Time-sensitive VPN traffic classification

by

Cristina Stoleriu

Thesis advisor:	G. Smaragdakis
Daily co-supervisor:	E. Bassetti
Project Duration:	November 2024-September 2025
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science

Cover: Canadarm 2 Robotic Arm Grapples SpaceX Dragon by NASA under CC BY-NC 2.0 (Modified)

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

Globalization and the aftermath of Covid-19 saw an increasing number of people working from home, using their company's VPN to access various internal resources hosted on enterprise servers. Similarly, VPNs have experienced more widespread use among the general public in recent years as a result of a growing understanding of digital privacy. While previous research has shown that VPNs are vulnerable to fingerprinting, our hypothesis is that attackers can discern even more precise information about VPN traffic. Specifically, because the VPN server and endpoint are hosted physically close in the business use case but not in the internet browsing case, an attacker can exploit the Round Trip Time (RTT) difference to distinguish between the two scenarios.

To compute the RTT of an encrypted VPN connection, we devise a method to identify underlying request-response packet pairs. We target TLS handshakes, using the fact that the order and size of their constituent packets remain unaffected by encryption and consistent among most connections. The latency is computed by subtracting the arrival times of the identified handshake packets. Applying our method on synthetic data, we find that the mean and median RTT of a business-use VPN are lower than those of a private-use VPN, implying an attacker can differentiate between the two scenarios by simply observing encrypted data.

Contents

1 Introduction

According to a 2023 Pew Research Center survey, 35% of workers in the US with jobs that can be done remotely were working from home [13]. Almost 9% of the total EU workforce worked from home in 2023 [5], while the US showed a larger percentage of 13.8% [3]. Similarly, due to the efforts of globalization, more people can work remotely while living in a different country. While the prospect of working from home is attractive to employees, it brings with it a new array of problems to consider for the employers. Aside from the difficulties of managing communication, companies must put greater emphasis on security. The infrastructure becomes more distributed, sensitive data travels a longer distance and is thus more vulnerable to interception by unauthorized parties. On-site services and resources must be made available to people working from home. As such, many businesses that want to ensure secure remote access and authorization use VPNs. The solution is elegant and flexible; employees can connect to the company network through a VPN server hosted on company premises, thus being able to access any internal resources; the tunnel is encrypted, adding a layer of security and authorization, and the solution can be quickly and easily integrated into the existing workflow. Company VPNs are indeed a reality that many face, as a 2024 survey showed that for 39% of 1000 Americans using a VPN was a job requirement [4].

On the other hand, people are using VPNs for personal reasons in increasing numbers. While the internet continues to carve its place in modern culture, more people become aware of what personal information is available on the web and how it can be used to track them. Commercial VPN providers like NordVPN and ExpressVPN advertise themselves as privacy-preserving services that protect from online tracking and identification, offer an added layer of security and ensure that third parties cannot listen in on conversations [6, 11]. Moreover, some countries enforce strict internet censorship laws. For example, the Great Firewall of China filters search results and controls private communication, while the Russian internet limits access to news about Ukraine. In such cases, the location obfuscation that VPNs provide makes their use crucial for journalists and citizens accessing the news or spreading different political beliefs.

However, despite some VPN services' promises of untraceability and obfuscation, research has shown that OpenVPN, the protocol most commonly used by commercial VPNs including the two mentioned above, can be easily identified by analyzing specific patterns like packet size and flow [20]. Moreover, with the aid of machine learning, researchers have discerned between type of traffic and even guessed the sites VPN users are visiting within a small margin of error [15].

We consider it important to explore the vulnerabilities of VPNs and what information attackers can potentially obtain by just capturing traffic. VPN providers should stay informed of such weak points in order to strengthen security, while users deserve to use a service with full knowledge of the potential risks they are accepting.

In this thesis we explore the feasibility of distinguishing a more abstract characteristic of VPN traffic: whether the user is using the VPN for private browsing or job-related purposes. To this end we exploit the fact that a VPN's latency is dependent on the physical distance between the VPN server and web endpoint. The latency is reflected in the Round Trip Time, the time between the client sending a request and receiving a response. Intuitively, a business is likely to host many of its services, including its VPN, on servers that are physically close together in the same room or building. Thus, the latency of accessing company resources and services through the company VPN would be very small. In comparison, the distance between a commercial VPN server and a random website's server would be much greater; in addition, the commercial VPN server itself tends to be far away from the user, typically in other countries. These observations imply that a VPN used for private browsing will exhibit larger latency than one used for job-related activities.

1.1. Research Question

We position ourselves as a passive adversary observing traffic between a client and VPN server. We seek to answer the following question:

Can latency be used to discriminate between private-use and business-use VPNs?

To this end we pose a set of sub-questions:

- *How can we compute the RTT of an encrypted connection? How precise is our method?*
- *What classification method can we use to distinguish the two use-cases?*

1.2. Contributions

In the process of answering our research questions, we make a number of contributions:

- We provide a set of Docker environments that generate synthetic VPN data according to the two scenarios. Using our set-up, we compile a dataset of VPN packet flows that can be used for future reseach.
- We develop a method of computing the RTT of an encrypted VPN connection by identifying encapsulated TLS handshakes and assess its accuracy.
- We show the effectiveness of statistical analysis in discriminating between the latency of business use and private use VPNs.
- We prove that protocol handshakes like the TCP and TLS handshakes are reasonable targets for fingerprinting algorithms and can constitute vulnerabilities even in encrypted traffic.
- Our results support and strengthen the notion that latency is a powerful feature in traffic analysis, able to reveal information about encrypted or encapsulated flows.

2 Background

2.1. Virtual Private Networks

A VPN is a service that creates and manages an encrypted virtual network over an untrusted public network. There are various ways to create the private network; popular VPN protocols attach to the TUN/TAP device, the kernel space virtual network device, to handle routing of either OSI Layer 2 or Layer 3 packets. The program that sets up the private network, known as the VPN server, is responsible for handling incoming connections, routing packets, and performing encryption. When connecting, clients are given IP addresses on the private network.

VPNs implement tunneling protocols: algorithms that encapsulate packets, repackaging the data and using the payload of the new packet to carry the original datagram. The additional layer of security that VPNs advertise is a result of the encryption performed when encapsulating packets. In an insecure connection (e.g. HTTP instead of HTTPS) packets contains plaintext data like usernames and passwords that ISPs and potential attackers can directly access. In a VPN connection, however, the same entities can only see the encapsulated packet, whose payload is encrypted.

The basic data flow of a VPN connection can be seen in Figure 2.1. Figure 2.2 shows a more detailed view, including the use of the TUN/TAP device and the routing the VPN server and client perform.

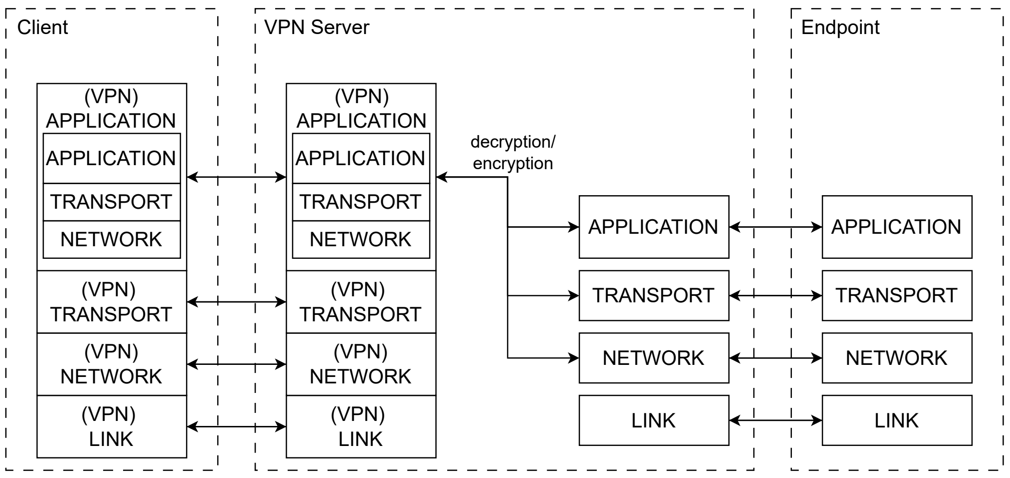

Figure 2.1: VPN data flow

Due to their capabilities VPNs see two main types of use:

- Remote access - VPNs can provide remote access to local area networks or intranets. This is typically used to connect multiple company sites or to allow remote workers access to an internal

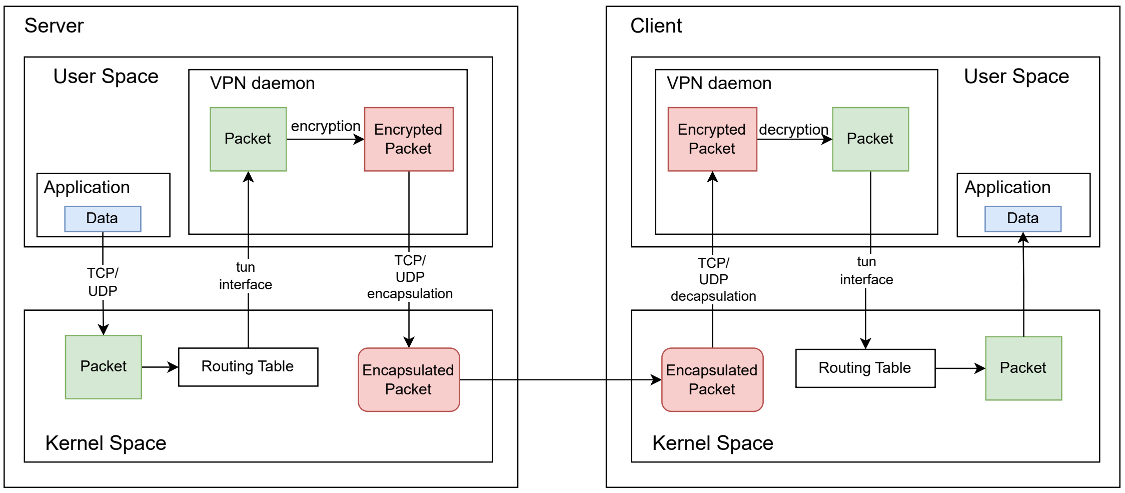

Figure 2.2: VPN detailed data flow

company network and its on-site data and services. Thanks to the encrypted nature of VPNs, such connections also ensure the safe transmission of potentially sensitive data over the public internet.

- Location obfuscation - When the VPN server and client are located geographically far, the server can act like a location obfuscating proxy. Because location as accurate as the zip code can be deduced from an IP address, such VPN servers simply set up the internal network on a range of IP addresses corresponding to their physical location. A connecting client receives an address on the private network and thus appears to be located at the server's geographical position when using the VPN. Clients typically connect to servers in other countries to take advantage of different internet laws and more lax censorship, connecting to websites that were otherwise inaccessible from their original country. This type of service is the one most commonly offered by commercial VPN providers like NordVPN; it can be used to circumvent censorship, view blocked foreign material or avoid tracking when accessing material monitored by the law. Additionally, such VPNs can also be used as an extra layer of protection against man in the middle attacks when browsing the web, thanks to the encryption they provide.

2.2. The OpenVPN Protocol

OpenVPN is an open-source VPN protocol that is used and built upon by many commercial VPN operators. It is highly versatile, offering tunneling services through the tap or tun device, and on top of UDP and TCP. Under best practices, OpenVPN creates a TLS connection between the server and a client for an additional layer of security. It supports several ciphers, including AES-GCM, CHACHA-POLY, AES-CBC, and other deprecated ciphers. As of version 2.5 OpenVPN defaults to AES-256-GCM if both the client and server support it.

The OpenVPN protocol distinguishes between two main types of packets: the first, P_CONTROL packets, transmit information about the VPN connection. They are used at the beginning of a connection to exchange a handshake that negotiates the cipher set, exchanges keys and sets up the TLS connection between client and server. P_DATA_V2 packets encapsulate the conversation between client and website. The basic structure of a data packet can be seen in Figure 2.3. The 5-bit opcode is the most important field, as its value determines the packet types. The remaining 3 bits of the byte constitute the key id, identifying the data channel key method. The following 3 bytes represent the peer id, the unique client id. Finally, the payload contains the encrypted plaintext and additional metadata fields like the HMAC and IV, which are dependent on the negotiated data channel cipher.

Starting with version 2.4 released in 2016, OpenVPN servers support only AEAD ciphers: AES-256-GCM, AES-128-GCM and CHACHA20-POLY1305. All 3 ciphers add 27 bytes of overhead to the plaintext packet, and their structure is documented on OpenVPN's crypto module page [12]. Previous versions of OpenVPN supported weaker ciphers like AES-CBC and BF-CBC, which are now deprecated; a

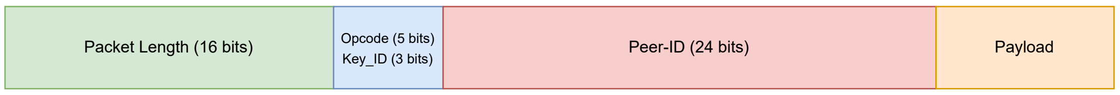

Figure 2.3: OpenVPN P_DATA_V2 structure

backwards compatibility option can be added to modern OpenVPN servers, however. The protocol guarantees the correct order of packets it delivers but it will frequently group multiple packets in one frame.

2.3. TLS Handshakes

TLS is an application layer cryptographic protocol that establishes a secure connection between a server and client by encrypting packet payloads. TLS used in conjuntion with HTTP is known as HTTPS, the default secure protocol used by most modern websites. Like TCP, TLS uses a handshake to initiate a connection. Initially, an asymmetric cipher is used to compute a shared secret between the client and server; session keys like the IV and private keys derived from this secret are used to then encrypt traffic.

TLS 1.2 stands as the most widely supported version of the protocol. The handshake proceeds as follows:

- The client sends a **ClientHello** packet advertising which TLS versions and ciphers it supports and the client random, a random value.
- The server responds with a **ServerHello**, which includes the cipher suite and protocol version it chose and the server random.
- The server sends its SSL **Certificate** and indicates it finished negotiating with a **ServerHelloDone** packet.
- The client confirms the server's identity via the certificate, computes a premaster secret using information it receievd from the server and shares it in a **ClientKeyExchange** packet. Using the random values and public keys, the peers generate the master secret.
- Finally, the client and server tell each other that communication will be encrypted with **ChangeCipherSpec** packets.

Version 1.3 streamlines the handshake. By removing many vulnerable ciphers, the server and client exchange data in only one round trip. The client computes the premaster secret immediately thanks to the very short list of possible ciphers and sends it in the **ClientHello**. Thus, the **ClientKeyExchange** step is not needed.

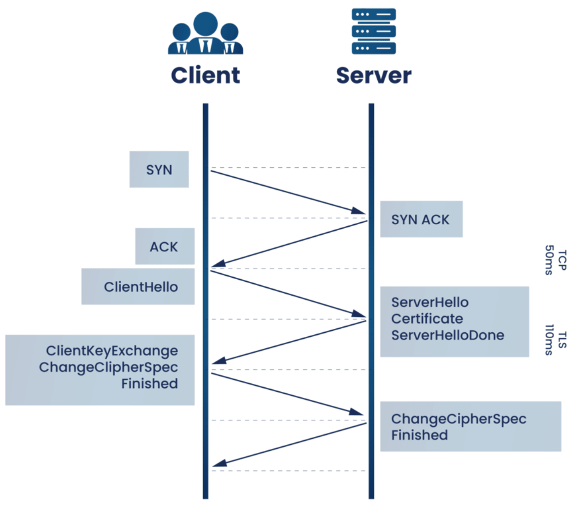

Figure 2.4: TLS 1.2 Handshake

3 Related Work

Research into VPNs and proxied connections explores many areas, from improving protocol efficiency and speed to developing new protocols for particular use cases like mobile phones or cloud computing. We focus our attention on papers relating to VPN and proxy fingerprinting, seeking what methods and features are most common.

Previous work in the area of VPN and proxy fingerprinting can be separated into two main categories: deep-packet inspection (DPI) based and machine learning approaches. The latter, more wide spread method involves training a model using various packet and flow features, then predicting whether unseen traffic is VPN traffic or not [7, 1, 8]. Other similar studies strive to also classify the type of VPN traffic (internet browing, video, VoIP, etc.) [15]. Many articles use the ISCXVPN2016 dataset [8] to conduct their experiments.

DPI approaches usually involve passive fingerprinting, where an observer monitors traffic and extracts relevant packet information to create heuristics. Xue et al. [20] took advantage of simple but identifying features of OpenVPN packets when developing fingerprinting methods. The first step of their algorithm analyzes sequences of packets to find an opcode sequence consistent with the OpenVPN handshake. Additionally, they exploit the consistent sizesof P_ACK packets; the number of P_ACK sized packets decreasing and eventually disappearing as the conversation continues indicates a likely OpenVPN connection. The second step of their system is an active probing part, wherein special packets are sent to potential VPN connections discovered with passive fingerprinting; the connection is confirmed to use VPN if the server responds in a way consistent with OpenVPN servers.

Latency in particular has proven to be a powerful traffic feature when studying encrypted traffic. Ramesh et al. [14] showed that in many cases, checking if the difference in RTT between the transport and application or network and transport layers is larger than 50ms can reveal proxy use. This method was also used to identify VPN users connected to far away servers. Web Tap, which detects covert HTTP tunnels, takes advantage of the fact that malicious servers typically make requests based on a timer and uses the inter packet delay as a filter [2]. Similarly, the RTT has been used to geolocate internet users. By using previous latency measures from set landmarks and latency-distance datasets, Arif et al. [10] developed GeoWeight, an algorithm that identified the location of 6 targets within 44km of error. In the proxy domain, Wang et al. [17] were also able to trace the path of proxy users through network hops using delay information. In 2018, Weinberg et al. [18] used TCP handshakes to compute latency and geolocate 2269 VPN servers from 7 providers, showing that a third are clustered in European countries with cheap server hosting possibilities.

The strict ordering and predictable packet sizes of TLS handshakes make them opportune targets for fingerprinting. In a 2024 paper, Xue et al. accurately identified proxied TLS connections (TLS-over-TLS) by observing packet sizes [19]. First, they grouped packets into 4 bins by size: $L_1 = [1 - 160]$, $L_2 = [161 - 600]$, $L_3 = [601 - 1210]$ and $L_4 = [1211+]$. Then, by analyzing 3-grams of discretized packets in both proxied and non-proxied TLS, Xue et al. identified 3-grams that characterize proxied TLS handshakes.

Work Title	**Year**	**Main Focus**	**Features Used**
Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features	2017	VPN fingerprinting	Time-related features
VPN Traffic Classification Based on Payload Length Sequence	2020	VPN fingerprinting	Packet payload length
Characterization of Encrypted and VPN Traffic Using Time-Related Features	2016	VPN fingerprinting	Flow time-based features (e.g. duration of flow)
FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition	2019	Encrypted traffic classification	Packet size and arrival time
OpenVPN is Open to VPN Fingerprinting	2022	VPN fingerprinting	OpenVPN headers and packet type distribution
CalcuLatency: leveraging cross-layer network latency measurements to detect proxy-enabled abuse	2024	Proxy detection	Round Trip Time
Web tap: detecting covert web traffic	2004	Tunneling detection	Inter-request delay, request size and regularity, bandwidth usage
Inter-Packet Delay Based Correlation for Tracing Encrypted Connections through Stepping Stones	2002	Tracing proxied connections	Inter-packet delay
GeoWeight: Internet host geolocation based on a probability model for latency measurement	2010	Geolocation	Latency
How to Catch when Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation	2018	Geolocation	Ping time measurements
Fingerprinting obfuscated proxy traffic with encapsulated TLS handshakes	2024	Proxy detection	Packet size

Table 3.1: Related work summary

As far as we are aware, there is no research focused on distinguishing VPN types using encrypted traffic.

4 Methods

We focus on OpenVPN as our protocol of choice; its status as the basis of several commercial VPNs implies that our methods can thus be applied to those providers. While tailoring our algorithm to other VPN protocols like WireGuard would require changes, the core idea remains the same as we use only packet metadata like size and direction. Additionally, we assume that the presence of the OpenVPN packets has already been confirmed through other means, using deep packet inspection or machine learning for example.

Our threat model consists of a global passive observer situated between the client and VPN server. The adversary has the ability to identify OpenVPN traffic; they record the packet flow and filter out non-OpenVPN traffic. Afterwards, the adversary's goal is to determine whether the client is using the VPN for private browsing reasons or business reasons by analyzing the flow.

Our method is based on two main observations. First, business VPNs are often used by employees to access resources unavailable from outside the company network, like databases, cloud services, team scheduling. etc. Many of these services, including the VPN, are not built from the ground up by businesses but acquired from third party providers. The services are then hosted on company servers, typically placed all together in a room or building. It follows that requests from the VPN to other business services have a very short physical distance to travel and thus a very short RTT to the VPN.

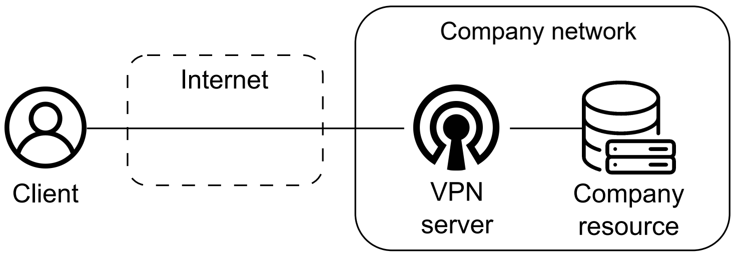

Figure 4.1: Enterprise VPN

On the other hand, VPNs used for private browsing purposes (Figure 4.2) will have a longer RTT. Many want to use a VPN to obfuscate their physical location, which can be determined by their IP address; it is common to connect to a VPN server in a different country or continent. Moreover, using a VPN to browse the internet means accessing website servers that are likely far away from the VPN server itself. For example, if a user in the Netherlands connects to a VPN server in Spain and uses it to access a website hosted on US servers, the distance the request must travel is very large. Crucially, the extreme physical closeness of the business use case is very unlikely to be replicated in the private use scenario; even if two of the three locations (user, VPN, website) are nearby, they will almost never be hosted on the same servers. Thus, the distances between the user and the VPN, and the VPN and the service will be large.

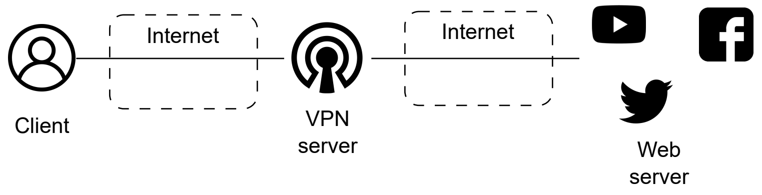

Figure 4.2: Private use VPN

These observations are the basis of our hypothesis. Assuming the use of a VPN is known, we propose that an attacker can identify its use case (private or business) by computing the RTT of the connection. However, since the connection is encrypted, the adversary cannot immediately identify request-response pairs. An additional roadblock is introduced by the fact that OpenVPN groups multiple packets in one frame. Instead, the adversary looks for handshakes, as they are uniform sequences of packets. For example, the TCP and TLS handshakes are sequences of packets with similar lengths, no matter what website the connection is initialized with.

4.1. OpenVPN packet detection

The tshark dissector, while fast, does not perform very well on OpenVPN packets - it often fails to recognize a packet and cannot reconstruct fragmented frames. We build our own OpenVPN dissector using the ScaPy Python library. We only handle the P_DATA_V2 packet type, as we are not interested in the handshake. A segment is found when the frame length is smaller than the length advertised in the OpenVPN header; these fragments are placed in a queue and matched with their second half when it arrives. Finally, because some segments may be orphaned, we introduce a fallback heuristic that tries to identify OpenVPN packets by finding an OpenVPN-specific sequence of bytes, like \x48\x00\x00\x00.

4.2. Computing the RTT

Perhaps the most important step in our process is obtaining accurate RTT values for OpenVPN connections. In a traditional TCP connection, the RTT is computed by subtracting the arrival time of a request from the arrival time of the corresponding response, typically the ACK packet with the correct sequence number. However, performing ACK analysis on VPN packets will compute the RTT to the VPN server rather than to the endpoint behind the VPN. Moreover, as the connection is encrypted, we cannot immediately identify request-response pairs between the client and endpoint.

We strive instead to identify encrypted TLS handshakes. Handshakes in general are more identifiable than other request-response pairs as they follow a deterministic pattern in packet size and direction, metadata accessible even when the flow is encrypted. TLS handshakes always happen between the client and endpoint of the conversation, ensuring that the RTT is computed between the correct targets. Furthermore, HTTPS is a key feature of all modern browsers; the widespread adoption of TLS ensures that TLS handshakes will be present in VPN traffic. Finally, the TLS handshake has two characteristics that make it a better fingerprinting target than other handshakes: its packet size sequence involves more varied sizes, unlike the TCP handshake for example, and it is longer than other handshakes, possibly ensuring fewer false positives.

We use packet size as our distinguishing feature when searching for TLS handshakes. The sizes of packets in the handshake can vary from connection to connection; as such, instead of looking for specific sizes, we use the discretized sizes. This method is adapted from Xue et al., who used 3-grams of discretized packet sizes to identify TLS-over-TLS: TLS packets carrying TLS handshakes as payload, an indication of a proxied connection. We use the same bins as Xue et al. [19] when discretizing, as shown in Table 4.1.

Unlike Xue et al. [19], we cannot use 3-grams. OpenVPN tunnels packets in the order it receives them, meaning encapsulated TCP packets provide their own reliability layer; however, clients can send new requests before receiving responses for previous ones. The server packet queue can further interfere with the packet order. For example, in a non-tunneled connection, one would expect a TLS ServerHello

Packet Size (bytes)	Client Packet Symbol	Server Packet Symbol
1-160	a	w
161-600	b	x
601-1210	c	y
1211+	d	z

Table 4.1: Packet size bins

to immediately follow a ClientHello. In an OpenVPN connection, the ServerHello is put in the VPN server packet queue and processed; meanwhile, the packets ahead in the queue are sent to the client, meaning unrelated packets are sent before the response to the latest client packet. This phenomenon is seen often in our data. As the 3-gram method relies on a strict packet order, it is not applicable in our case.

4.2.1. Stream handshake identification

We generalize the 3-grams identified by Xue et al. to a single sequence that describes a TLS handshake:

$$b, z, z, z/y/a$$

The sequence does not perfectly align with the various 3-grams due to OpenVPN's behavior, which differs from that of the proxies described in the paper. For example, the OpenVPN MTU is set to 1500 bytes by default, meaning the typical ServerHello packet will always be fragmented into at least two z packets.

We attempted to describe the sequence with a regex. However, the synthetic data generated by Selenium is complex. Connecting to a website involves multiple, possibly interleaved, TLS handshakes and additional communication between client and website, leading to many unrelated packets between TLS handshake parts. There is too much possible variation to capture in a regex. Instead, we employ a *stream approach*, iterating over the OpenVPN flow and searching for possible handshake packets while ignoring any packets in between.

We differentiate between 'open' and 'closed' TLS handshakes. An open handshake is a series of discretized packet lengths that is a prefix of the handshake identifying sequence. In other words, an open handshake is a sequence of OpenVPN packets that likely contains at least a ClientHello, but not an entire TLS handshake. The handshake is closed when the final symbol in the handshake sequence is found.

While iterating through a flow, we create and manage a list of open handshakes, each keeping track of the indexes and arrival times of its constituent packets. We create a new handshake whenever a client packet between 300 and 600 bytes is found - the average size of a ClientHello. We then look for two z symbols to signify a (segmented) ServerHello and close the handshake when the final packet is found. We also throw away open handshakes that are not closed within a second, with the assumption that the underlying TLS handshake failed. Finally, the RTT can be computed by subtracting the arrival time of the ClientHello packet from the arrival time of the ServerHello packet.

4.2.2. Improving the false positives rate

While the sequence we look for does indeed describe a TLS handshake, a random sequence of data packets may happen to follow the same pattern of discretized lengths. Our data contains the unencrypted conversation between client and endpoint, which we use to compute how many found handshakes are true positives; if TLS handshake packets are found within the interval defined by the packet indices of the first packet and last packet in the found sequence, then that sequence is a true positive handshake. In actuality, many of the packet sequences found using our initial sequence are false positives and thus do not accurately reflect the ground truth RTT.

We try to minimize this limitation by using a heuristic based on a simple observation: TLS handshakes typically occur immediately following a TCP handshake. By opening handshakes when a TCP SYN packet is found, we remove many false positive sequences that occur later in conversations between the client and webserver.

Unlike TLS handshake packets, the TCP SYN and SYNACK packets do not vary in size. Moreover, the ciphers used by modern OpenVPN incur the same overhead of 27 bytes when encrypting packets. This means that we can find a TCP handshake using the sequence 87, −87, which we prepend to the existing handshake sequence to obtain the TCP-TLS sequence:

$$87, -87, b, z, z, z/y/a.$$

4.3. Classification

Once handshakes have been collected from each data file, we apply statistical analysis to distinguish between the two scenarios. Subtracting packet arrival times, we obtain the RTT for each found handshake; then, we compute the per experiment file mean and median RTT.

Our classification method requires samples from both scenarios. An attacker may attempt to apply further heuristics to determine whether their data belongs to the same or different scenarios. For example, using existing VPN datasets, which typically contain internet browsing data, the attacker can obtain flows corresponding to our proxy scenario. This would be useful if the attacker unknowingly collected only enterprise scenario data, or if the number of proxy flows they collected is too small.

We chose statistical analysis over other classification methods due to its versatility. The RTT will vary greatly among different connections, making it impractical to set a RTT threshold between the proxy and enterprise scenario that is applicable to all flows. The RTT is not a good choice of feature for a machine learning algorithm for the same reason.

5 Dataset

To test our hypothesis we must first gather data. Using data generated by real people brings multiple legal and ethical concerns. A main consideration in the domain of traffic analysis is the potential for user reidentification and leaking of private information. For these reasons, we chose to generate synthetic data through Docker.

We create two Docker environments corresponding to the two scenarios. The two have the same general structure and program loop: a client container connects to an OpenVPN server and sends requests to various websites, depending on the scenario. While the containers are separated in different networks, there is no physical distance affecting the RTT like in real life. As such, we introduce artificial delay using the `tc` Linux utility, which offers different traffic control functionalities including the ability to delay outgoing packets by a specified amount of milliseconds.

5.1. Enterprise scenario

This scenario abstracts the business use case. To make data generation possible, we simplify the use case; the user accesses company resources by visiting a single website, hosted using WordPress and equipped with self-signed SSL certificates to enable TLS access.

Thus, our Docker set up consists of the following containers:

- **client**, which connects to the VPN server and makes between 20 and 30 GET requests to the website. For more realism, the container waits between 1 and 10 seconds between requests. Initially, requests were made using `curl`, which was later replaced by Selenium to simulate the browsing experience as accurately as possible.
- **home gateway** and **remote gateway** add the `tc` delay to outgoing packets. A 2024 survey [16] found that the average latency of 12 commercial VPNs is 105 ms, while a different 2021 report [9] shows an average of 156 ms (excluding outliers) and an average delay jitter of 4.5 ms. Due to the positioning of the capture container as can be seen in Figure 5.1, we must add 100ms delay per gateway to obtain our desired delay value.
- **WordPress website**, a basic, non-edited site hosted on an Apache server that enables TLS; the functionality is native to the WordPress Docker image and only needs self-signed certificates. We generated snakeoil certificates since they are preferred by the image.
- **OpenVPN server**, the server the client connects to running OpenVPN version 2.4. Additionally, it adds a small delay of 5ms to outgoing packets towards the WordPress website, the "business resource". Our motivation is that while there would be low latency between a business VPN and a business resource on physically close servers, it would still be greater than the communication latency between Docker containers hosted on the same machine.
- **capture**, a container capturing and saving the packet flow using `tcpdump`.

There are three networks connecting the containers: **home**, between the client and home gateway, **remote**, connecting the OpenVPN server, website, and remote gateway, and **internet**, facilitating communication between the two gateways. The gateways add `tc` delay only on the **internet** network.

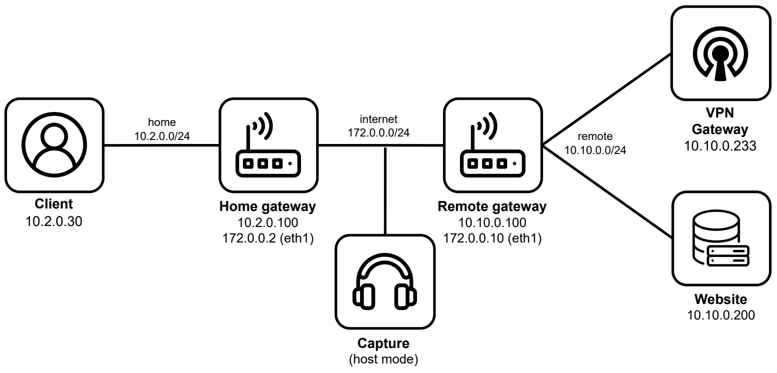

Figure 5.1: Docker Enterprise Scenario Structure

5.2. Proxy scenario

The proxy scenario corresponds to the private use case and includes the following containers:

- **client**, who connects to the VPN server and makes requests to various websites. To simulate a typical browsing experience, the client container visits a set of sites chosen randomly from the top 1000 most visited websites and waits a random amount of seconds between requests.
- **home gateway** and **remote gateway**, which manage the `tc` delay. Like the previous scenario, each adds 100ms of delay to outgoing packets.
- **OpenVPN server**, a container running OpenVPN version 2.4.
- **capture**, the data capturing container

Note that this time the VPN server adds no delay of its own. The RTT is fully dependent on which site the client sends requests to.

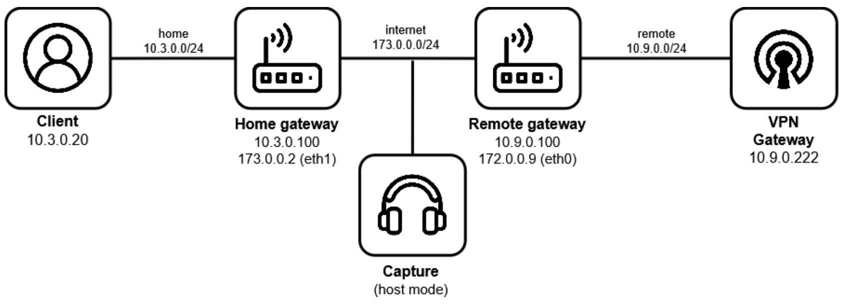

Figure 5.2: Docker Proxy Scenario Structure

The capture container listens on the **internet** interface in both scenarios. However, to gather ground truth data and accurately assess the accuracy of our method, we capture packets on the other interfaces as well.

5.3. Data preprocessing

We run the experiment 20 times for each scenario, saving the flow captured by the **capture** containers. The pcap files are relatively large and contain a lot of information that is not relevant. As such we extract only the features we are interested in and write them to csv files. For our purposes, these are the index, size and arrival time of each OpenVPN packet. We keep track of packet direction by assigning a sign to the packet lengths: packets originating from the client have positive lengths, while packets from the

VPN server are negative. Finally, following Xue et al. [19], we discretize the packet lengths and add them as a dataframe column.

5.4. Real data considerations

While out of the scope our this paper, we invite other researchers to apply our method on real data. However, we recognize the concerns of gathering and using data generated by real people:

- Packet payloads may contain sensitive data like usernames and passwords. As our method does not use the encrypted OpenVPN payloads, we advise collecting only packet headers and other metadata like arrival time.
- IP addresses and and even latency information can be used to identify users, for example through geolocation. Anonymization techniques should be employed while maintaining accuracy as high as possible.

6 Results

We apply our methods to data generated by the Docker environments. Each scenario produces 20 pcap files, which are preprocessed by extracting the features we are interested in to csv. The resulting files are fed as input to the handshake identification algorithms.

While our Docker environments are simplified representations of the real life network architectures, we consider the generated data to be accurate to the real life case. Artificial delay values were chosen by referring to empirical surveys into VPN latency times, ensuring that our computed RTTs are similar to those found in the average real life VPN connection. Similarly, the endpoints the user containers make requests to are representative of the use case: proxy endpoints are chosen from a list of top websites, meaning people using VPNs for their private browsing are likely to access the same sites, while the enterprise endpoint represents any number of company resources that an employee may access, hosted together and thus with minimal variety in latency.

6.1. TLS-only method

We apply the first version of the stream handshake identification method on the generated data, searching only for TLS handshakes. We obtain as output an array of potential handshakes for each experiment, in total 20 arrays for each scenario. To compare the difference between the proxy and enterprise scenarios, we compute the mean RTT for each array of potential handshakes (Figure 6.1).

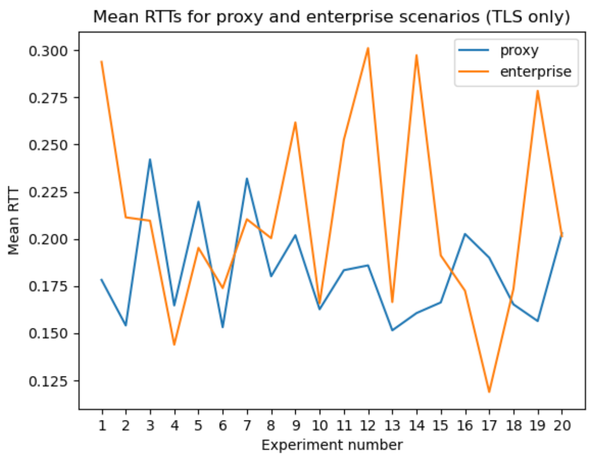

Figure 6.1: Mean RTT per scenario (TLS method)

The mean RTT varies greatly between experiments; for example, the smallest and largest mean RTT in the enterprise scenario are separated by nearly 0.2 seconds, a very long amount of time in networking terms. Due to the similar computed values of the two scenarios, the mean RTT is not a suitable classification metric. However, since it is impacted by outliers, we also compute the median RTT per experiment, shown in Figure 6.2.

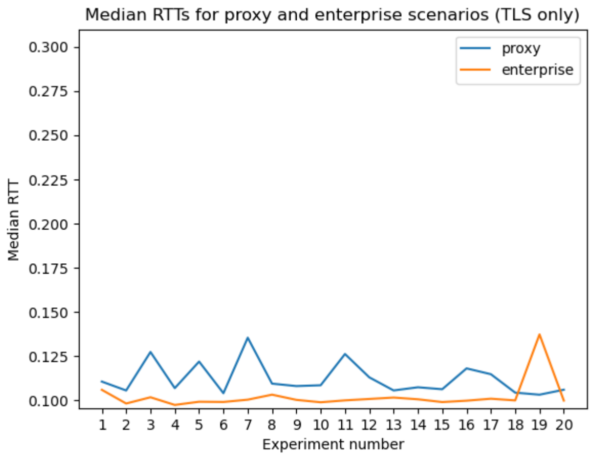

Figure 6.2: Median RTT per scenario (TLS method)

The median RTTs are overall much smaller than the means and they show a clear distinction between the two scenarios: the enterprise median RTT is smaller than that of the proxy scenario, which conforms to our hypothesis. There is still one experiment where the opposite is true, perhaps caused by network congestion or increased load on the side of the web server.

Before we can assert the validity of our results we must ensure that our metrics (the mean and median) are applied on the RTT of actual TLS handshakes. False positives, or sequences of packets misidentified as TLS handshakes, can greatly impact the mean and median RTTs as data packets are exchanged much quicker than handshakes. To ascertain the accuracy of the TLS-only method, we compute the number of true and false positives among found handshakes for each experiment (Figures 6.3 and 6.4).

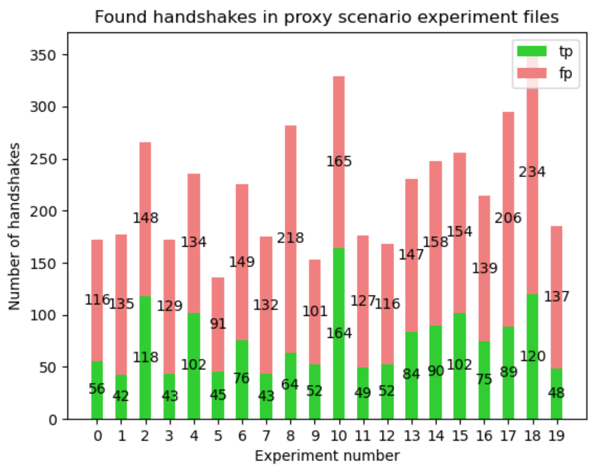

Figure 6.3: Number of true and false positive handshakes in proxy scenario (TLS method)

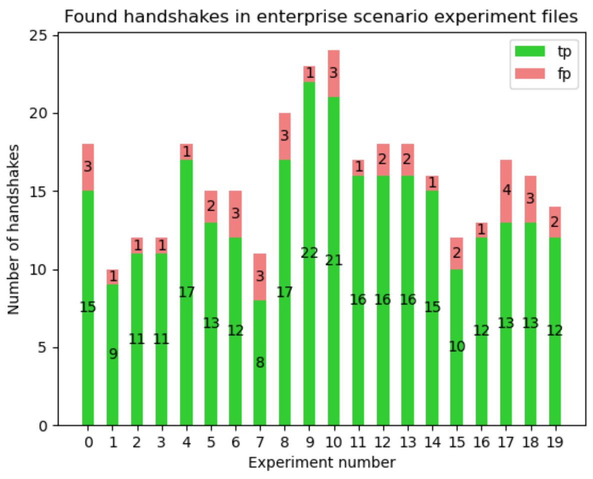

Figure 6.4: Number of true and false positive handshakes in enterprise scenario (TLS method)

As expected, this method incurs a large amount of false positives. This can be explained by two main factors: the sequence length is relatively small, increasing the chance that non-handshake packet sequences will have the necessary sizes. Moreover, the use of only discretized sizes contributes to the ambiguity of the sequence. Random TLS data packets, QUIC packets, and interleaved TLS handshakes may all be falsely flagged as TLS handshakes.

Important to note is that our goal is not to identify all TLS handshakes in a flow, which is why we are not interested in false negatives. Our goal is to obtain the RTT, which can be computed from only a few handshakes; for this reason it is important that we maximize the number of true positives among found handshakes, rather than maximizing the number of found handshakes. Additionally, another detail to point out is that that many more handshakes are found in the proxy scenario than in the enterprise one; this can be explained by the simplified site the enterprise client sends requests to. Real sites such as the ones used in the proxy case often require multiple TLS handshakes with different third-parties like ad and site analytics providers.

We can compute the precision of the TLS-only handshake identification method, defined as the number of true positives divided by the total number of positives, to obtain an average of 0.33 for the proxy scenario. While the enterprise scenario precision is much higher, requests are only sent to a dummy website. Since the proxy scenario interacts with multiple actual websites, we can view the proxy scenario precision to be more indicative of real data than the enterprise one.

Finally, we seek to understand the impact of false positives on the RTT, which we can judge by plotting the differece in mean RTT between true and false positives for each scenario.

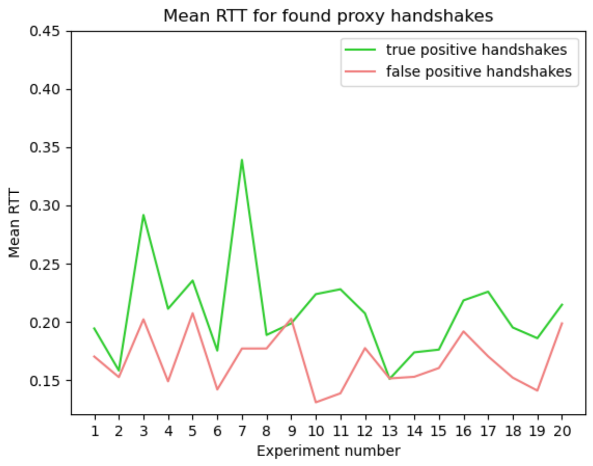

Figure 6.5: Difference in mean RTT between true positive and false positive handshakes in proxy scenario (TLS method)

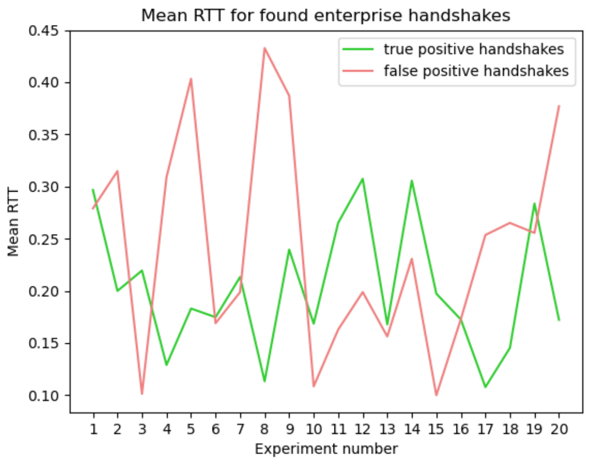

Figure 6.6: Difference in mean RTT between true positive and false positive handshakes in enterprise scenario (TLS method)

The false positives certainly negatively impact the RTT - the mean RTT of proxy false positives (Figure 6.5) being smaller than that of true positives, combined with the large number of false positives means that the computed proxy mean RTT is smaller than its ground truth value. Conversely, the mean RTT of enterprise false positives (Figure 6.6) is at times larger than that of true positives, raising the overall mean RTT. These results highlight the importance of accurately identifying handshakes.

False positives outnumber true positives in each experiment, and their values negatively influence the mean. We conclude that the TLS-only method does not accurately identify TLS handshakes and that computed RTTs are not representative of the ground truth RTT. As such, we turn our attention to the TCP-TLS method, designed to tackle this exact limitation.

6.2. TCP-TLS method

We note a change in results when including the TCP handshake in our sequence: the mean RTTs (Figure 6.7) are constrained to a smaller interval of values when using the TCP-TLS method - the highest and lowest values are separated by 0.12 seconds, compared to the 0.2 seconds of the TLS-only method. Despite this fact, the enterprise scenario means still take highly variable values inside this interval and show no clear distinction from the proxy scenario means. Calculating the mean on unprocessed data does not result in a useful classification metric. We compute the median instead.

Like when using the TLS-only method, the median RTTs in Figure 6.8 demonstrate the difference between the two scenarios: the proxy scenario median RTT is higher than the enterprise one. We also note that the values of the median, like the mean, are smaller when using the TCP-TLS method.

We compute the number of true and false positives in Figures 6.5 and 6.6. The TCP-TLS method places more constraints on the sequence that identified handshakes. The longer length of the sequence and the

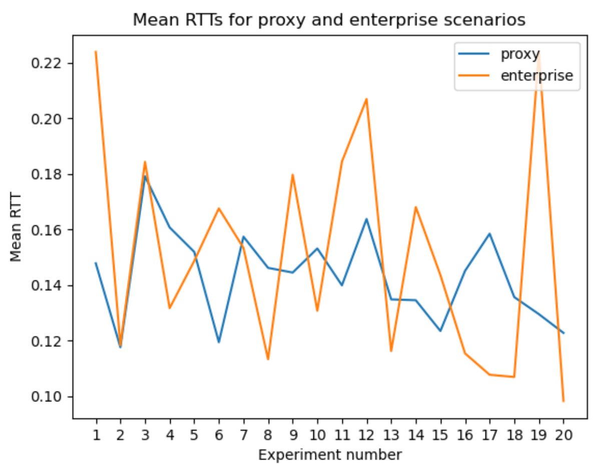

Figure 6.7: Mean RTT per scenario (TCP-TLS method)

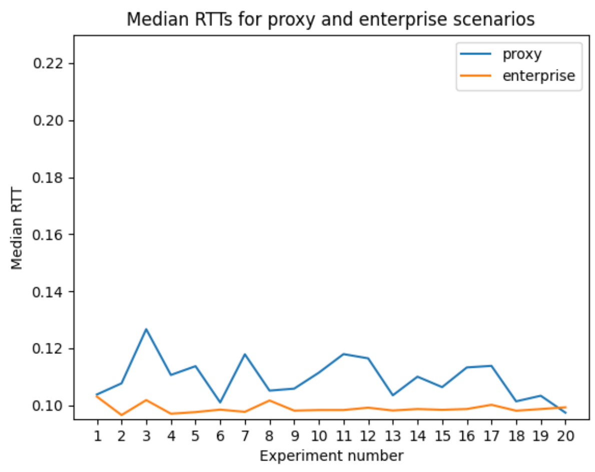

Figure 6.8: Median RTT per scenario (TCP-TLS method)

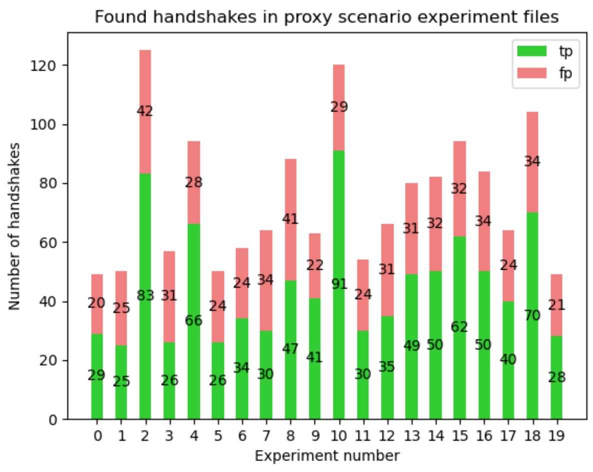

Figure 6.9: Number of true and false positive handshakes in proxy scenario (TCP-TLS method)

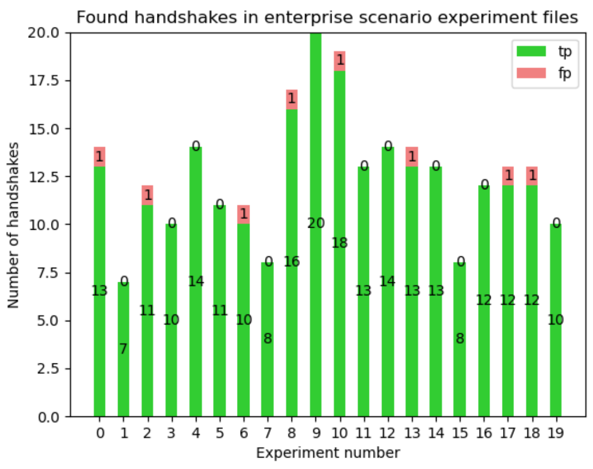

Figure 6.10: Number of true and false positive handshakes in enterprise scenario (TCP-TLS method)

addition of specific packet lengths greatly increases the precision of the method. We focus again on the proxy scenario as it is more indicative of real data than the enterprise scenario: as expected, the number of false positives greatly decreases, meaning the computed mean and median RTTs are closer to their ground truth values. While the number of found true positives per experiment also decreases compared to the previous method, the average precision increases to 0.62 for the proxy scenario.

Unlike our expectations, the higher precision of the TCP-TLS method seems not to affect the mean RTT. We speculate that extreme outliers are present, and seek to better understand our data by computing the mean and standard deviation of each experiment's RTTs.

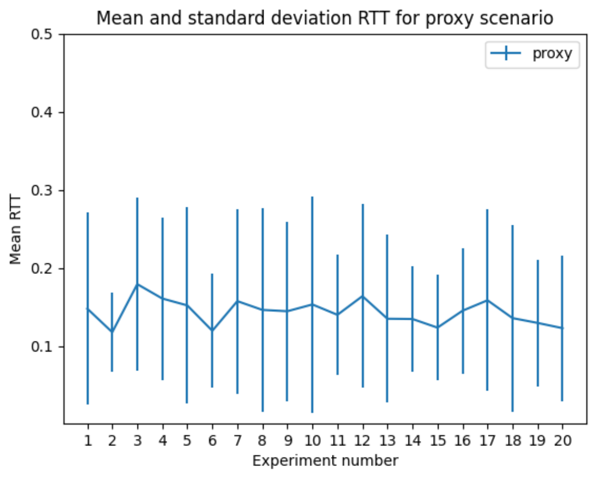

Figure 6.11: Mean and standard deviation of proxy scenario RTTs (TCP-TLS method)

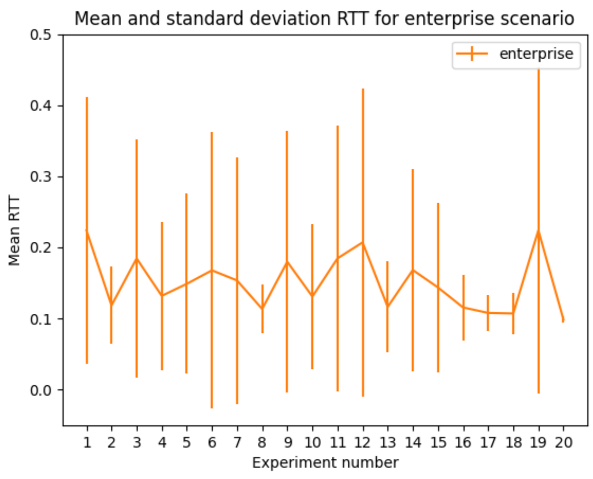

Figure 6.12: Mean and standard deviation of enterprise scenario RTTs (TCP-TLS method)

The difference between the means of the two scenarios is highlighted by Figures 6.11 and 6.12. Many experiments in the enterprise scenario exhibit large standard deviations, suggesting that the mean is not truly representative of the data. The lack of similar standard deviation values across experiments explains the variability of the enterprise mean RTTs seen in Figure 6.7. On the other hand, the proxy scenario experiments all show relatively uniform standard deviations.

The standard deviations suggest that outliers are present and influence the computed mean. We consider two ways to mitigate their effect: either we use the median as the classification metric, or we restrict the mean to only a portion of the data, excluding high and low value outliers using quantiles.

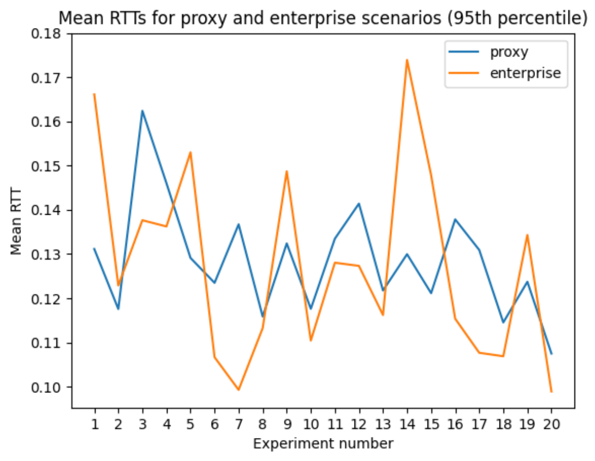

Figure 6.13: Mean RTT per scenario (95th percentile)

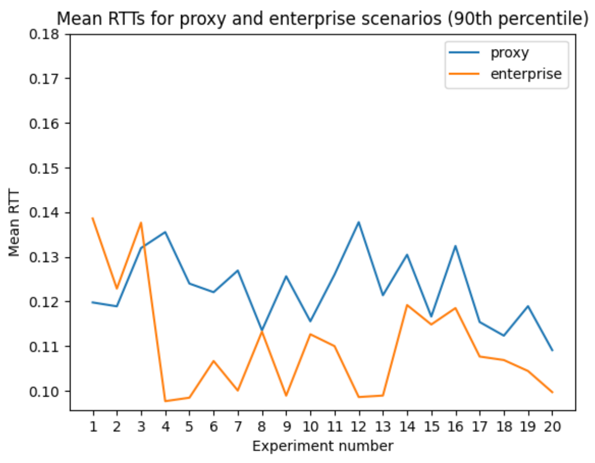

Figure 6.14: Mean RTT per scenario (90th percentile)

A significant difference between the means of the two scenarios can be observed when excluding values above the 85th quantile or below the 15th quantile (Figure 6.15), though a less clear distinction is also

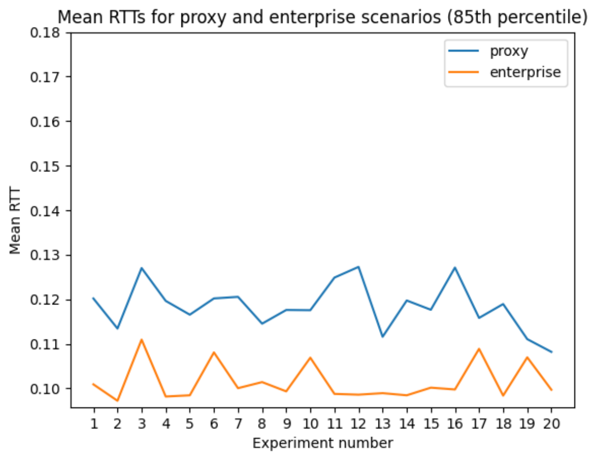

Figure 6.15: Mean RTT per scenario (85th percentile)

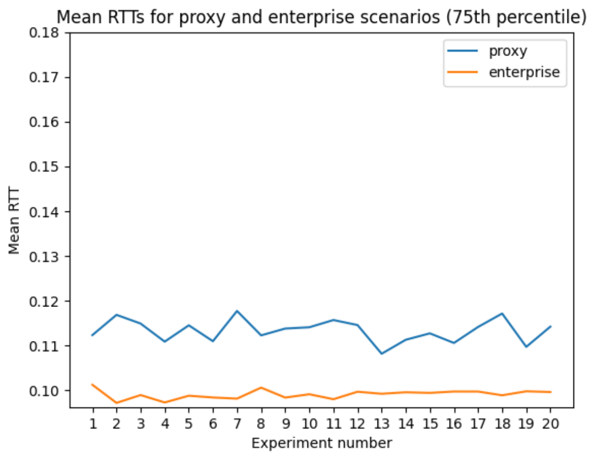

Figure 6.16: Mean RTT per scenario (75th percentile)

seen when using the 90th and 10th quantiles (Figure 6.14). These results imply that when removing outliers, the mean is a valuable metric for distinguishing the two scenarios.

We applied our two methods on the Docker generated data. In both cases, the mean RTTs of the two scenarios showed no clear distinction, while the proxy median RTTs were higher than the enterprise median RTTs. As expected, when using the TLS-only method the number of proxy false positives outnumber the true positives; we do not consider the TLS-only method viable for classification, as the computed RTTs are not indicative of the ground truth. The TCP-TLS method, on the other hand, greatly improved precision by placing more restrictions on handshake identification. Seeking to understand the RTTs better, we found that the standard deviations of RTTs per experiment were quite large. This indicated the presence of outliers that influence the mean, which we removed using quantiles.

Due to the large number of false positives found using either method, accounting for outliers when applying metrics is crucial. We found that the median is the most reliable classification metric. Applying the mean after removing outliers gives similar results, but may require excluding a large portion of the data. Nevertheless, these metrics show that the proxy RTT is larger than the enterprise RTT, which confirms our hypothesis.

7
Discussion

Our results show that latency is a distinguishing factor between business use and private use VPNs, and that an outside observer can detect the latency of a tunneled connection with no previous knowledge. Given flows from both scenarios, an attacker can determine which is which by comparing the mean and median RTT. To the adversary, the private-use VPN is the connection with a longer median RTT, as shown in Figure 6.7; said use-case will also exhibit a larger mean RTT, but only after outliers are eliminated.

7.1. Privacy implications

Adversaries can integrate our method into existing VPN-detecting applications, as it relies on knowing whether a connection uses a VPN. Similar to other VPN fingerprinting techniques, our algorithm can be thought of as a way for attackers to find and scope future targets.

Our findings have important ramifications for enterprises that use VPNs to connect remote workers. An attacker can now more easily identify encrypted connections carrying sensitive data, a valuable target to hackers. They can record the behavior of the VPN server to later target it in an attack, or connect to it and gain access to company resources.

For commercial VPN providers, this means that even more information than previously thought can be gleamed about their users from encrypted traffic. Our results reinforce that many claims of perfect obfuscation and privacy made by VPN companies are not true, as shown by previous research into VPN fingerprinting.

Additionally, the difference in RTT can be used to fine-tune censorship. Censors typically block VPN use, but may want to make exceptions for remote workers or businesses that benefit them. Integrating our algorithm into existing censorship algorithms would achieve this goal.

Finally, our stream handshake identification method computes the RTT of an encrypted VPN connection. Delay has been shown to be a powerful metric, used to fingerprint proxies and geolocate servers. For example, by identifying a private-use VPN conversation and computing its RTT, a censor could estimate the location of the webserver the user is connecting to, which can impact the work of journalists and reporters in censorship regimes.

7.2. Limitations

Our results show that an adversary can differentiate between private and business use VPN traffic simply by applying statistical analysis. At the same time, our approach has a number of limitations:

- Our classification approach requires samples from both scenarios. This means that if an attacker happens to only have data from one, some of the flows may be wrongly classified as the other scenario. We described additional heuristics an attacker can use to further differentiate the two scenarios, but there are certainly corner cases where they do not work.

- Our decision to use synthetic data greatly simplified our process. However, this generated data may not accurately reflect the real world use case. This is particularly an issue in the enterprise scenario, where requests are made to a very simplified website. The bare bones nature of the WordPress website may lead to simple TLS handshakes, which would affect the computed RTT. On the other hand, the fact that all requests are made to the same webserver is not a problem; the website is an abstraction for the various resources that an employee would access, all hosted on company servers and thus all showing similar RTTs.
- A secondary issue with our synthetic data concerns the artificial delay added using `tc`. While the value is based on previous data about VPN latency, it would be much more variable in a real life scenario. Additionally, `tc` seems to introduce a lot of TCP connectivity issues on our Docker networks; the generated data exhibits many 'Previous segment not captured' warnings in Wireshark. This can affect our identification of both OpenVPN packets and unencrypted TLS packets used to find the number of true positives, causing potential false negatives.
- Our algorithm relies on the presence of TLS handshakes. While it is a popular protocol, this means that our method is not applicable to non-TLS traffic, as the computed RTT will not be correct. Moreover, our RTT computation method leads to a relatively small sample size - computing the RTT of an unencrypted connection can be done with any two data packets, while we depend on the less frequently occurring handshake.

7.3. Future work

We discovered several targets for future research.

- Testing our method on real data is a top priority in order to most accurately assess its strengths and weaknesses and make improvements. Afterwards, integrating our algorithm into a deployable application would test its effectiveness as a tool used by adversaries.
- While our goal was to document the fingerprinting power of latency on VPN protocols, attention should be brought to potential ways to circumvent this attack. Developing and testing countermeasures improves the security of both commercial and business VPNs and keeps users safe from adversaries.
- Our method builds on top of the 3-gram approach detailed by Xue et al. [19]. Their paper shows that some 3-grams have more 'distinguishability' power than others, meaning that their presence in a flow increases the likelihood of it being a proxied connection more than other 3-grams. An extension to our algorithm could similarly assign weights to different handshake sequences, which might improve the precision of the TLS handshake identifying algorithm and thus compute a more accurate RTT.

8 Conclusion

We develop an algorithm that distinguishes business-use VPNs from private-use VPNs based on encrypted traffic. We generate and apply our algorithm to synthetic data generated with Docker. By exploiting the packet size uniformity of TCP and TLS handshakes, we compute the RTT between the client and endpoint behind the VPN. We expect that enterprise VPNs will exhibit a smaller RTT than commercial ones as a result of the small physical distance between the servers hosting the VPN and endpoint; this hypothesis is confirmed by our results, which show that the mean and median RTTs of the proxy experiments are larger than those of the enterprise experiments.

The outcomes of our experiments help answer our main research question, showing that latency can be used to discriminate between the two types of VPNs. This paper supports conclusions drawn by previous research into privacy and security: VPN protocols are vulnerable to various fingerprints, leading to adversaries being able to obtain important information about users from encrypted traffic. Additionally, our results reinforce the power of latency as a fingerprinting feature on encrypted data, especially concerning geolocating efforts.

Our method has a number of limitations that open avenues for future work. Perhaps most importantly, our handshake identification algorithm must be tested on real data to truly assess its effectiveness. We explored only one classification method; exploring other options like using machine learning models may give different results. Finally, developing protection against latency-based fingerprints like ours should concern both researchers and VPN companies. We urge commercial VPN providers and business VPN users to stay informed of state of the art VPN attacks and develop countermeasures.

References

[1] Sikha Bagui et al. “Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features”. In: *Journal of Cyber Security Technology* 1.2 (2017), pp. 108–126. DOI: 10.1080/23742917.2017.1321891.

[2] Kevin Borders and Atul Prakash. “Web tap: detecting covert web traffic”. In: *Proceedings of the 11th ACM Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2004, 110–120. ISBN: 1581139616. DOI: 10.1145/1030083.1030100.

[3] United States Census Bureau. *American Community Survey; Commuting Characteristics by Sex*. 2023. URL: https://data.census.gov/table/ACSST1Y2023.S0801.

[4] Brett Cruz. *2024 VPN Trends, Statistics, and Consumer Opinions*. https://www.security.org/resources/vpn-consumer-report-annual/. 2024.

[5] Eurostat. *Employed persons working from home as a percentage of the total employment, by sex, age and professional status (%)*. 2025. DOI: https://doi.org/10.2908/LFSA_EHOMP. URL: https://ec.europa.eu/eurostat/databrowser/view/lfsa_ehomp__custom_12158505/default/table?lang=en.

[6] ExpressVPN. *The VPN that just works*. 2025. URL: https://www.expressvpn.com/.

[7] Ping Gao et al. “VPN Traffic Classification Based on Payload Length Sequence”. In: *2020 International Conference on Networking and Network Applications (NaNA)*. 2020, pp. 241–247. DOI: 10.1109/NaNA51271.2020.00048.

[8] Arash Habibi Lashkari et al. “Characterization of Encrypted and VPN Traffic Using Time-Related Features”. In: Feb. 2016. DOI: 10.5220/0005740704070414.

[9] J. Han and D. Wren. *VPN Products Performance Benchmarks (Edition 1)*. https://www.passmark.com/reports/VPN_Products_Performance_Benchmarks_2021_Ed1.pdf. 2021.

[10] Arif Mohammed, Shanika Karunasekera, and Santosh Kulkarni. “GeoWeight: Internet host geolocation based on a probability model for latency measurements”. In: vol. 102. Jan. 2010, pp. 89–98.

[11] NordVPN. *The best VPN service for online security. Get started risk free*. 2025. URL: https://nordvpn.com/.

[12] OpenVPN. *Data Channel Crypto module*. 2025. URL: https://build.openvpn.net/doxygen/group__data__crypto.html.

[13] Kim Parker. *About a third of U.S. workers who can work from home now do so all the time*. https://www.pewresearch.org/short-reads/2023/03/30/about-a-third-of-us-workers-who-can-work-from-home-do-so-all-the-time/. 2024.

[14] Reethika Ramesh et al. “CalcuLatency: leveraging cross-layer network latency measurements to detect proxy-enabled abuse”. In: *Proceedings of the 33rd USENIX Conference on Security Symposium*. SEC ’24. Philadelphia, PA, USA: USENIX Association, 2024. ISBN: 978-1-939133-44-1.

[15] Tal Shapira and Yuval Shavitt. “FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition”. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2019, pp. 680–687. DOI: 10.1109/INFCOMW.2019.8845315.

[16] Aliza Vigderman and Gabe Turner. *VPN Speed Tests: VPN Speeds Compared*. https://www.security.org/vpn/speed-test/#latency. 2024.

[17] Xinyuan Wang, Douglas S. Reeves, and S. Felix Wu. “Inter-Packet Delay Based Correlation for Tracing Encrypted Connections through Stepping Stones”. In: *Computer Security — ESORICS 2002*. Ed. by Dieter Gollmann, Günther Karjoth, and Michael Waidner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 244–263. ISBN: 978-3-540-45853-1.

[18] Zachary Weinberg et al. “How to Catch when Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation”. In: *Proceedings of the Internet Measurement Conference 2018*. Association for Computing Machinery, 2018, 203–217. ISBN: 9781450356190.

[19] Diwen Xue et al. “Fingerprinting obfuscated proxy traffic with encapsulated TLS handshakes”. In: *Proceedings of the 33rd USENIX Conference on Security Symposium*. SEC ’24. Philadelphia, PA, USA: USENIX Association, 2024. ISBN: 978-1-939133-44-1.

[20] Diwen Xue et al. “OpenVPN is Open to VPN Fingerprinting”. In: *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 483–500. ISBN: 978-1-939133-31-1. URL: `https://www.usenix.org/conference/usenixsecurity22/presentation/xue-diwen`.