Impact of replacing TCP by QUIC in Tor on website fingerprinting resistance

Cyril H. Trap
4441338
Cyber Security Specialisation, MSc Computer Science
Faculty EEMCS, Delft University of Technology
Thesis defence: 24-07-2023
Supervised by Stefanie Roos
Thesis committee: Stefanie Roos, Georgios Smaragdakis, Jérémie Decouchant

Abstract—**Privacy is a human right, yet, people's behavior on the web is constantly tracked. Tor, an anonymity network, is an effective defence against tracking. However, Tor's multiplexing of logically independent data streams into a single TCP connection causes issues. Tor with QUIC has been implemented as an alternative with better performance but it has not been studied whether and by how much QUIC increases the vulnerability to timing-based attacks.**

The most threatening attacks are website fingerprinting attacks, which can track a Tor user by only controlling the guard node, first of the relays that forward traffic in Tor. In this work, Tor with QUIC is evaluated against website fingerprinting attacks with various levels of defences active. Without defences, Tor is vulnerable to website fingerprinting for both TCP and QUIC but the attacks are more effective on QUIC. On the positive side, defences against website fingerprinting remain effective for QUIC in that they decrease the effectiveness of the attack by a similar fraction as for TCP.

Index Terms—**Tor, QUIC, anonymity, privacy, fingerprinting, machine learning, network traffic analysis, anonymous communication**

I. Introduction

A. Motivation

For most people regular encryption is sufficient, ensuring that the content of their messages or business online cannot be viewed by others. In other words, others cannot see what message they are communicating, however others can still see with whom they are communicating. For most people that is not an issue, however for example for whistle blowers, oppressed citizens or journalists, the fact that they are communicating with non approved parties can have severe consequences. This is were Tor[1] comes into play, by hiding the communicating parties as well.

Tor becomes more secure to use when more people use it - since it becomes harder the trace individuals when there is more data/noise from other users. More people tend to use a program when it offers a faster and smoother experience. One of the current issues causing lag spikes for users is the head of line blocking problem in the TCP protocol. This can be resolved by replacing TCP with QUIC. It also promises to bring some other speed optimizations to the table. However the introduction of QUIC should not degrade Tor's ability to provide anonymity. Thus we should test QUIC for its capability to maintain the current level of anonymity that Tor can provide for its users.

B. What is Tor?

Tor aims to provide anonymity to its users by hiding which parties are communicating. This differs from regular encrypted internet traffic where only the messages are hidden while the sender and receiver are known. Tor achieves this anonymity by using multiple layers of encryption and sending traffic through a series of nodes. Each node decrypts the outer layer, revealing the next node and forwards the message to the next node. Until it arrives at the last node where after decryption a regular internet message is sent to its final destination. The current Tor standard uses 3 nodes in between the user and final destination. This is sufficient to protect the user from deanonymization: The first node only knows the sender (the user) and the second node, but the final destination is still encrypted and therefore incomprehensible. The second node knows the first node and third, but cannot see the original sender (i.e. the user) as this information is no longer present or the final destination as it is still encrypted. The third node knows the second and the final destination, but cannot know the original sender as this information was already gone. So none of the three nodes in between is aware of both the original sender and the final destination as long as they are not colluding.

Tor uses TCP connections between nodes to deliver data. The data is divided into fixed size packets, adding padding if required, which are then encrypted. These encrypted packets along packets from other connections are stored in a buffer until they can be sent out to the next node. One of the properties of TCP is in-order delivery of data. This means that if a packet is lost, transmission of other packets is halted until the missing packet has been sent again. Normally this is a good property to have in a connection. However in the case of Tor multiple logically independent streams (sometimes even from different users) are multiplexed into a single TCP connection. This means that when a packet is lost for a given datastream and/or user all others are halted and have to wait even though their own packets are still ordered correctly. This causes delays and unnecessarily slows down the Tor network. This problem is known as Head-of-line blocking[2].

C. What is QUIC?

QUIC (Quick UDP Internet Connections) is a transport layer network protocol that establishes multiplexed connections between endpoints over UDP. It aims to have a minimal handshake to enable low-latency connection establishment. Its connections are encrypted by default. The protocol is described in RFC9000[3].

Replacing TCP by QUIC would also introduce more efficient congestion control and flow control[4]. The most straightforward approach to replacing TCP by QUIC is to use it for individual connections in between nodes. However, some designs for replacement would allow end-to-end control, from the end user to the exit node across the nodes in between (i.e. coupling the individual connections in between nodes to share information regarding congestion and flow control). Unfortunately, these designs currently have flaws reducing user anonymity, which need to be fixed before the end-to-end designs can be used[5].

QUIC was introduced in 2012[6]. The process of becoming an internet standard was started in June 2015 with the submission of a draft to the IETF[1]. A working group was established in 2016. In May 2021 it was standardized with the release of the aforementioned RFCs.

D. Contribution

This work focuses on assessing the influence of replacing TCP by QUIC in Tor by evaluating and combining it with existing attacks and defences. This contrasts to previous work that would only evaluate the performance of QUIC relative to TCP[5] or the impact of individual attacks[7][8][9][10][11][12][13][14][15][16][17] and defences[18][19][20][21][22][23][24]. This is the first work to combine all three components into a single setup. To facilitate this, effort was undertaken to merge and to instrument the required codebases while maintaining their individual properties and abilities. Two different implementations of Tor over QUIC are considered. The setup is run numerous times for each required configuration to be able to produce statistically sound results. Some combinations of Tor over QUIC, attacks and defences resulted in unstable codebases that would crash, preventing the gathering of data from completing. In this occasion a patch was produced to stabilize the codebase enough to at least make to the end of the data gathering process. After the data collection had finished the data was checked for consistency and subsequently processed to provide insight. During the evaluation an intriguing difference between TCP and QUIC is discovered and is attempted to be attributed and/or explained by further investigation, albeit unfortunately inconclusive.

[1]https://mailarchive.ietf.org/arch/msg/i-d-announce/zSk53ClZRO6eSH4s5a7bQ5Jiuns/

II. Background

Tor is originally an acronym of The Onion Routing[25] network. Referring to multiple layers of encryption, each being peeled off at a node, much like the layers of an onion. It finds its origin as a US Navy research project but became a standalone project of an organization called "Tor Project"[25].

Mixnets and Proxies aim to fulfill a similar goal to that of Tor, but they have different trade-offs[26].

A. Mixnets

When comparing Tor with mixnets the main difference is preservation of order. Tor keeps the packets of individual streams in order during transportation and they are delivered in order. Concurrency and buffers may result in a couple of packets of different logical streams being reordered compared to packets of the other stream, but within their own stream the order stays fixed. Mixnets on the other hand rely on shuffling the order of packets to hide their origin and destination. For example, Bitcoin mixers use this construction to obfuscate the involved parties of transactions for money laundering purposes. The use of permutation means that in order to hide the origin of a particular message several other messages are required. This potentially causes delays as some time is spent waiting for enough messages. This property together with the computationally expensive public-key cryptography operations means that mixnets are suitable for applications that tolerate high latency (email for example), but not for applications requiring low latency (e.g. web browsing). Tor on the other hand can handle both.

B. Proxies

Proxies aim to protect the identity of a user from becoming known by the service they intend to use. It relays traffic thereby replacing the true origin of a request with itself. The more traffic is sent through a proxy the harder it becomes to trace back the original origin of requests. Proxies support low latency applications as there is no need for computationally expensive public-key cryptography operations. However, as all traffic is being routed through a single point, it has a single point of failure. This is a disadvantage from both a reliability perspective as well as a security perspective. Although, if a user decides to use a regular direct connection when a proxy is unavailable instead of waiting for it to come back online, then reliability becomes security as this user's identity is now at risk of being discovered. An honest mistake or other unintentional failure can result in complete unavailability of the proxy service. This can also be the result of an intentional targeted attack (e.g. DDoS[27]) as well. Probably even worse is a targeted attack where access to the actual machine running the proxy is gained, this means all information about who is using the proxy and where they are connecting to is compromised. Tor is considerably less vulnerable to such failures and attacks as there is no single point of failure or single entity holding all routing (or other sensitive) information to attack.

C. Connections

Tor uses TLS to encrypt the TCP connections between nodes. A single TCP connection is created between nodes. Multiple logical independent streams can be transported via this single connection. The full path of all 3 nodes is called a circuit and is changed every 10 minutes or at the user's request. The first node (aka entry guard) however is maintained for the duration of the session, since this lowers the chance of an attacker to perform a successful correlation attack[28].

D. Onion sites

An onion site can be used to protect the identity of the user and that of the server. This can be seen as starting the process of creating multiple layers of encryption via nodes from two different end nodes which then meet in the middle, at the so called rendezvous point. Websites serving their content in this way can be recognized by the distinctive .onion TLD and usually seemingly random alphanumeric domain names of 56 characters (in version 3 onion addresses)[29]. These are the result of a hash performed over the identifying properties of the server. Since some fields can be set at will, they can be used the generate partially human-readable hashes by brute forcing them until the desired prefix in the hash is found. Two real world examples, one with subdomain[2], and one without[3], can be found in the footnotes.

E. Website Fingerprinting

Website Fingerprinting is the practice of trying to deduce what website a user is visiting by looking for unique patterns in the encrypted and anonymized communication. Mostly the direction and size of packets are used to find these unique patterns. In the Tor network an adversary running an malicious entry node would be in the position to mount such an attack.

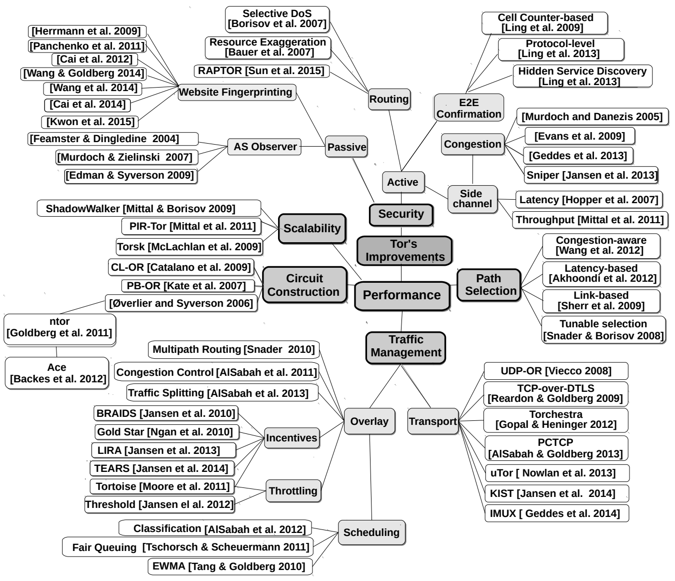
In evaluating website fingerprinting attacks on Tor there are two scenario's for the attacker. In the Closed world scenario a finite set of websites is fixed beforehand which a victim may visit. The challenge for the attacker lays in identifying which website in the set is being visited by the client. As this was not deemed very representative of how things would work in the real world the Open world scenario was introduced[30]. Here the victim is allowed to visit all available websites on the Internet. This naturally makes identifying which website the client visits much harder for an attacker. However, sometimes the requirement for the attacker is relaxed to only determining whether the client is visiting a website out of a fixed finite set[7]. This simulates a scenario where the client is facing an entity which is monitoring people for visiting a website on a list of "forbidden" websites. In this case, the websites not on this list are referred to as unmonitored websites.

[2]https://www.nytimesn7cgmftshazwhfgzm37qxb44r64ytbb2dj3x62d2lljsciiyd.onion/

[3]https://duckduckgogg42xjoc72x3sjasowoarfbgcmvfimaftt6twagswzczad.onion/

III. Related work

In 2016 AlSabah and Goldberg wrote an overview of the research on performance and security improvements for Tor[31]. Besides discussing the various papers, they provided a classification of research directions with accompanying papers and how these all relate to each other. Figure 1 shows this.



Tor's Improvements
Security
Performance
Passive
Active
Website Fingerprinting
[Herrmann et al. 2009]
[Panchenko et al. 2011]
[Cai et al. 2012]
[Wang & Goldberg 2014]
[Wang et al. 2014]
[Cai et al. 2014]
[Kwon et al. 2015]
AS Observer
[Feamster & Dingledine 2004]
[Murdoch & Zielinski 2007]
[Edman & Syverson 2009]
Routing
Selective DoS [Borisov et al. 2007]
Resource Exaggeration [Bauer et al. 2007]
RAPTOR [Sun et al. 2015]
E2E Confirmation
Cell Counter-based [Ling et al. 2009]
Protocol-level [Ling et al. 2013]
Hidden Service Discovery [Ling et al. 2013]
Congestion
[Murdoch and Danezis 2005]
[Evans et al. 2009]
[Geddes et al. 2013]
Sniper [Jansen et al. 2013]
Side channel
Latency [Hopper et al. 2007]
Throughput [Mittal et al. 2011]
Scalability
ShadowWalker [Mittal & Borisov 2009]
PIR-Tor [Mittal et al. 2011]
Torsk [McLachlan et al. 2009]
Circuit Construction
CL-OR [Catalano et al. 2009]
PB-OR [Kate et al. 2007]
[Øverlier and Syverson 2006]
ntor [Goldberg et al. 2011]
Ace [Backes et al. 2012]
Path Selection
Congestion-aware [Wang et al. 2012]
Latency-based [Akhoondi et al. 2012]
Link-based [Sherr et al. 2009]
Tunable selection [Snader & Borisov 2008]
Traffic Management
Overlay
Multipath Routing [Snader 2010]
Congestion Control [AlSabah et al. 2011]
Traffic Splitting [AlSabah et al. 2013]
Incentives
BRAIDS [Jansen et al. 2010]
Gold Star [Ngan et al. 2010]
LIRA [Jansen et al. 2013]
TEARS [Jansen et al. 2014]
Throttling
Tortoise [Moore et al. 2011]
Threshold [Jansen el al. 2012]
Scheduling
Classification [AlSabah et al. 2012]
Fair Queuing [Tschorsch & Scheuermann 2011]
EWMA [Tang & Goldberg 2010]
Transport
UDP-OR [Viecco 2008]
TCP-over-DTLS [Reardon & Goldberg 2009]
Torchestra [Gopal & Heninger 2012]
PCTCP [AlSabah & Goldberg 2013]
uTor [Nowlan et al. 2013]
KIST [Jansen et al. 2014]
IMUX [Geddes et al. 2014]

Fig. 1: An exact copy of the mind map figure from the AlSabah and Goldberg survey[31] for reference.

Although QUIC itself was already around before the publication of the survey, the idea of Tor over QUIC was not. If Tor over QUIC was to be classified into the topology shown by figure 1, it would be a leaf under Tor's Improvements → Performance → Traffic Management → Transport.

A. Attacks

Website fingerprinting attacks already have a leaf in the topology from figure 1: Tor's Improvements → Security → Passive → Website Fingerprinting. Website fingerprinting attacks are prevalent among the currently most relevant attacks. As a result, the following enumeration starts off with 5 website fingerprinting attacks. For attacks the best possible performance is not required for a comparison between TCP and QUIC, as long as attacks can be applied equally well to both they will be helpful in the comparison.

1) DeepFingerprinting: The DeepFingerprinting[7] (DF) attack leverages a Convolutional Neural Network to classify to which website unique patterns in a captured sequence belong. It has near flawless performance against Tor without defences with 98% accuracy. The attack still performs well when faced with lightweight defences, such as WTF-PAD[19] and Walkie-Talkie[20]. As threat model DF assumes a local passive attacker, for example an eavesdropper on the Wi-Fi network, a local system administrator or the Internet Service Provider. The traffic is intercepted between the Tor client and the entry node of the Tor network.

2) Var-CNN: Although Var-CNN[8] performs better than DF, this mainly manifests settings where there is little training data available. This enables attackers to perform this attack sooner as less time is required for collection data. Also, if for some reason the attacker is forced to attack with less data than planned, the chance of still successfully identifying websites is increased. Since for this work the amount of data collected can be fully controlled, a low data scenario can be mitigated by collecting enough data for the performance difference to become negligible. DF requires data with a lower dimension (i.e. a lower number of different features) than Var-CNN to be collected and does not require the manual extraction of features. These benefits of DF outweigh the slightly inferior performance (i.e. 1 or 2 percent points on an accuracy of around 98%)[4].

3) SDAE, AWF, k-NN, CUMUL and k-FP: When Stacked Denoising Autoencoders[9] (SDAE), Automated Website Fingerprinting[10] (AWF), k-Nearest Neighbors[11] (k-NN), a Support Vector Machine with feature set based on a cumulative sum[12] (CUMUL) and k-Fingerprinting[13] (K-FP) are being compared with DF in an open world scenario, DF still performs best in terms of both True positive rate and False positive rate regardless of how many unmonitored samples are provided in the training dataset.

However, when comparing DF amongst the aforementioned attacks in a closed world scenario when faced with the following defences; BuFLO[21], Tamaraw[22], WTF-PAD and Walkie-Talkie, it is outperformed by a couple of percent points when BuFLO or Tamaraw are active. Unfortunately, these 2 defences have a severe overhead in terms of bandwidth and latency, respectively 246%, 137% for BuFLO and 328%, 242% for Tamaraw. This impact is so severe that it would hinder adaption in Tor since it would impact performance for its users heavily. If we only consider the other 2 remaining defences; WTF-PAD (Overhead: 64%, 0%) and Walkie-Talkie (Overhead: 31%, 34%) when evaluating attacks, then DF clearly emerges as winner when ranking them using their achieved classification accuracy with the 2 defences active. For more details on the individual attacks and defences the reader is referred to Table 3 and figure 6, along with sections 3.1 and 3.2, in the DF paper[7].

The authors of k-fingerprinting performed an systematic feature selection on the Wang et al. dataset[11]. They concluded that the number of incoming packets is the most informative feature. Which is not surprising given that this is probably the reason why deep learning[32][33] techniques trained only on packet direction sequences perform well. However, a consequence of this extensive feature selection procedure is that more work is required to apply this attack.

4) Triplet Fingerprinting: The Triplet Fingerprinting attack[14] aims to reduce the minimal size of the dataset required to achieve proper classification accuracy. It does not improve the state-of-the-art when considering only raw accuracy. Since we control the data gathering process we can ensure there is enough data, thereby losing the advantage this attack had. Note that this is the same goal Var-CNN, the authors acknowledge that they share this goal and that it is unfortunate that no comparison is present due to its paper already being written when Var-CNN was presented.

5) Tik-Tok: Tik-Tok[15] is build upon DF. DF uses packet direction, encoded as a sequence of -1's and +1's. Tik-Tok uses that as well, but adds timing information. The timing information is represented in a sequence of the same length and is multiplied with the direction sequence (i.e effectively scaling each entry). This use of extra timing information besides packet direction allows for modest improvement in accuracy. For example, against WTF-PAD in closed world setting the classification error goes down by 2.5 percent points. In open world setting against WTF-PAD when tuned for precision it performs 2 percent point better. However in the remaining open world settings it performs equally or worse. So in overall in terms of performance is it quite similar to DF. The mathematics behind the individual timing features is not particularly complicated. However, it still an extra step that needs to be performed when extracting the training sequences from the raw captured packets. Therefore it is easier to use DF, since the data processing is easier and the resulting accuracy is very comparable to the Tik-Tok attack. The Tik-Tok attack is most relevant in establishing that, when done correctly, timing information can be used to perform website fingerprinting quite well and under the right circumstances is not inferior to using directional information.

6) Flow correlation: Correlation attacks would in figure 1 fall under Tor's Improvements $\rightarrow$ Security $\rightarrow$ Passive $\rightarrow$ AS Observer. Flow correlation assumes a more powerful adversary than required for website fingerprinting. The idea behind this attacks is to look for traffic flows that show a high correlation, as with a higher correlation there is a higher probability that the traffic flows carry the same items. This adversary is capable of passively listening to Tor traffic at multiple locations in order to determine if there exists traffic that enters at a particular point and leaves at another, thereby identifying the parties communicating.

a) DeepCorr: DeepCorr[16] uses deep learning[32][33] to predict which network flows are correlated. Attacks from previous work on flow correlation were deemed not practical on a large-scale Tor network, due to the use of generic techniques from statistics. DeepCorr significantly increases the correlation accuracy leveraging machine learning while using substantially less observations compared to previous work, it is capable of achieving 96% accuracy after collecting roughly 900 packets each traffic flow (state-of-the-art using classical techniques yielded 4% here). Due to the use of machine learning DeepCorr uses a custom correlation function specific to Tor and this use case. In contrast to other attacks utilizing machine learning, such as fingerprinting attacks, DeepCorr does not need training data of target websites in order to be able to identify them as this information comes from a successfully deanonymized Tor connection.

[4]Based on the Var-CNN paper, figure 8 (the amount of Monitored Traces was 2000 for most, although 2 measurements had 1000 traces)

Although DeepCorr presents a rather interesting topic for comparison we did not use it due to time constraints.

b) AttCorr: AttCorr[17] is similar to DeepCorr, however it is based on a different deep learning model. Unfortunately, the paper is really scarce in providing details. It contains only the description of the model and a comparison with DeepCorr. It yields similar results in terms of accuracy, but computes slightly faster.

This attack was omitted from comparison due to the similarity to DeepCorr, but less informative paper and time constraints.

B. Defences

For defence there exist several options against website fingerprinting. As the topology in figure 1 does not distinguish between attacks and defences, but rather focuses on theme, all defences against website fingerprinting end up together with their attacks under Tor's Improvements $\rightarrow$ Security $\rightarrow$ Passive $\rightarrow$ Website Fingerprinting.

1) TrafficSliver: TrafficSliver[18] is a defence against website fingerprinting and tries to break the unique patterns found in the sequence of encrypted packets by splitting the traffic over multiple entry nodes, each former their own unique path via other nodes towards the destination. These extra paths are called subcircuits and form an additional parallel connection besides the regular path which unmodified Tor would create, aka the circuit. The latter is now called the main circuit. Due to this splitting, an individual node is no longer in the position to view the recognizable patterns since it now only sees fractions of the original sequence with large gaps in between. TrafficSliver has several modes for splitting the traffic, each resulting in different patterns. The modes vary in their ability to thwart attacks. The main goal of TrafficSliver is to defend against malicious entry nodes performing Website Fingerprinting. The TrafficSliver defence results in a slightly increased latency and a insignificant overhead in terms of extra bandwidth consumption, thereby exceeding the capabilities of all existing website fingerprinting defences. It seems therefore the most likely to be implemented into Tor, hence it deems most representative as chosen defence for this work.

TrafficSliver has two implementations: one working at the network layer and another for the application layer. The network layer version is the more interesting of the two for this research as it allows for full control over individual packets, it can achieve better performance and works with all kinds of traffic. The network layer version requires adjustments to the Tor source code. Where this is not required for the application layer version, as it works fully independent. It forms a proxy between the browser and 3 entry nodes, each connected to a separate independent circuit. The current implementation exploits some HTTP-specific features to enhance its splitting strategy and therefore only works web traffic.

Being a newer phenomenon, TrafficSliver was unknown when figure 1 was constructed. Categorizing it results in 3 applicable locations. Since it is an defence against website fingerprinting, as mentioned it can be placed under Tor's Improvements $\rightarrow$ Security $\rightarrow$ Passive $\rightarrow$ Website Fingerprinting. But it has two more valid locations since the splitting take places at either the network layer, yielding the Tor's Improvements $\rightarrow$ Performance $\rightarrow$ Traffic Management $\rightarrow$ Transport, or the application layer, resulting in Tor's Improvements $\rightarrow$ Performance $\rightarrow$ Traffic Management $\rightarrow$ Overlay.

Note: The amount of entry nodes actually used when defending is either as configured (3 by default) or 1 more. The latter occurs when the desired extra entry node for the construction of a new subcircuit is are already used as part of another circuit and thus cannot be used anymore as an entry node for a new subcircuit as this would result in using that node twice. Therefore another new node is chosen as entry node for the subcircuit.

2) Mockingbird: Machine learning[32][33] can be used as part of an attack, in that case does it learn recognize the situation of a succesful attack. However, it can also be used to learn the situation where another machine learning algorithm is struggling. This information can then be used to alter the current situation, to lower the chances of a successful attack and thereby turning the situation in favour of the defender. This principle is called adversarial machine learning. Altering the situation towards favourable conditions for the defender, means adjusting the input samples that the attacking has to classify. These adjusted inputs are called adversarial samples.

Mockingbird[23] proposes to use adversarial samples as defence against website fingerprinting attacks. The new idea is here to fight machine learning with machine learning instead of a classical algorithm, as would be the case with TrafficSliver for example. The Mockingbird defence is evaluated against DF and Var-CNN, the two most potent deep learning attacks. It halves the accuracy of these state-of-the-art WF attacks depending on the scenario using a 58% overhead in bandwidth for full-duplex traffic. For half duplex traffic the bandwidth overhead is increased to 62% or 70% depending on how exactly the pool of adversarial samples was filled. The higher overhead case provides even better results pushing the accuracy down to 38% for DF and 30% for Var-CNN. Walkie-Talkie is comparably effective against DF and around 10 percent points worse against Var-CNN. WTF-PAD cannot defend well against both, leaving DF at 86% and Var-CNN at 90% accuracy.

The Mockingbird defence is also evaluated against some non-deep machine learning attacks and is really successful. It reduces the accuracy of CUMUL and k-FP to a third or better depending on the adversarial pool used. The results on k-NN are even better reducing its accuracy to a sixth or better. However, WTF-PAD and Walkie-Talkie are also successful against these 3 attacks.

The authors also compared Mockingbird with WTF-PAD and Walkie-Talkie in a Top-2 scenario (is the correct website amongst the 2 most probable sites according to the classifier/attack) against DF and Var-CNN. Interestingly, in this case WTF-PAD and Walkie-Talkie lose their ability to defend as the accuracy shoots up to at least 92%. Mockingbird performs much better here with accuracy rising to around 50% except for one particular scenario (again related to how the adversarial sample pool is filled) where it rises to around 70%.

To deploy Mockingbird in a real-world setting a couple of issues would need to be addressed:

- Like Walkie-Talkie, Mockingbird requires the maintenance of a database with relatively recent burst sequences to fill its adversarial sample pool.
- In order to be able to use padding on bursts, they need to be identified. Doing this reliably in a live setting can be a tricky problem.
- Like the deep learning attacks it tries to defend against, Mockingbird itself also requires a powerful GPU to be able to execute it in a reasonable amount of time. This would increase the computational requirements for running Tor significantly. However, luckily running ML models on low-end hardware is an active research field and the results look promising. The authors did not pursue to implement techniques from this direction themselves.

Mockingbird looked promising to also use in our comparison of Tor over QUIC versus vanilla Tor, unfortunately due to time constraints it is not present. Mockingbird was not chosen as defence in favour of TrafficSliver as it has a larger overhead in terms of bandwidth and requires more computational power to run. The latter is a direct result of Mockingbird's design, since it uses machine learning which is much heavier compared to the, even for a classical algorithm, lightweight TrafficSliver. On top of that, TrafficSliver also performs better when defending against DF.

3) BiMorphing: The BiMorphing[24] defence is not evaluated against the most potent attacks, namely DF and Var-CNN. The two attacks we can compare it with to TrafficSliver are CUMUL and k-NN. In a closed world setting BiMorphing reduces the accuracy of the attack to 19.64% for CUMUL and 12.93% for k-NN, while TrafficSliver manages 4.63% for CUMUL and 3.15%. In open world setting only CUMUL is available for both defences. BiMorphing supplies a single point with a true positive rate of 86.91% and false positive rate of 19.64%, while TrafficSliver provides a full ROC curve. Plotting the single point of BiMorphing on the ROC graph, it still lays far away from the performance of TrafficSliver (which makes CUMUL perform marginally better than random guessing).

Clearly TrafficSliver is superior in terms of defensive capabilities to BiMorphing and therefore we do not include the BiMorphing defence in our comparison. Also BiMorphing did not come with a reference to its codebase, while TrafficSliver did.

C. UDP-OR

UDP-OR is located in the topology from figure 1 under Tor's Improvements $\rightarrow$ Performance $\rightarrow$ Traffic Management $\rightarrow$ Transport. UDP-OR[34] in 2008 was the first work to introduce the idea of Tor over UDP. It shows that fairness was improved and latency did not change significantly. The authors acknowledge that they still need to address several issues and mention that they did leave many open questions regarding anonymity of its users. So security is not tested in any way. Note that website fingerprinting attacks were not even around as they first appeared in this paper[35] from 2009.

D. uTor

Like UDP-OR, uTor is also located in the topology from figure 1 under Tor's Improvements $\rightarrow$ Performance $\rightarrow$ Traffic Management $\rightarrow$ Transport. To resolve the head-of-line blocking problem in Tor the authors of uTor[36] proposed to replace TCP and TLS by their unordered counterparts, respectively uTCP and uTLS. Full backwards compatibility, modular applicability and no weakening of the privacy and anonymity of users were its design requirements. Full backwards compatibility would ensure Tor's user base adapting easily. The modular applicability would help with a gradual adaption, as some links could use it while older links would still work. This also ensures the change would be transparent to the user client. The authors remark that uTLS does not introduce security loss over TLS, however that they left a formal and thorough security analysis to future work. Since uTCP and uTLS are wire compatible with TCP and TLS, minimal changes to the codebase are required (95 extra lines of code, an increment of 0.001%). A simple version number check suffices to distinguish between regular Tor and uTor. In a test setup a significant reduction of latency is shown when using uTor compared to regular Tor.

E. Torchestra

Staying in the same location in figure 1 for Torchestra: Tor's Improvements $\rightarrow$ Performance $\rightarrow$ Traffic Management $\rightarrow$ Transport. The main idea behind Torchestra is splitting heavy and lightweight traffic between parallel TCP connections. The focus of this work lays on creating performance improvements. The concept might recall TrafficSliver to mind. The difference is that TrafficSliver is using the splitting of traffic as a defensive measure, while Torchestra is using it to gain speed by eliminating delays. Thus Torchestra is focused on performance while TrafficSliver aims for security, consequently they are found on two completely different branches in the topology from figure 1.

IV. Implementations of Tor over QUIC

Two implementations of Tor over QUIC were found. One developed by a fellow student at the TU Delft and one developed by researchers at Qatar University. The TU Delft version was eventually chosen because it was based on a newer Tor version and it had good support options since the original author could easily be contacted.

A. Developed by a fellow TU student

This implementation uses the Quiche[37] library developed by CloudFlare to provide QUIC support. This is an open-source implementation of the QUIC and HTTP/3 protocols written in Rust. The core code of Tor is augmented with an implementation of a Tor channel over QUIC. It touches relatively little existing code and is mostly coded parallel to their TCP counterparts. In the cases where existing Tor source code needs to be altered, a simple boolean switch is used to pick between a classical channel using TCP or the new option with a channel over QUIC. A few extra commands needed to be introduced to the Tor protocol. The source code reveals that a few shortcuts had to be taken because the author was running out of time[5]. Thus the code contains some potential security flaws, which could easily be studied and exploited by an adversary as this implementation is open-source. It had some memory leaks which prevented it from running for extended periods of time. To fix this, a patch was submitted to the original author to mitigate some of them, but a few are still remaining.

This implementation comes with documentation in the form of a MSc Thesis[38] and is based on Tor version 0.4.5, commit: https://github.com/torproject/tor/commit/fe49a2474d54ea9443e71bd738b189b0de8cd58e

B. Developed by a Qatar University group

This implementation uses the simple-quic library, which came in a `.zip` file, for QUIC support. This in turn leverages the boringssl library developed by Google. However as stated in the `README.md` there is no intention for general use even though it is open-source and that it may exhibibit braking changes without any notice. This is probably why the received `.zip` file also contained a fixed copy of boringssl. Unfortunately this setup is tougher to work with and required more time to get to a compiling state than the other implementation. Getting to this state even took using old Ubuntu distributions (16.04.7) as it would only compile on there. Porting it to more up-to-date releases yielded varying success and this meant that continuing with this implementation was not possible as the other tools required for the complete setup of this research run on more modern Ubuntu distributions.

However, it is running on older releases and it successfully completes chutney network tests. It's based on Tor version 0.3.3, commit: https://github.com/torproject/tor/commit/72e1f19249abae96530aa78a1a1441242c9b2239

V. Experimental setup / Methodology

To assess what influence the QUIC proposals have on existing defences and attacks, their codebases are merged one-by-one. The goal is to create a table where the accuracy of each combination of an attack and a defence can be looked up for the different QUIC implementations. From this table the influence of the QUIC implementations on the success of attacks and defences can be determined.

Merging the codebases proved of varying difficulty. It was of real help that TrafficSliver was developed as a Tor module, so much of its code could coexist with the rest without merge conflicts. The option to preemptively create circuits under TrafficSliver needed to be disabled to prevent it from interfering with measurements. For the Qatar implementation the merging process meant going through the codebase to update API calls to a newer version. Unfortunately, this implementation proved to be not compatible with the other tools required and was therefore abandoned. For the student implementation the process was smoother and required fixing the version of a build dependency (i.e. `cmake = "=0.1.45"`). Additionally, a small edit to the codebase in order for it to expose the port numbers used for building Tor circuits was needed, so the data gathering process could listen to the correct ports. After running this implementation for a longer period of time, it turned out that it was severely leaking memory. To remediate this, a patch was developed together with running it with a periodic restart it was stable enough. It can be useful to disable to `info` level logging for Tor as this prevents large log files from collecting on the computer under long and frequent use.

After the codebase of the used defence is merged with the codebase of the QUIC implementation, the resulting Tor version is compiled into a binary. This binary in combination with a bash script is used to collect traces of the top 100 used websites (based on the Alexa top 2019[39], as this provides a representative list of websites users would visit in the real world)[6]. cURL[40] is used to collect the traces using a SOCKS5 proxy that is exposed by Tor. The Tor network is run locally using chutney[41] since this allows for Tor versions with breaking modifications (i.e. changes that are incompatible with the existing tor network or protocol) to be run as well[7]. To collect the traces tcpdump[42] is used to capture the traffic between the client and the guard node(s). This emulates a scenario where a malicious entry node would try to identify the website/service a client is visiting. Since not all combinations of codebases run equally stable for an extended period of time, after a round of traces is collected, that is all 100 sites once, the local Tor network is torn down and a new one is built up for the next round. As the problems causing the instability rise only after running the network for longer periods, the traces are not influenced when the network is run for a short time and rebuild before the instability issues have a chance to manifest themselves.

After running a sufficient amount of rounds to have gathered enough, the data is filtered. Sufficient is defined by at least as much data points as were used by the authors of the original paper. The filtering checks ports, number of connections made and that it is a connection between the client and an entry node. It also performs some logical checks to ensure internal consistency:

- the amount of rounds is as expected,
- for each round we have correctly identified which ports were used,
- cURL returned status code 0,
- no timeout occurred,

[5]See, e.g.: `src/feature/relay/relay_handshake.c:187`

[6]See appendix A for the complete list

[7]NetMirage was also considered for this. However, because https://crysp.uwaterloo.ca/software/netmirage warns that "It is not quite ready for production use yet.", chutney was preferred.

accuracy DeepFingerprinting attack closed world scenario 100 sites				
	No defence	TrafficSliver (3RR)*	TrafficSliver (5RR)	TrafficSliver (5BWR)
QUIC	0.8415 (0.0030)	0.6957 (0.0022)	0.7079 (0.0024)	0.3020 (0.0084)
TCP	0.6307 (0.0010)	0.5627 (0.0031)		0.2145 (0.0103)

TABLE I: RR = Round Robin, BWR = Batched Weighted Random, * = default configuration of TrafficSliver, # = number of parallel circuits — All values in the table are the average of 10 samples. The sample standard deviation is given between brackets. All numbers are rounded to 4 decimals. Since TrafficSliver for 5RR performs very similar to 3RR for QUIC, the TCP results were omitted to save time.

- the number of connections detected is consistent with the configuration of the active defence.

The filtered data is then written to a python pickle file in the form of a sequence of 1's and -1's indicating the direction of the packets.

These sequences are the input data for the attacks. The sequences are loaded and formatted as expected by the specific attack (padded to a fixed length for example) and the attack is run. Since state-of-the-art attacks are all machine learning based, this means splitting the sequences in a training set, test set and validation set[32][33]. The machine learning model is trained and tested to establish the accuracy of the attack for the given defence and used transport protocol.

The crawling does not require specifically powerful machines and was performed on a desktop with an i7-4770 (max 3.90GHz) and an old laptop with an i3-M330 (max 2.13GHz), both running Ubuntu 18.04.6 LTS. Since all evaluated attacks leverage machine learning a powerful machine was used with a i9-12900KF (max 6.70GHz) with a NVIDIA GeForce RTX 3090 running Ubuntu 20.04.4 LTS with NVIDIA drivers 510.73.05 and CUDA version 11.6 installed. To ease the use of installation of the required software packages for machine learning the tensorflow docker image (`tensorflow/tensorflow:latest-gpu`) was used, with the following alias added to easily start it:

```
alias dgpu='sudo docker run -u $(id -u):\
  $(id -g) -v /home/user/:/home/user/ \
  --gpus all -it --rm tensorflow/tensor\
  flow:latest-gpu bash'
```

VI. Evaluation

To be able to select which defences and attacks are active, the following options are available: To switch QUIC on and off: Use `QUIC 0` / `QUIC 1` in the Tor configuration file, which are located here when using Chutney: `chutney/torrc_templates/common.i`. To switch TrafficSliver on (default when merged) and off run the configuration script of Tor:

```
./configure --disable-asciidoc \
  --disable-module-split
```

The script is located in the main directory of Tor: `tor/`, after running the configuration script, Tor needs to be rebuild (and reinstalled if required) in order for the changes to become active. To select which mode is used to defend by TrafficSliver the following options are available in the Tor configuration file: To select the amount of entry nodes to be used for splitting: `SplitSubcircuits 4` To select the splitting algorithm: `SplitStrategy BATCHED_WEIGHTED_RANDOM`

Despite the python random seed being fixed to 0 (with `random.seed(0)`) and being ran on the same input data, the DeepFingerprinting attack exhibits some non-deterministic behaviour. To mitigate this effect it was ran 10 times in all configurations and the average was taken to fill out the values in table I.

Recall that TrafficSliver can be configured to use various algorithms for splitting packets across circuits. Using more parallel circuits or using a stronger algorithm results in a stronger defence. In table I the number of parallel circuits in use by the TrafficSliver defence is denoted by the number preceding the letters. The letters come from abbreviating the names of the splitting algorithm used: RR = Round Robin and BWR = Batched Weight Random. See the original paper[18] for more details.

Looking at table I, it shows that the current QUIC implementation is somewhat more vulnerable to the DeepFingerprinting attack in comparison with the regular vanilla Tor over TCP. However, the same pattern of reduction in accuracy when faced with increasingly stronger defence configurations[8] can be seen for QUIC as with TCP, suggesting that the TrafficSliver defence might be equally applicable to QUIC as to TCP. Since QUIC is more vulnerable without defence this suggests that if this baseline can be lowered, there is a possibility that when done so, it will achieve similar results when defended with TrafficSliver.

(QUIC - TCP) / TCP		
NoDef	3RR	5BWR
0.3343	0.2364	0.4080

TABLE II: Different transport protocol, same defence configuration, this delta is an indicator for how well TrafficSliver under QUIC is able to match the performance of TrafficSliver under TCP.

To access what impact the different transport protocols have on the ability of TrafficSliver to defend, the change in accuracy of the DeepFingerprinting attack is calculated as shown in table II. Reviewing these numbers shows that this QUIC implementation performs between 24% and 41% worse

[8]Thus going from left to right in table I

than the regular Tor over TCP. The most inferior performance occurs under the 5BWR defence configuration.

	QUIC	TCP
(3RR - NoDef) / NoDef	-0.1733	-0.1078
(5BWR - 3RR) / 3RR	-0.5659	-0.6188

TABLE III: Same transport, different defence config, if this yields the same delta, it indicates that TrafficSliver is equally effective for both transport protocols.

To evaluate what impact the different defence configurations have on the ability of TrafficSliver to protect, the change in accuracy of the DeepFingerprinting attacks is calculated as shown in table III. Given that DeepFingerprinting achieves about the same relative accuracy when faced with different defence setups, it shows that the different defence configurations retain their relative strength compared to each other under both transport protocols. Thus the strongest defence setup for TCP is also the strongest one for QUIC. Furthermore, ordering the defence configurations by strength yields to same outcome for both QUIC and TCP.

When looking at the the relative changes in table II, it can be noted that 5BWR yields inferior performance under QUIC. This called for a more in-depth look in order to find out what causes this. It is especially interesting since TrafficSliver only uses direction and order of packets as input information. Which raises the question: how come there is a difference between QUIC and TCP? Both should relay the same data in the setting in which this experiment is run. The use of cURL ensures that only a single resources is fetched at a time, so there are no concurrent streams fetching other resources that could interfere. It also rules out race conditions in buffers and the like. Preemptive circuits are disabled, the only information relayed internally by the Tor network brought up by chutney is exchanged before the measurements are started. Note that this all provides the attack with the best chance of identifying the visited website correctly. A possible explanation that could not be ruled out by the environment created for this setup, is a difference coming from non-data packets. These could be i.e. control packets or other overhead specific to the used transport protocol.

To allow for more insight into the origin of the discovered difference, plots of the sequence lengths (figures 2, 6 and 10), number of packets per direction (figures 3, 7 and 11) and the Levenshtein distance between sequences (figures 4, 8, 12, 5, 9 and 13) were created. These 4 plots were made for 3 defence configurations: no defence, 3RR and 5BWR.

A. *No defence configuration*

Starting off with no defence this gives the following plots as shown in figures 2, 3, 4 and 5.

When looking at figure 2, where the number of times a sequence of a certain length is encountered under both QUIC and TCP is plotted, one may note that QUIC uses longer sequences of packets to transfer the same information than TCP uses. Additionally, one could observe that for sequences

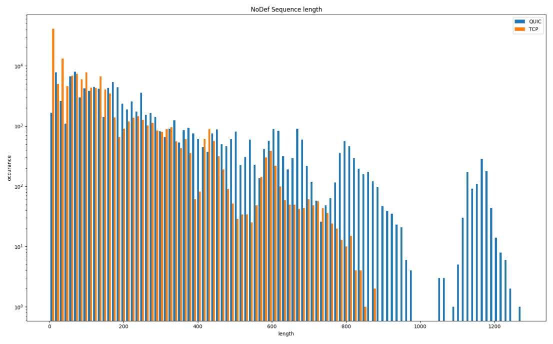


Fig. 2: Distribution of sequence lengths with no defence. Note the logarithmic scale used for the vertical axis.

of equal length, except the shortest sequences, QUIC uses more sequences to exchange the same data.

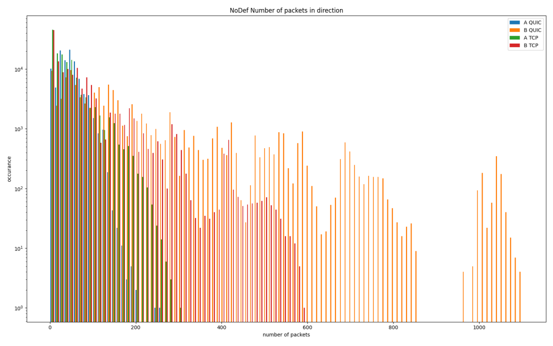


Fig. 3: Distribution of number of packets per direction with no defence. Note the logarithmic scale used for the vertical axis.

To see if these extra packets QUIC is sending are going in a specific direction the plot in figure 3 was created. As can be seen the effect is predominantly present in packets going from the webserver to the client. In the other direction it cannot be seen and if anything might be the opposite case. One could argue that under TCP more than 130 packets going from the client to the webserver occurred more often than under QUIC.

Since we are looking for an indication as to why TrafficSliver in the 5BWR configuration is performing relatively subpar and DeepFingerprinting is performing well when using QUIC, we need to compare the sequences that are being sent under QUIC and TCP for similarity. Recall that the sequences only consist of 1's and -1's, encoding the direction of packets and that there is no other information as all packets are encrypted and of equal size (there is timing information, but that is not used by the DeepFingerprinting attack). This makes it possible to use the Levenshtein distance as a similarity metric. To analyse the similarity of the sequences 3 aspects are investigated: the internal similarity of QUIC sequences, the

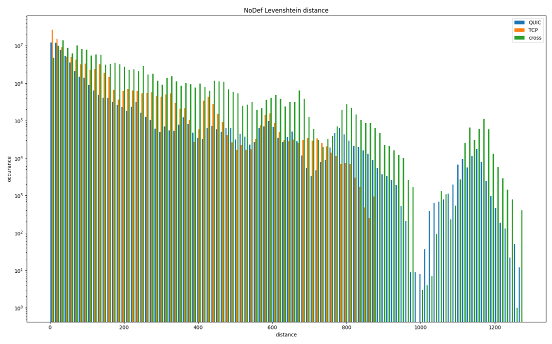


Fig. 4: Distribution of internal Levenshtein distances with no defence. Note the logarithmic scale used for the vertical axis.

internal similairity of TCP sequences and the cross similarity, see figure 4.

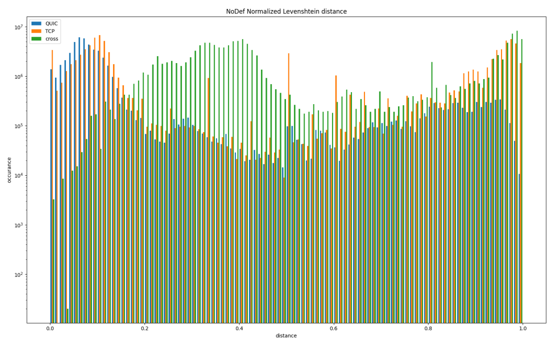


Fig. 5: Distribution of internal normalized Levenshtein distances with no defence. Normalized means that the Levenshtein distance is divided by the combined length of sequences, this ensures all inputs are mapped to [0,1]. Note the logarithmic scale used for the vertical axis.

Using the Levenshtein distance directly allows for 2 sequences to be compared, however we would like to be able to also use the distances to be able to say something about how distances relate to each other. Take for example a pair of sequences of length 20 of which 5 differ and another pair of length 100 of which 10 differ. This would give a distance of 5 for the shorter pair and 10 for the longer pair, however the longer sequences are more similar since 90% of the sequences is equal as opposed to 75% for the shorter ones. To be able to compare differences in distances for short and long sequences fairly they need to be normalized, this is achieved by dividing the Levenshtein distance by the combined length of the sequences. This ensures that all distances are mapped to the [0,1] interval and can be compared fairly, see figure 5.

B. *3RR configuration*

Moving on to the default TrafficSliver defence configuration, 3 subcircuits and using Round Robin to distribute the packets among them. This configuration gives these plots as shown in figures 6, 7, 8 and 9.

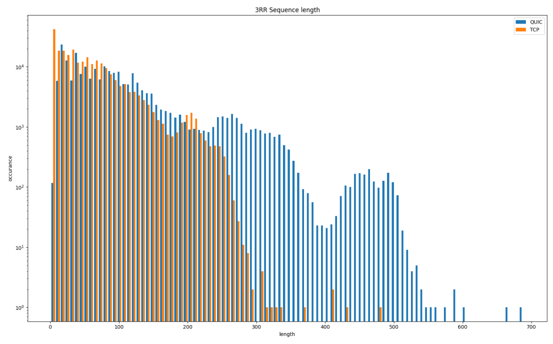


Fig. 6: Distribution of sequence lengths under the 3RR defence configuration. Note the logarithmic scale used for the vertical axis.

When the 3RR defence is introduced note that the sequences become a lot shorter. This is to be expected as each node now only encounters one-third of the packets. Were the horizontal axis of figure 2 extends beyond lengths of 1200 packets, the axis of figure 6 maxes out below 700 packets and almost all sequences are actually shorter than 550 packets. This holds for both QUIC and TCP sequences, although overall TCP sequences are shorter than QUIC sequences. It is difficult to say whether the introduction of the 3RR defence enlarged this difference between sequence lengths under QUIC and TCP.

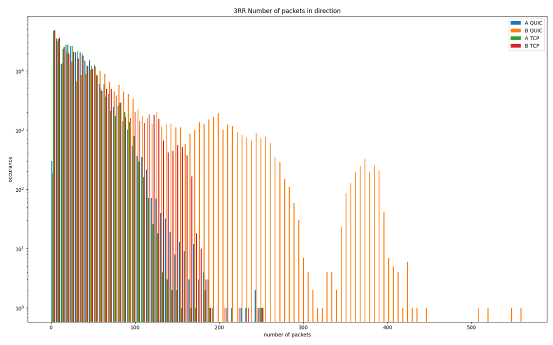


Fig. 7: Distribution of number of packets per direction under the 3RR defence configuration. Note the logarithmic scale used for the vertical axis.

With the 3RR defence active we still clearly see in figure 7 that there are much more packets from the webserver to the client under QUIC than under TCP. However the small effect in the opposite direction is no longer present as the distributions of packets going from the client to the webserver under QUIC and TCP are almost identical.

When comparing distances between no defence and 3RR it stands out that overall distances are a lot smaller after the

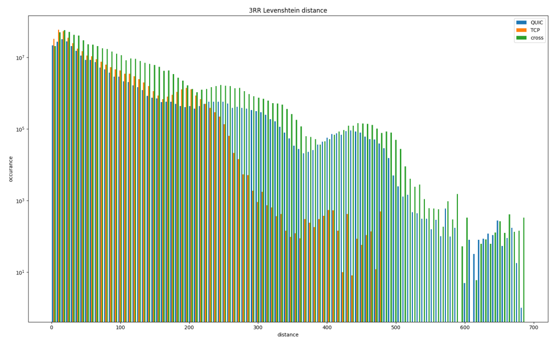


Fig. 8: Distribution of internal Levenshtein distances under the 3RR defence configuration. Note the logarithmic scale used for the vertical axis.

introduction of 3RR. This makes sense as sequence lengths also went down going from no defence to 3RR. They roughly halved as can be seen when comparing figures 2 and 6. This translates nicely into distances also roughly halving going from figure 4 to figure 8. Note that overall TCP has smaller distances than QUIC, this was already noticeable in figure 4, but in figure 8 one can observe that under TCP the number of occurrences of distances steeply decreases around 300 and remains significantly lower above that. This drop under TCP was not visible with no defence present.

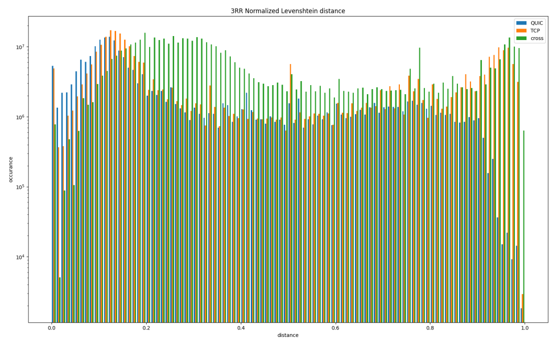


Fig. 9: Distribution of internal normalized Levenshtein distances under the 3RR defence configuration. Normalized means that the Levenshtein distance is divided by the combined length of sequences, this ensures all inputs are mapped to [0,1]. Note the logarithmic scale used for the vertical axis.

Looking back at figure 5. After adjusting for differences in sequence length by normalizing them, QUIC and TCP have very similar distributions. TCP has more occurrences of normalized distances closer to 1 and some peaks of around 1 order of magnitude larger here and there. Moving on to cross distances. Near 0 the number of occurrences is relatively low, so QUIC and TCP sequences do not really resemble each other in terms of packet order. However a local peak in between 0.2 and 0.5 is visible. Distances near 1 are more frequent again.

Introducing the 3RR defence results in figure 9. In between 0.1 and 0.9 both TCP and QUIC went up in the number of absolute occurrences, which is to be expected as splitting sequences in three parts creates more sequences. Both went up equally so they still have similar distributions. Note that the effect of TCP having more occurrences closer to 1 is more explicit, also because QUIC drops off quicker than under the no defence configuration. TCP is no longer showing peaks, except for one smaller peak around 0.5.

C. 5BWR configuration

And lastly the strongest TrafficSliver defence configuration, using 5 subcircuits and the Batched Weighted Random strategy to decide which subcircuit packets should travel on. This configuration results in following plots.

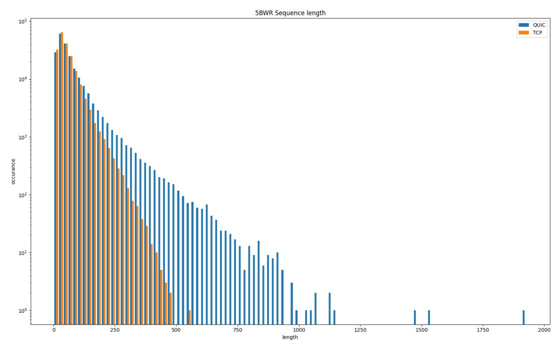


Fig. 10: Distribution of sequence lengths under the 5BWR defence configuration. Note the logarithmic scale used for the vertical axis.

One aspect that immediately stands out when looking at figure 10 is the triangular shape under both QUIC and TCP. Also, under QUIC the base of this triangle seems to be twice as wide as under TCP. Looking back at figure 6 under 3RR there also is a hint of QUIC having the same shape as TCP but with a twice as wide base, however looking back at no defence in figure 2 this effect cannot be seen.

Going from no defence in figure 2 to 3RR in figure 6 sequences became a lot shorter. Going from 3RR to 5BWR this cannot be seen. This is counter-intuitive as packets are now distributed over 5 nodes instead of 3 (3RR) or 1 (no defence). As given by the shape of the triangles, TCP sequences are still overall shorter than QUIC sequences. This observation was also the case under the 3RR configuration. Note that given the geometric nature of the distributions in figure 10, one can estimate the difference to be QUIC using twice the amount of packets TCP needs to convey the same information.

In figure 11 there is practically always the same number of packets going from the client to the webserver under QUIC and TCP, just like was visible under 3RR in figure 7. In the opposite direction the two distinct triangle shapes from figure 10 can be seen again, from which can be concluded that the

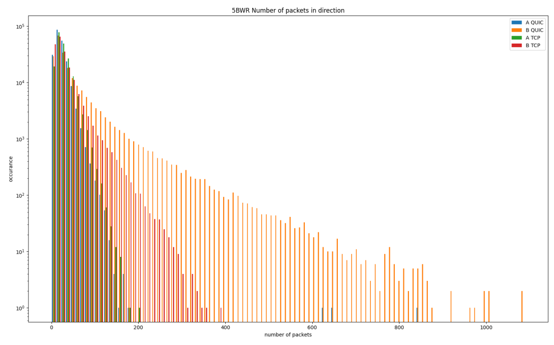


Fig. 11: Distribution of number of packets per direction under the 5BWR defence configuration. Note the logarithmic scale used for the vertical axis.

factor 2 in width of their base is fully caused by packets going in that direction, that is from the webserver to the client. Note that more packets in this direction being sent by QUIC was also the case under the 3RR configuration as can be seen in figure 7.

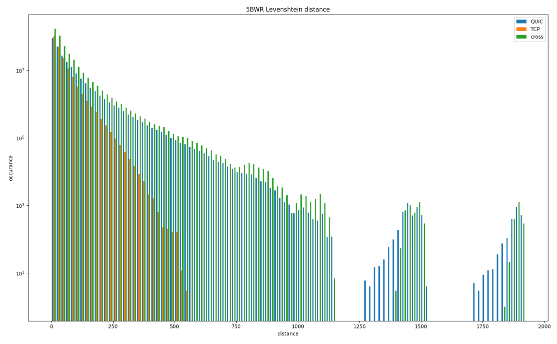


Fig. 12: Distribution of internal Levenshtein distances under the 5BWR defence configuration. Note the logarithmic scale used for the vertical axis.

When comparing distances between 3RR in figure 8 and 5BWR in figure 12 one can observe that under TCP distances follow roughly the same distribution. This is consistent with the observation that sequence lengths did not go down either when going from 3RR to 5BWR, so it makes sense that their distances did not change much.

Overall TCP remains to have smaller distances than QUIC, like was the case under 3RR. One could argue that this aspect is even more profound under 5BWR. The two triangular shapes are distinguishable here once again, although QUIC has two isolated peaks next to the main triangular shape.

Note that the horizontal axis of figure 12 is considerably longer than in figure 8. For QUIC the distribution remains similar to 3RR as well, although this only applies to distances up to 700. This raises the question where these extra distances beyond 700 come from under QUIC. Looking back at figures 7 and 11 with attention to the domain of their horizontal axes it can be noted that under both QUIC and TCP there are more packets coming from the webserver to the client when using the 5BWR configuration than when using 3RR. The other direction does not show much difference. This would explain the extra larger distances since longer sequences are being sent under QUIC. However these larger sequences can also be observed under TCP when looking at figures 7 and 11 while they do not translate into additional peaks of larger distances in figure 12.

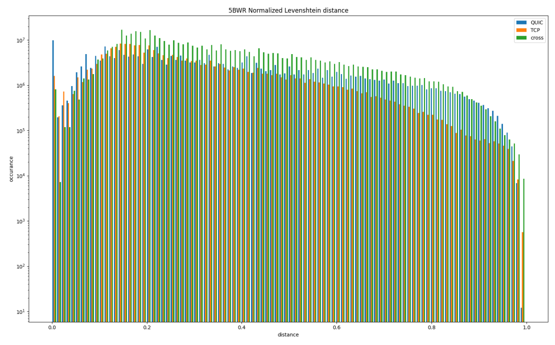


Fig. 13: Distribution of internal normalized Levenshtein distances under the 5BWR defence configuration. Normalized means that the Levenshtein distance is divided by the combined length of sequences, this ensures all inputs are mapped to [0,1]. Note the logarithmic scale used for the vertical axis.

In figure 13 TCP and QUIC have very similar distributions, although under QUIC distances larger than 0.5 occur consistently more often than under TCP. The largest difference occurs around 0.8, being almost 1 order of magnitude in size. Since TCP and QUIC have such resembling distributions it follows logically that the cross distribution also mostly takes their shape.

An interesting observation is that there is no longer a peak visible in the distribution of distances near 1 under TCP, which was the case with no defence and 3RR active.

D. *Hypothesis*

Note that in the figures above the distribution under QUIC is similar in shape to TCP but it is twice as broad at the base. This implies that QUIC uses longer sequences than TCP to relay the same information. This observation is also backed up by looking at the average sequence lengths: 678 packets for QUIC vs 251 for TCP (no defence), 245 QUIC vs 116 TCP (3RR) and 158 QUIC vs 107 TCP (5BWR). Note that these averages are from the 10 websites[9] with the largest averages sequence length instead of the full list of 100 websites.

The difference was deemed by the authors too large to be explained by sending some extra control packets only.

[9]In descending order of average size with indices 88, 18, 52, 7, 1, 27, 67, 34, 61, 16 as listed in appendix A.

To ensure that the difference was not caused by different congestion control algorithms used by the transport protocols it was verified that they both used the "cubic" congestion control protocol. For TCP using the following command `sysctl net.ipv4.tcp_congestion_control`[10]. And for QUIC it was verified using the source code, specifically line 425 of file `src/core/or/channelquic.c`[11].

Based on the observation a hypothesis was formulated that the difference could be explained by the fact that the maximum number of data bytes (MTU[43]) that a single TCP packet (1460) can transport is larger than that of a QUIC packet (1350). Regardless of the used transport protocol the exit node forms a TCP connection with the webserver. Thus large returning TCP packets would need to be split into 2 QUIC packets in order to be able to be transported across the Tor network. This would result in longer sequences for QUIC compared to TCP and up to a factor 2 higher number of packets going from the webserver to the client.

In order to test this hypothesis an experiment was set up where the network traffic between the exit node and the webserver was forced through a link with a significantly reduced MTU (632 bytes). This was accomplished by using 2 virtual machines which had (`sudo ip link set enp3s8 mtu 632`) set on their network interface cards (NIC) connection to the internal network. One could verify that the MTU of the link is indeed reduced by pinging the other side with a large packet with the Don't Fragment bit set in its IP header (`ping 1.1.1.1 -M do -s 1300`). One VM was acting as the connection to the outside world by having an additional NIC (Bridged Adapter) to be able to connect to the websites and IP forwarding enabled (

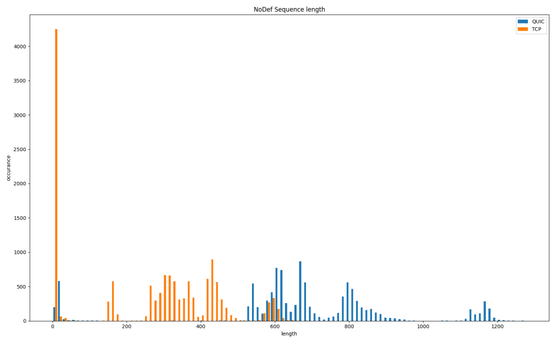
```
sysctl net.ipv4.ip_forward = 1 & sudo \
  iptables -t nat -A POSTROUTING -o \
  enp0s3 -j MASQUERADE
```

). The second VM, running the Tor network and collecting the data, used the aforementioned VM as its gateway and thereby all its internet traffic was forced through the reduced MTU link. This should eliminate the difference in sequences as packets have to be split for both transport protocols or for neither.

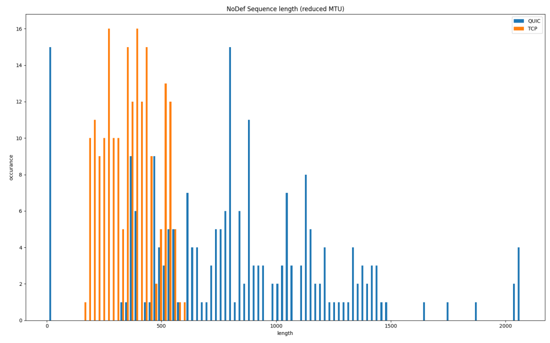
The following figures will show a comparison between base conditions and the setup described above with a reduced MTU. The figures showing base conditions use the same data collected for the main experiment filtered on the 10 selected websites. Over 20 runs were performed with 50 rounds each times 10 websites gives at least 10000 samples. In case of the reduced MTU setup, a single run was performed with 20 rounds for 10 websites resulting in 200 samples. This is substantially lower than for the base conditions. This means that there exists a large difference in scale on the vertical axis between figures of base conditions and those with a reduced

[10]Returning `net.ipv4.tcp_congestion_control = cubic`.

[11]Line 425: `quiche_config_set_cc_algorithm(config, QUICHE_CC_CUBIC);`



(a) Base conditions



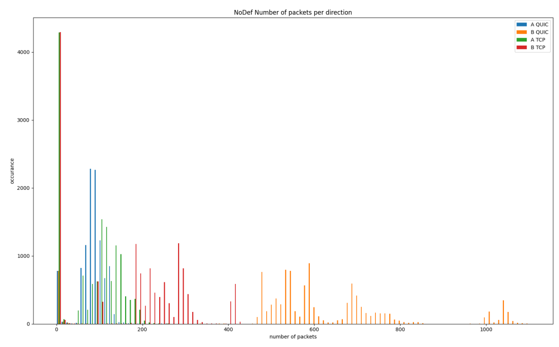
(b) Reduced MTU

Fig. 14: Distribution of sequence lengths for 10 largest websites with no defence.

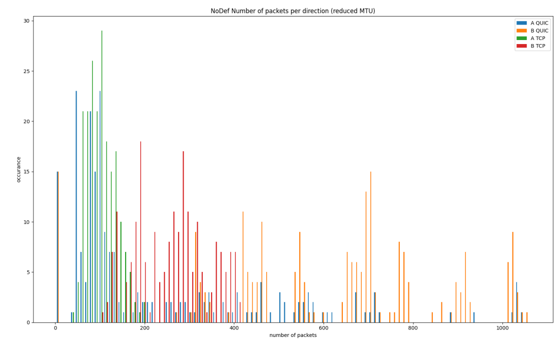
MTU. However, to test the hypothesis it is sufficient to see if the shape of the figures change when going from base conditions to a reduced MTU setup. Collecting fewer samples means the required time is reduced significantly, while the 10 largest websites will show the biggest change.

1) No defence configuration: Comparing figure 14a with base conditions and figure 14b with the reduced MTU setup active one can note the following: TCP shows a peak near 0 under base conditions, however with a reduced MTU it disappeared. The range of sequence lengths under TCP remains roughly the same, from a little under 200 to about 600 packets. The range of length under QUIC spreads out a bit when faced with a reduced MTU, going from 3 regions 0-50, 500-950 and 1100-1200 under base conditions to 0-50 and 300-1500 with a couple individual peaks up to 2100.

Splitting the sequences up in packets per direction results in figure 15a for base conditions and figure 15b with a reduced MTU. Under base conditions TCP shows peaks near 0 of nearly identical height in both directions, about twice as large as the second largest peak. This indicates a lot of very short sequences being send between the client and server with an equal amount of packets in both directions. These sequences of packets are probably carrying very little payload (as one would expect longer sequences and of asymmetrical proportions in

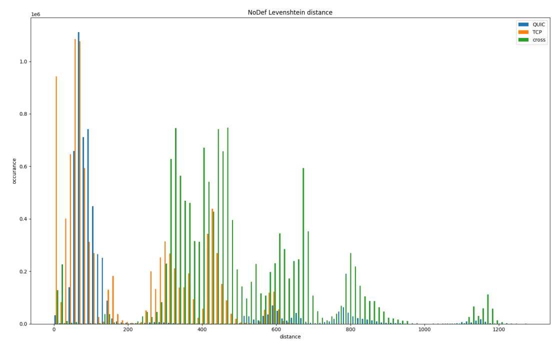


(a) Base conditions

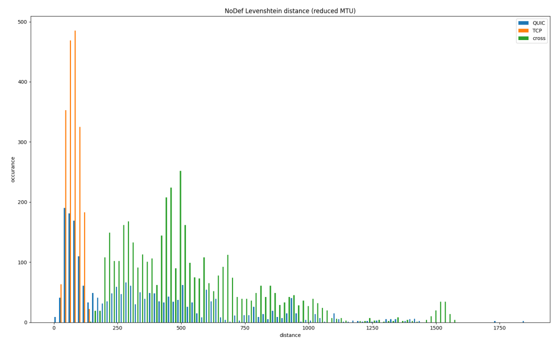


(b) Reduced MTU

Fig. 15: Distribution of number of packets per direction for 10 largest websites with no defence.



(a) Base conditions



(b) Reduced MTU

Fig. 16: Distribution of internal Levenshtein distances for 10 largest websites with no defence.

that case). Besides the peaks near 0, with TCP the number of packets in the client-to-server direction ranges from 50 to a bit over 200. Going from server-to-client it varies from 100 to a bit over 400 packets. Under QUIC in client-to-server direction the number of packets ranges from 0 to 200, with a peak centered around 100. In the opposite direction QUIC shows a very low number of packets, i.e. the single peak near 0, or a number of packets ranging from 250 to 850, or a little over a 1000 packets.

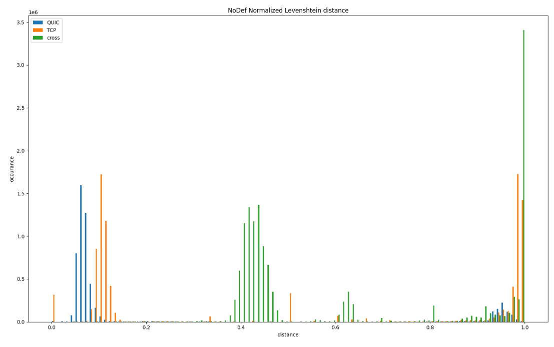
In figure 15b the single peak at 0 for TCP again completely disappeared under a reduced MTU, coherent with the observation made above on sequence lengths in figures 14a and 14b. The number of packets in client-to-server direction with TCP ranges from a bit under 50 to 200. In opposite direction it varies from 100 to a bit over 400. For QUIC in client-to-server direction it ranges from 0 to 1000, although there is a peak around 100, numbers above 200 become less frequent and numbers above 600 are sparse. In opposite direction QUIC has either very little packets, the single peak near 0, or the number of packets ranges from 300 to over 1000 packets.

Measuring the Levenshtein distance in base conditions provides figure 16a and in case of a reduced MTU figure 16b. TCP has distances ranging from 0 to a little over 600 under base conditions. QUIC shows a much broader spectrum of distances going from 0 to 1200, however it has a large peak a little under 100, while the rest is not even 10% the height of this peak.

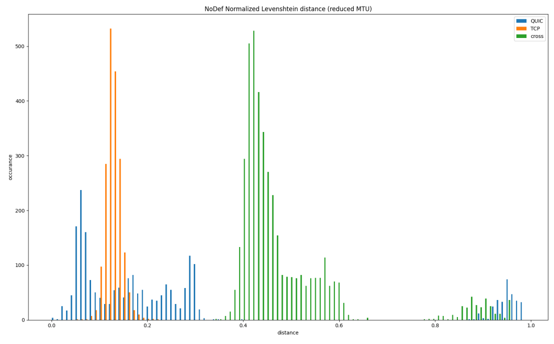
When reducing the MTU, figure 16b, the distances under TCP range from 0 to under 200, a significant reduction from base conditions. The range of distances under QUIC is not severely altered, a few of the observed distances are now large than the maximum 1200 under base conditions. However, the large peak around 100 has collapsed, although still being there it is now only a little over twice as large as the remaining distances outside of this peak.

Interestingly, both QUIC and TCP are altered due to a reduced MTU, but in a completely different dimension: TCP is altered in range, while QUIC is altered in height.

Normalizing the Levenshtein distance results in figure 17a for base conditions and figure 17b with a reduced MTU. In base conditions both TCP and QUIC have a peak between 0 and 0.2 and a few distances outside of that range. TCP has very close to 1 a smaller peak, which is absent under QUIC. Cross distances show a peak at 1 meaning that there are differences between QUIC and TCP packet sequences that are complete opposites (so they do not contain repetitive substrings and cannot easily be shifted to resemble to others) or, perhaps more likely, long sequences in either one and very short sequences

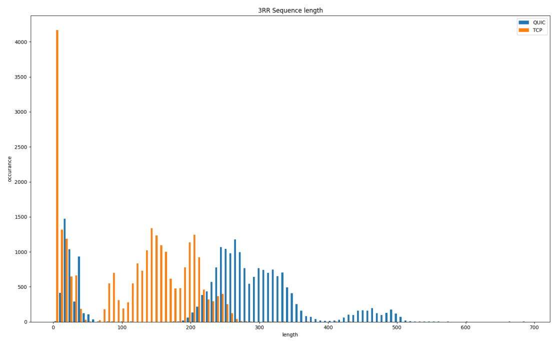


(a) Base conditions

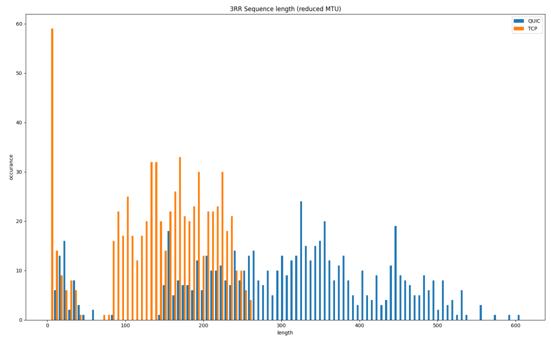


(b) Reduced MTU

Fig. 17: Distribution of internal normalized Levenshtein distances for 10 largest websites with no defence. Normalized means that the Levenshtein distance is divided by the combined length of sequences, this ensures all inputs are mapped to [0,1].



(a) Base conditions



(b) Reduced MTU

Fig. 18: Distributions of sequence lengths for 10 largest websites under the 3RR defence configuration.

in the other.

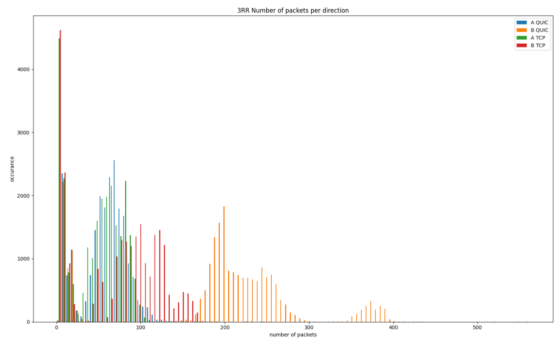
After introducing a reduced MTU, figure 17b, under TCP the peak between 0 and 0.2 reoccurs. However, the smaller peak very close to 1 no longer exists. In fact, the former peak is now the only data in the whole TCP distribution. When looking at QUIC one can note that its peak between 0 and 0.2 is reduced in height and spread around from 0 to a little over 0.3. The rest of QUIC's distribution remained roughly intact when faced with a reduced MTU. The cross distribution also remained similar except for the peak at 1, which has completely vanished when reducing the MTU.

2) 3RR configuration: With the 3RR defence configuration active, sequence lengths under base conditions in figure 18a are compared with figure 18b with a reduced MTU. Like with no defence, TCP has a peak near 0 under base conditions. Unlike no defence, the peak near 0 remains when reducing the MTU. QUIC does not show this peak in both cases.

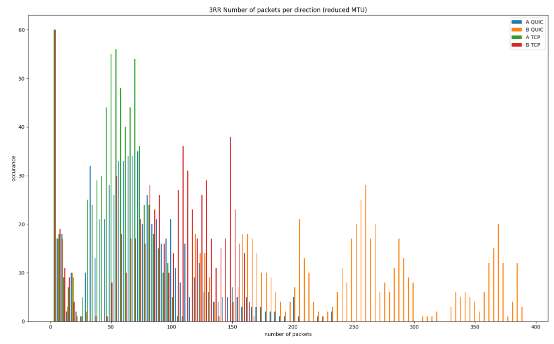
The range of sequence lengths under TCP is very similar under both conditions, ranging from 0 to a little under 300. Also, both show a little dip in number of occurrences of sequence lengths around 50 (base conditions show a few longer sequences, however so little that in the graph these have a height of a single pixel). The range of sequence lengths under QUIC goes from 0 to a bit over 500 with a gap around 100. It remains the same when faced with a reduced MTU (base conditions show a few longer sequences here as well, also having a height of one pixel in the graph).

Figure 19a shows the number of packets per direction under base conditions, while figure 19b shows it with a reduced MTU. Regarding base conditions, TCP shows, similar to no defence, peaks near 0 again of nearly identical height in both directions, also about twice as large as second largest peak. In the direction client-to-server the number of packets for TCP ranges from 0 to a bit over 100. In the other direction it varies from 0 to a bit under 200. For QUIC in the client-to-server direction the number of packets ranges from 0 to a bit over 100 with a few larger numbers of up to 250. The number of packets in opposite direction varies from 0 to 400 with gaps from 50 to 150 and from 300 to 350. A few larger numbers are seen here as well, going up to 550.

In case of a reduced MTU, figure 19b, and unlike with no defence the peak near 0 under TCP does not disappear with the 3RR defence configuration. However, it decreased in height, being now roughly equal to the other larger peaks. The number of packets for TCP in the client-to-server direction ranges from

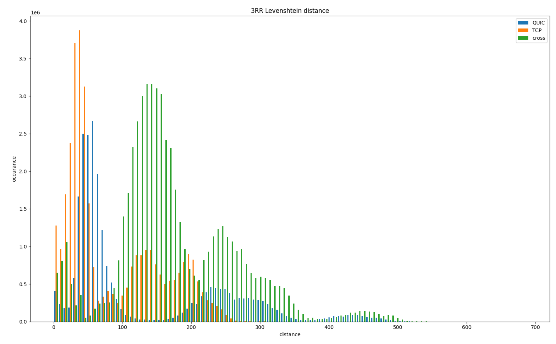


(a) Base conditions

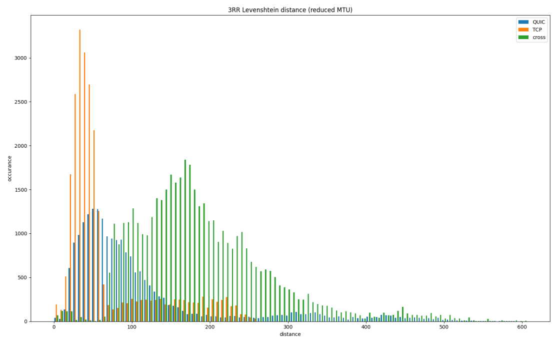


(b) Reduced MTU

Fig. 19: Distribution of number of packets per direction for 10 largest websites under the 3RR defence configuration.



(a) Base conditions



(b) Reduced MTU

Fig. 20: Distribution of internal Levenshtein distances for 10 largest websites under the 3RR defence configuration.

0 to a bit over 100. In the server-to-client direction TCP varies from 0 to 175 with a gap from 25 to 50. The number of packets for QUIC ranges from 0 to 225 in the client-to-server direction. In opposite direction it ranges from 0 to under 400, whit a gap from under 50 to over 100. The overall shapes of distributions under QUIC and TCP are similar to their counterparts under base conditions.

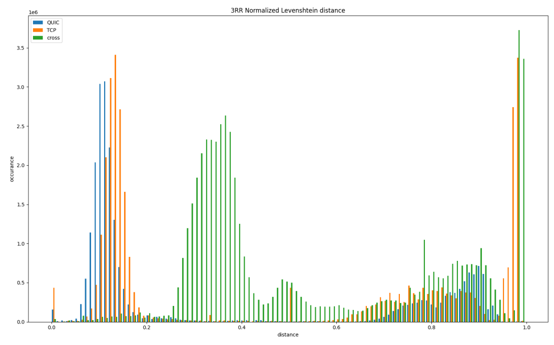
The internal Levenshtein distance under the 3RR defence configuration is shown in figure 20a for base conditions and in figure 20b with a reduced MTU. In base conditions the distances under TCP range from 0 to a bit under 300. This corresponds to the range of no defence distances from figure 16a divided by 3, as would be expected since traffic is split over 3 paths and therefore sequences are 3 times shorter. It shows a peak a little under 50, being almost 4 times as high as the second largest peak. QUIC shows distances ranging from 0 to 500, a bit larger than its range with no defence divided by 3. However, distances between 400 and 500 occur much less than shorter distances. QUIC has a peak a little over 50, also being almost 5 times larger than its second largest peak. There are two valleys in QUIC's distribution: one in between 100 and 200 and the other around 375.

When introducing a reduced MTU, figure 20b, the range of TCP is not altered at all and stays from 0 to a bit under 300. Also, its peak a little under 50 is still there, although relative to the now second largest peak it has grown and it is at least 6 times as high. Under QUIC the range of distances stayed from 0 to 500. It still has a peak a little over 50, while with distances from 175 and upwards occurrences are very low, with no other peaks visible.

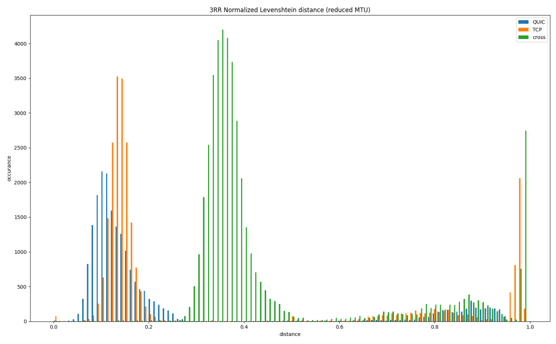
Without defence TCP was altered by reducing the MTU in range, which not visible under the 3RR defence configuration, and QUIC in height, which happens under 3RR as well.

After normalizing the internal Levenshtein distance figure 21a is obtained for base conditions and figure 21b with a reduced MTU. Under base conditions both QUIC and TCP have a peak between 0 and 0.2, like they had with no defence. However, unlike with no defence both now also show distances between 0.6 and 1. TCP has a second peak just below 1 of nearly equal height as the one between 0 and 0.2. QUIC on the other hand shows almost no distances of just below 1. Cross distances show a broader distribution of distances under the 3RR defence configuration than without defence. The peak just under 1 is still present, however, it lost its factor 2 in height compared to other peaks as they have all grown to a similar height.

After enabling the reduced MTU, figure 21b, the peak in between 0 and 0.2 under QUIC is reduced in height and

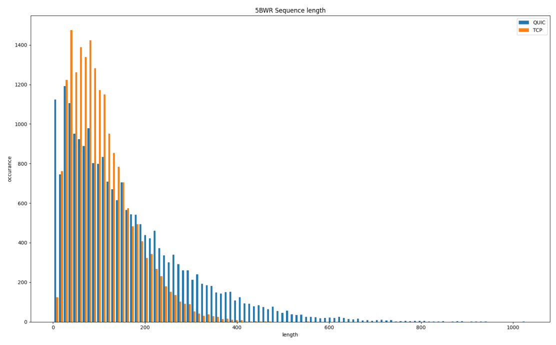


(a) Base conditions

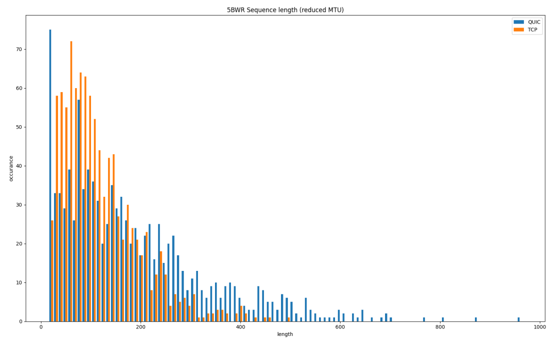


(b) Reduced MTU

Fig. 21: Distribution of internal normalized Levenshtein distances for 10 largest websites under the 3RR defence configuration. Normalized means that the Levenshtein distance is divided by the combined length of sequences, this ensures all inputs are mapped to [0,1].



(a) Base conditions



(b) Reduced MTU

Fig. 22: Distributions of sequence lengths for 10 largest websites under the 5BWR defence configuration.

widened at the base. Under TCP the peak in between 0 and 0.2 is not significantly altered. For both TCP and QUIC the number of occurrences of distances in between 0.6 and 1 has been greatly reduced. This includes TCP's second peak just below 1. Cross distances have a similar shape to base conditions, except that its peak just below 0.4 has grown, while the rest outside of this peak has all been reduced in number of occurrences. Unlike with no defence the cross distribution did not lose its peak just below 1 when faced with a reduced MTU under the 3RR defence.

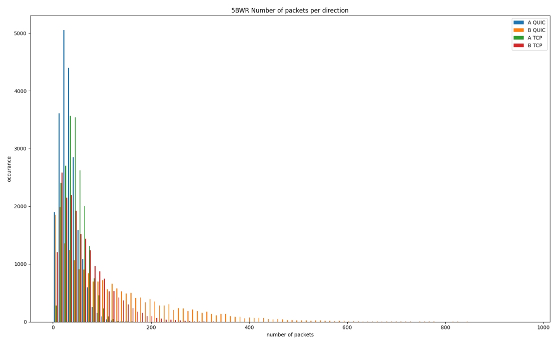
3) 5BWR configuration: Sequence lengths under the 5BWR defence configuration for base conditions in figure 22a are compared with a reduced MTU in figure 22b. Unlike under the 3RR configuration and with no defence, TCP does not show a peak at 0 in both base conditions and when faced with a reduced MTU.

For TCP the range of sequence lengths under base conditions is 0 to 400, with some outliers up to lengths of 500. The majority of the sequences has a length between 0 and 200. When reducing the MTU the distribution of sequence lengths under TCP does not change significantly.

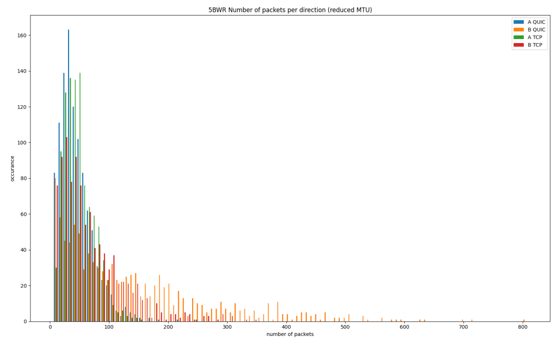
QUIC's range of sequence lengths spans from 0 to 900, however, it has a really long small tail. Like TCP the bulk of the sequences are shorter than 200 packets. QUIC's drop off in number of occurrences for larger sequences is less severe than under TCP. QUIC's distribution of sequence lengths also does not change significantly when reducing the MTU.

Splitting up the sequences in packets per direction results in figure 23a for base conditions and figure 23b with a reduced MTU. Under base conditions TCP shows in the client-to-server direction numbers of packets ranging from 0 to 150. Going to opposite way the numbers range from 0 to a bit over 300 for TCP. Under QUIC in the client-to-server direction the number of packets ranges from 0 to a bit under 150. In the other direction it ranges from 0 to a but over 800. Here QUIC displays a very long small tail in the graph, much longer than its TCP counterpart. TCP had the bulk of the server-to-client packets below 100 packets. QUIC has less server-to-client packets in that range and starts to have more occurrences above 100 packets.

The situation with a reduced MTU in figure 23b is very similar to base conditions. TCP has a range of 0 to 150 for the number of packets in client-to-server direction. In server-to-client direction TCP ranges from 0 to 300 packets, with a few outliers up to 350 packets. QUIC in the client-to-server

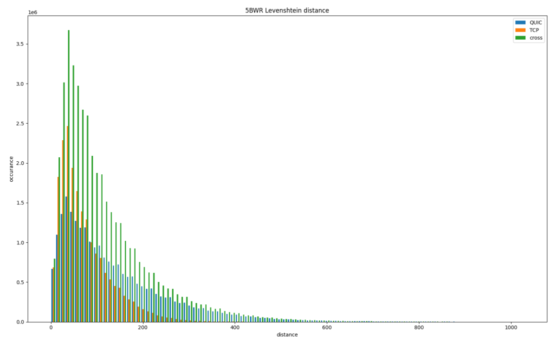


(a) Base conditions

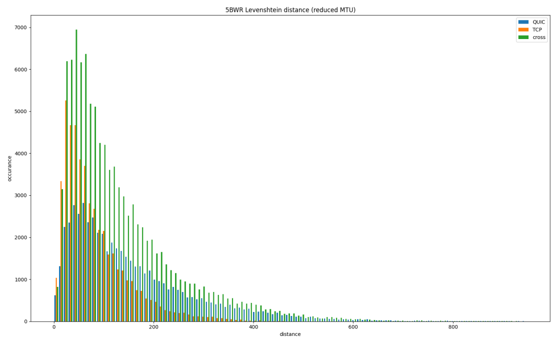


(b) Reduced MTU

Fig. 23: Distribution of number of packets per direction for 10 largest websites under the 5BWR defence configuration.



(a) Base conditions



(b) Reduced MTU

Fig. 24: Distribution of internal Levenshtein distances for 10 largest websites under the 5BWR defence configuration.

direction has a range of 0 to 150 packets, with a few outliers up to 250 packets. In the server-to-client direction QUIC ranges from 0 to 800, with the same long tail as under base conditions. It becomes quite sparse above 500 packets. For server-to-client packets the crossover point lies again around 100 packets, with TCP having more packets below and QUIC more above that. The fact that this crossover point is not disappearing or at least moving is an indicator that hypothesis should be rejected.

Measuring the Levenshtein distance in base conditions results in figure 24a and in case of reduced MTU in figure 24b. TCP has distances ranging from 0 to a bit over 400. This is not a shorter range than under the 3RR defence configuration. The shape of the distribution is different as well, under the 5BWR defence configuration it takes a shape reminiscent of a Poisson distribution. QUIC has a wider range than TCP, going from 0 to over 800. QUIC's distribution follows that of TCP in shape, except that its peak is lower and its right tail goes down slower and is therefore longer.

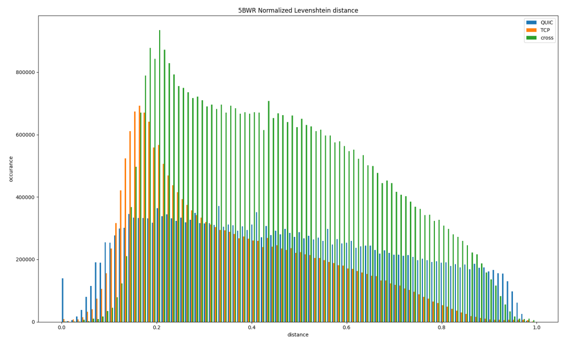
With a reduced MTU the range of TCP is not altered, it still going from 0 to a bit over 400. The shape of the distribution is the same as under base conditions with some small variations in height. For QUIC this holds as well, it has the same range, 0 to 800, and shape, also with some small variations.

Normalizing the Levenshtein distance results in figure 25a for base conditions and figure 25b with a reduced MTU. Under base conditions with the 5BWR defence configuration active, the plot looks very different from their counterparts under the 3RR defence configuration and with no defence. A single solid body across the full width of possible distances is visible in figure 25a instead of isolated peaks with gaps in between them like in figures 17a and 21a. TCP still has a peak around 0.2. QUIC, on the other hand, does not show any peaks, except a small one near 0 with a height below half of the main body's height. The cross distribution shows a peak around 0.2.

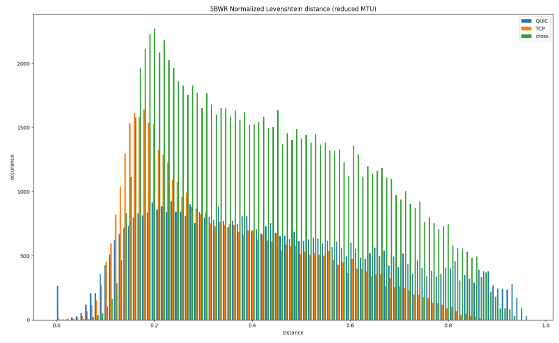
Like with figure 24b the introduction of a reduced MTU does not alter the distribution in figure 25b in a significant way, as was to be expected since they are based on the same underlying data.

4) Concluding: The overall goal of was to test the hypothesis that large TCP packets coming from the webserver would be split up into two QUIC packets in order to be able to be transported across the tor network, causing significant longer sequences when QUIC was used as transport protocol as opposed to TCP. Evidence to support that would come in the form of figures with a reduced MTU no longer showing a double amount of packets for QUIC compared to TCP.

With no defence or 3RR defence configuration active, when looking at the sequence length a few smaller differences are



(a) Base conditions



(b) Reduced MTU

Fig. 25: Distribution of internal normalized Levenshtein distances for 10 largest websites under the 5BWR defence configuration. Normalized means that the Levenshtein distance is divided by the combined length of sequences, this ensures all inputs are mapped to [0,1].

visible after introducing the reduced MTU, these are however not what one would expect to see if the hypothesis was true. Turning to number of packets per direction the situation remains the same, a few differences but mostly uninfluenced, while here one would expect to see that in the server-to-client direction there would be a difference under QUIC, as there the results of no longer splitting packets would manifest themselves most clearly.

Focusing on the 5BWR defence configuration, were the QUIC doubling effect was most profound. Sequences having a length of around 150 packets occur twice as frequent under QUIC than under TCP in figure 10 and the longer the sequences get the more extreme the difference in occurrence becomes. Or formulated from another perspective: for sequences that occur around 10 times or less those under QUIC are at least twice as long as those under TCP. When looking at 5BWR within the reduced MTU setup, figure 22b, the numbers are shifted a bit but the story remains similar. Sequences having a length of around 250 packets occur twice as frequent under QUIC than under TCP and for longer sequences the difference is even greater. Focusing on sequences that occur equally often but are twice as long under QUIC as they are under TCP, that happens only for sequence lengths occurring 4 times or less.

So evidence to reject the hypothesis is present across all defence configurations, therefore it is quite obvious that reducing the MTU does not have a significant impact on how the difference in packet sequences under QUIC and TCP looks like. This is furthermore supported by the fact that with a reduced MTU the average sequence lengths still show a discrepancy: 839 packets for QUIC vs 362 for TCP (no defence), 304 QUIC vs 143 TCP (3RR) and 187 QUIC vs 118 TCP (5BWR)[12]. Consequently, we cannot conclude that difference in MTU between QUIC and TCP is solely responsible for observed differences in the sequences (and also not for the difference in performance of TrafficSliver between QUIC and TCP).

An interesting side note, when using the reduced MTU link, it seemed that the experiment ran much smoother than before, that is the cURL connection did not time out and the weird behaviour where sometimes cURL would hang for multiples of 20 seconds before starting to transfer any data was no longer observed.

VII. Conclusion

Tor becomes more secure to use when more people use it. Replacing TCP by QUIC in Tor aims to improve the user experience and thereby making Tor more attractive to use. However, the main selling point for Tor is the anonymity it provides to its users. Performance improvements should never compromise Tor's ability to provide anonymity. Arriving at the main question this work intends to answer: What is the impact of replacing TCP by QUIC in Tor on its resistance to website fingerprinting attacks? Various attacks, defences and two implementations for Tor over QUIC were considered and evaluated. Ultimately, a single implementation of Tor over QUIC was picked to be combined with the TrafficSliver defence to shield it from the state-of-the-art and very potent DeepFingerprinting attack. It turns out that for TrafficSliver 5BWR is the best performing defence configuration under both QUIC and TCP. Although overall, the performance of TrafficSliver under TCP cannot be matched by QUIC, showing a consistent disadvantage for QUIC.

With the instances of Tor over TCP and Tor over QUIC used in this research, QUIC is characterized by having a sizeable overhead when longer sequences of packets are transported, up to a factor 2 in extreme cases. It was proven that this difference cannot be solely attributed to the difference in MTU between QUIC and TCP. Nonetheless, this overhead remains remarkable given that QUIC is designed to be efficient. It can also be seen as an indication that this particular instance of Tor over QUIC still needs fine-tuning. This lack leaves room for an potential improvement of its resistance to the DeepFingerprinting attack. This potential can only be properly

[12]Note that all averages have gone up after reducing the MTU compared to base conditions, showing that a lower MTU does indeed cause packets to be split for transport over tor.

investigated after significant time and effort is invested in polishing Tor over QUIC as much as its TCP counterpart already has received. Until then, it remains a bit of an uphill battle for QUIC.

So, in its current state Tor over QUIC is not quite ready to compete with Tor over TCP. However, given that QUIC is rapidly evolving and Tor over QUIC is far from being as well scrutinized as its TCP counterpart, it has potential to close the gap in performance between TCP and QUIC for TrafficSliver. Therefore, Tor over QUIC is definitely worth looking into again once its codebase is a bit more mature. This presents an opportunity for future work to investigate in which areas precisely QUIC is lacking and to determine whether these elements are just due to the program not being optimized or that it is inherent to the QUIC protocol.

More on opportunities for future work: The original plan of this research was to evaluate a broad spectrum of attacks and defences. However, the scope had to be narrowed, leaving ample room for future work to expand this work to its original intended scope. For example, the attacks and defences discussed in the related work section of this paper that were omitted due to time constraints but would be interesting to apply should be considered. Another approach angle could be the fact that, although, for the class of the evaluated attack, i.e. website fingerprinting, it appears there are no insurmountable obstacles stemming from replacing TCP by QUIC, there could be other classes of attacks were using QUIC instead of TCP poses a major issue.

For this research cURL was used to download websites instead of a full browser, which results in a smaller footprint for each website as the returned file is not interpreted and subsequently resources for, for example, rendering the webpage, interactive links or other links are not loaded. Using a full browser would result in larger and perhaps more distinctive sequences, which could influence the success rate of certain attacks or defences. Firefox supports SOCKS5 proxies for example. But this would come with its own challenges, as this likely means losing a major benefit of command line tools, being fully compatible with scripts.

All the setup files, patches, run scripts and data processing scripts used for this research, along with all the gathered data are available. Thus future work can extend and build upon this research if desired, to bring a more secure QUIC over Tor implementation closer to reality.

References

[1] Tor Project, "Tor." [Online]. Available: https://www.torproject.org/

[2] D. Stenberg, "TCP head of line blocking," HTTP/3 explained, 12 2019. [Online]. Available: https://http3-explained.haxx.se/en/why-quic/why-tcphol

[3] J. Iyengar and M. Thomson, "Quic: A udp-based multiplexed and secure transport," Internet Requests for Comments, RFC Editor, RFC 9000, May 2021.

[4] J. Iyengar and I. Swett, "Quic loss detection and congestion control," Internet Requests for Comments, RFC Editor, RFC 9002, May 2021.

[5] K. Hogan, "Security analysis of Tor over QUIC," Master's thesis, Massachusetts Institute of Technology, 11 2020.

[6] Chromium Code Reviews, "Issue 11125002: Add QuicFramer and friends. - Code Review." [Online]. Available: https://chromiumcodereview.appspot.com/11125002/

[7] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1928–1943, **code/dataset: https://github.com/deep-fingerprinting/df**. [Online]. Available: https://doi.org/10.1145/3243734.3243768

[8] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-cnn: A data-efficient website fingerprinting attack based on deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, pp. 292–310, 10 2019, **code/dataset: https://github.com/sanjit-bhat/Var-CNN**.

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

[10] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," *Proceedings 2018 Network and Distributed System Security Symposium*, 2018, **code/dataset: https://distrinet.cs.kuleuven.be/software/tor-wf-dl/**. [Online]. Available: http://dx.doi.org/10.14722/ndss.2018.23105

[11] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, ser. SEC'14. USA: USENIX Association, 2014, p. 143–157.

[12] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at internet scale," in *Proceedings 2016 Network and Distributed System Security Symposium*. Internet Society, 2016. [Online]. Available: https://doi.org/10.14722/ndss.2016.23477

[13] J. Hayes and G. Danezis, "K-fingerprinting: A robust scalable website fingerprinting technique," in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16. USA: USENIX Association, 2016, p. 1187–1203, **code/dataset: https://github.com/jhayes14/k-FP**.

[14] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1131–1148, **code/dataset: https://github.com/triplet-fingerprinting/tf**. [Online]. Available: https://doi.org/10.1145/3319535.3354217

[15] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright, "Tik-tok: The utility of packet timing in website fingerprinting attacks," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 3, p. 5–24, Jul 2020, **code/dataset: https://github.com/msrocean/Tik_Tok/**. [Online]. Available: http://dx.doi.org/10.2478/popets-2020-0043

[16] M. Nasr, A. Bahramali, and A. Houmansadr, "Deepcorr: Strong flow correlation attacks on tor using deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1962–1976, **code/dataset: https://people.cs.umass.edu/~amir/FlowCorrelation.html**. [Online]. Available: https://doi.org/10.1145/3243734.3243824

[17] J. Li, C. Gu, X. Zhang, X. Chen, and W. Liu, "Attcorr: A novel deep learning model for flow correlation attacks on tor," in *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 2021, pp. 427–430.

[18] W. D. la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engel, K. Wehrle, and A. Panchenko, "TrafficSliver: Fighting website fingerprinting attacks with traffic splitting," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2020. [Online]. Available: https://doi.org/10.1145/3372297.3423351

[19] M. Juárez, M. Imani, M. Perry, C. Díaz, and M. Wright, "WTF-PAD: toward an efficient website fingerprinting defense for tor," *CoRR*, vol. abs/1512.00524, 2015. [Online]. Available: http://arxiv.org/abs/1512.00524

[20] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1375–1390, **code/dataset: https://crysp.uwaterloo.ca/**

software/webfingerprint/. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao

[21] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 332–346.

[22] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Nov. 2014. [Online]. Available: https://doi.org/10.1145/2660267.2660362

[23] M. S. Rahman, M. Imani, N. Mathews, and M. Wright, "Mockingbird: Defending against deep-learning-based website fingerprinting attacks with adversarial traces," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1594–1609, 2021, **code/dataset: https://github.com/msrocean/mockingbird/**.

[24] K. Al-Naami, A. El-Ghamry, M. S. Islam, L. Khan, B. Thuraisingham, K. W. Hamlen, M. Alrahmawy, and M. Z. Rashad, "BiMorphing: A bi-directional bursting defense against website fingerprinting attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 505–517, Mar. 2021. [Online]. Available: https://doi.org/10.1109/tdsc.2019.2907240

[25] Tor Project, "Tor history." [Online]. Available: https://www.torproject.org/about/history/

[26] P. Syverson. Making anonymous communication. Center for High Assurance Computer Systems Naval Research Laboratory. [Online]. Available: https://www.onion-router.net/Publications/Briefing-2004.pdf

[27] Cloudflare, "What is a distributed denial-of-service (DDoS) attack? — Cloudflare." [Online]. Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

[28] Tor Project, "What are Entry Guards? — Tor Project — Support." [Online]. Available: https://support.torproject.org/about/entry-guards/

[29] Tor Project, "V2 Onion Services Deprecation — Tor Project — Support." [Online]. Available: https://support.torproject.org/onionservices/v2-deprecation/

[30] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, ser. WPES '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 103–114. [Online]. Available: https://doi.org/10.1145/2046556.2046570

[31] M. Alsabah and I. Goldberg, "Performance and security improvements for tor," *ACM Computing Surveys*, vol. 49, no. 2, pp. 1–36, Sep. 2016. [Online]. Available: https://doi.org/10.1145/2946802

[32] IBM, "What is Machine Learning? — IBM." [Online]. Available: https://www.ibm.com/cloud/learn/machine-learning

[33] Google, "Descending into ML: Training and Loss — Machine Learning — Google for Developers." [Online]. Available: https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss

[34] C. Viecco, "Udp-or: A fair onion transport design," *Proceedings of Hot Topics in Privacy Enhancing Technologies (HOTPETS'08)*, 2008.

[35] D. Herrmann, R. Wendolsky, and H. 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*, ser. CCSW '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 31–42. [Online]. Available: https://doi.org/10.1145/1655008.1655013

[36] M. F. Nowlan, D. Wolinsky, and B. Ford, "Reducing latency in tor circuits with unordered delivery," in *IEEE Symposium on Foundations of Computational Intelligence*, 2013.

[37] Cloudflare, "GitHub - cloudflare/quiche: Savoury implementation of the QUIC transport protocol and HTTP/3." [Online]. Available: https://github.com/cloudflare/quiche/

[38] J. Heijligers, "Tor over QUIC," Master's thesis, Delft University of Technology, 10 2021. [Online]. Available: http://resolver.tudelft.nl/uuid:1f473ff9-7d2c-43c9-8cfc-8f42e629b80f

[39] "Alexa The Web Information Company." [Online]. Available: https://www.alexa.com/

[40] D. Stenberg, "cURL." [Online]. Available: https://curl.haxx.se

[41] Tor Project, "chutney." [Online]. Available: https://gitweb.torproject.org/chutney.git

[42] The Tcpdump team, "tcpdump." [Online]. Available: https://www.tcpdump.org/

[43] Cloudflare, "What is MTU (maximum transmission unit)? — Cloudflare." [Online]. Available: https://www.cloudflare.com/learning/network-layer/what-is-mtu/

Appendix

The urls of the 100 websites (based on the Alexa top 2019) used in this research.

0) google.com
1) youtube.com
2) facebook.com
3) baidu.com
4) wikipedia.org
5) qq.com
6) taobao.com
7) yahoo.com
8) tmall.com
9) amazon.com
10) twitter.com
11) sohu.com
12) live.com
13) jd.com
14) vk.com
15) instagram.com
16) sina.com.cn
17) weibo.com
18) reddit.com
19) login.tmall.com
20) 360.cn
21) yandex.ru
22) linkedin.com
23) blogspot.com
24) netflix.com
25) twitch.tv
26) whatsapp.com
27) pornhub.com
28) yahoo.co.jp
29) csdn.net
30) alipay.com
31) naver.com
32) pages.tmall.com
33) microsoft.com
34) livejasmin.com
35) aliexpress.com
36) bing.com
37) ebay.com
38) github.com
39) tribunnews.com
40) google.com.hk
41) amazon.co.jp
42) stackoverflow.com
43) mail.ru
44) okezone.com
45) google.co.in
46) office.com
47) xvideos.com
48) msn.com
49) paypal.com
50) bilibili.com
51) hao123.com
52) imdb.com
53) t.co
54) fandom.com
55) imgur.com
56) xhamster.com
57) wordpress.com
58) apple.com
59) soso.com
60) google.com.br
61) booking.com
62) xinhuanet.com
63) adobe.com
64) pinterest.com
65) amazon.de
66) amazon.in
67) dropbox.com
68) bongacams.com
69) google.co.jp
70) babytree.com
71) detail.tmall.com
72) tumblr.com
73) google.ru
74) google.fr
75) google.de
76) so.com
77) cnblogs.com
78) quora.com
79) amazon.co.uk
80) detik.com
81) google.cn
82) bbc.com
83) force.com
84) deloplen.com
85) salesforce.com
86) pixnet.net
87) ettoday.net
88) cnn.com
89) onlinesbi.com
90) roblox.com
91) aparat.com
92) thestartmagazine.com
93) bbc.co.uk
94) google.es
95) amazonaws.com
96) google.it
97) tianya.cn
98) xnxx.com
99) rakuten.co.jp