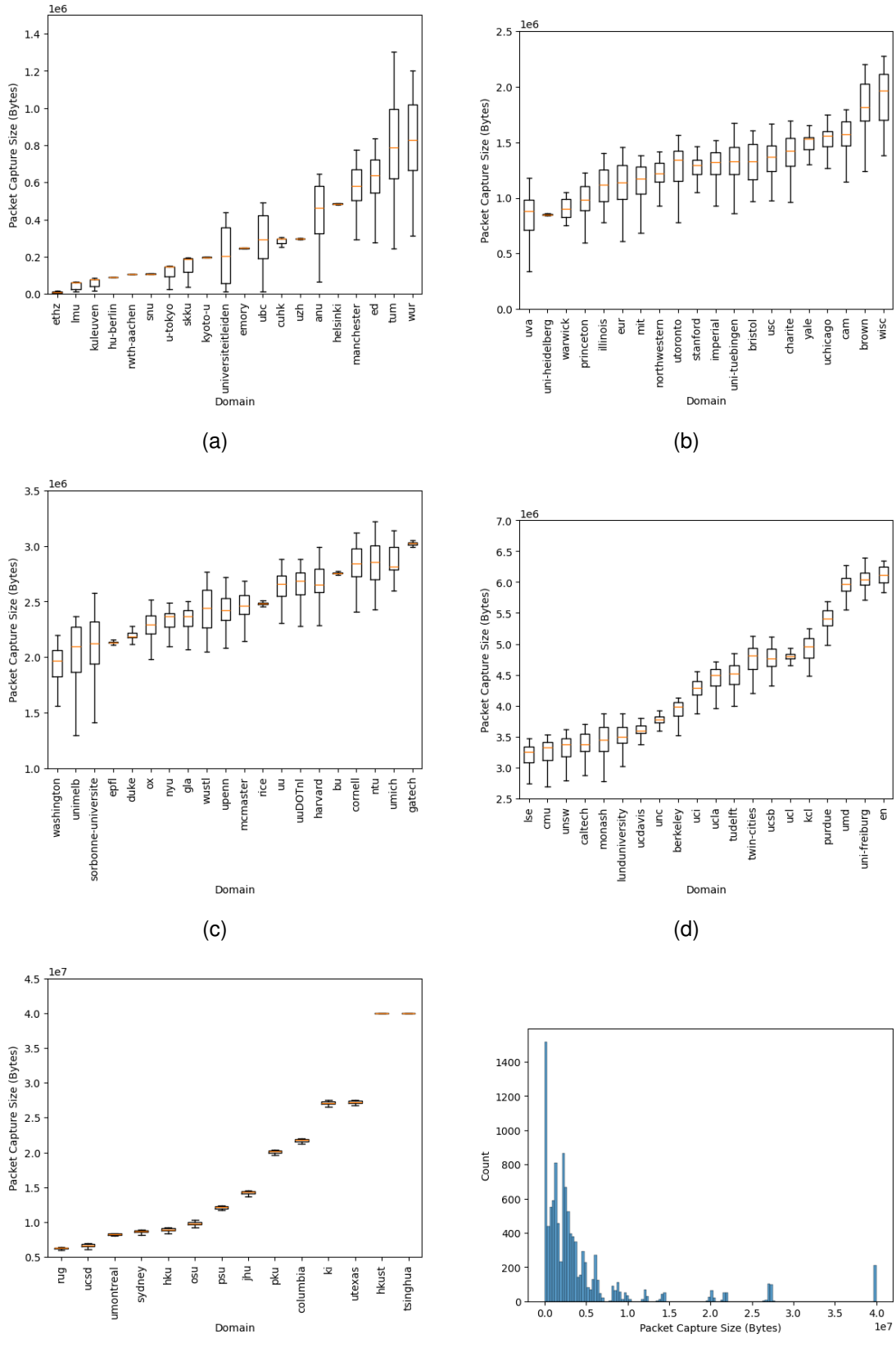

Figure A.1: Data Collection for HTTP2

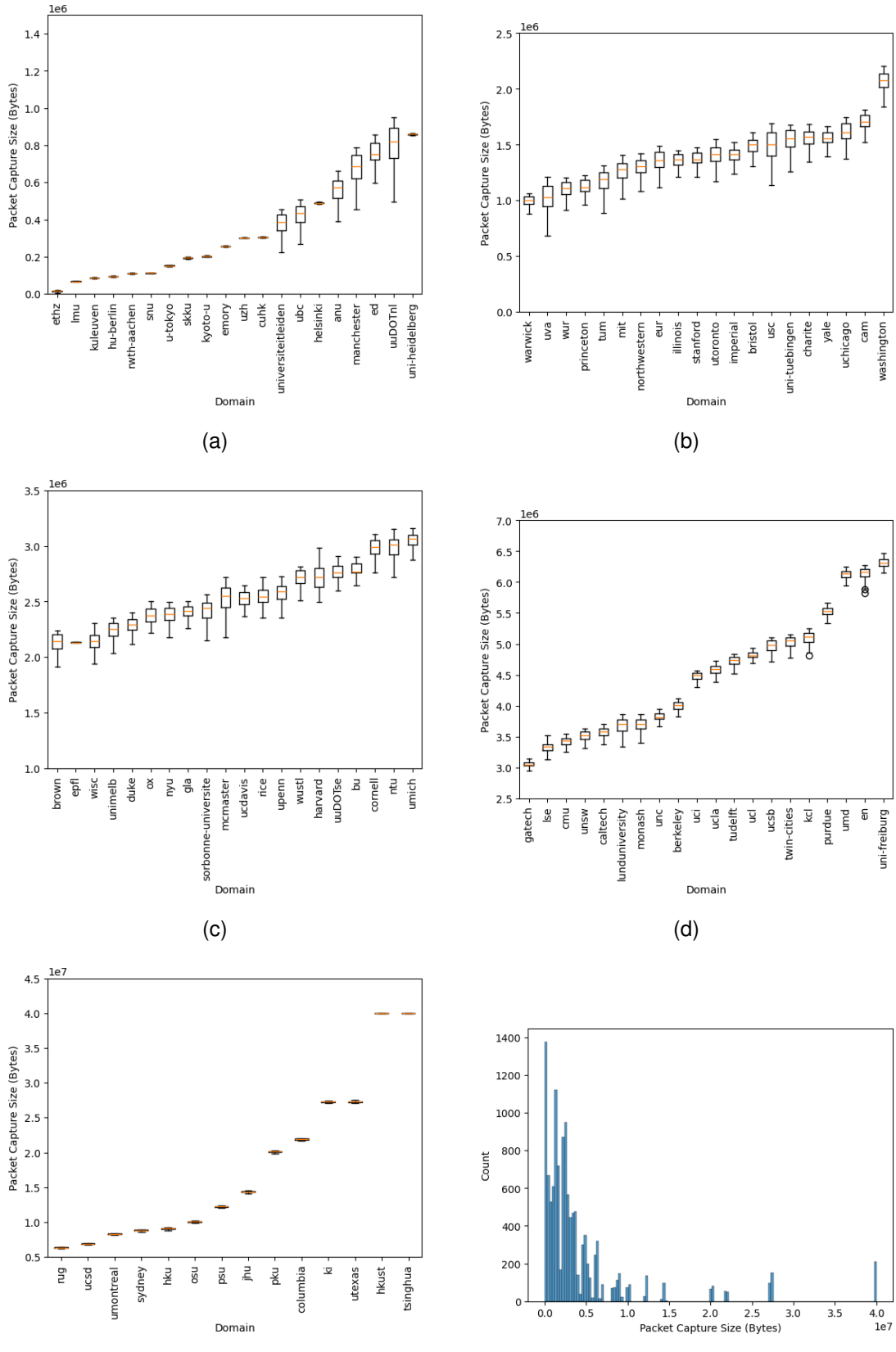

Figure A.2: Data Collection for HTTP3

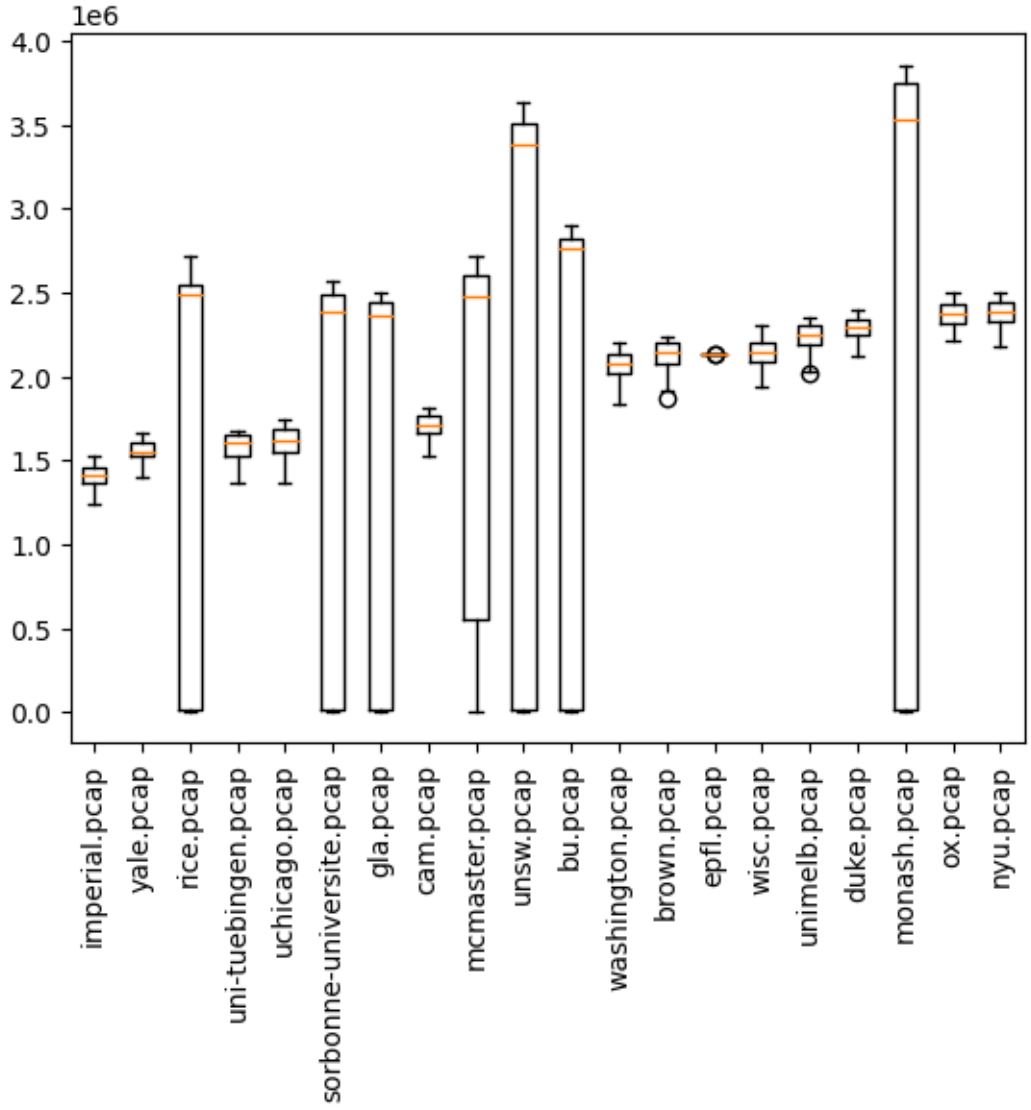

Figure A.3: Suspicious QUIC boxplot after the first round of cleaning

```
et_grid = {
    'n_estimators':[100,200,300,400,500],
    'max_features':[None, 'sqrt', 'log2'],
    'max_depth' : max_depth_arr,
    'min_samples_split' : [2,5,10],
    'min_samples_leaf' : [1,2,4],
}

rf_grid = {
    'n_estimators':[100,200,300,400,500],
    'max_features':[None, 'sqrt', 'log2'],
    'max_depth' : max_depth_arr,
    'min_samples_split' : [2,5,10],
    'min_samples_leaf' : [1,2,4],
}

knn_grid = {
    'n_neighbors' : list(range(1, 50, 2))
}

nb_grid = {
    'var_smoothing' : np.logspace(0,-9,num=100)
}

svc_grid = {
    'C' : [0.1, 1, 10, 100, 1000],
    'gamma' : [1, 0.1, 0.01, 0.001, 0.0001],
}
```

Figure A.4: Values used for exhaustive grid search tuning