Vera Rimmer*, Theodor Schnitzler, Tom Van Goethem, Abel Rodríguez Romero, Wouter Joosen, and Katharina Kohls

Trace Oddity: Methodologies for Data-Driven Traffic Analysis on Tor

Abstract: Traffic analysis attacks against encrypted web traffic are a persisting problem. However, there is a large gap between the scientific estimate of attack threats and the real-world situation. As traffic analysis attacks depend on very specific metadata information, they are sensitive to artificial changes in the transmission characteristics. While the advent of deep learning greatly improves the performance rates of traffic analysis attacks on Tor in research settings, deep neural networks are known for being implicitly vulnerable to artifacts in data. Removing artifacts from our experimental setups is essential to minimizing the risk of evaluation bias. In this work, we study a state-of-the-art end-to-end traffic correlation attack on Tor and propose a novel data collection setup. Our design addresses the key constraint of prior work: instead of using a single proxy node for collecting exit traffic, we deploy multiple proxies. Our extensive analysis shows that in the multi-proxy design (i) end-to-end round-trip times are more realistic than in the original design, and that (ii) traffic correlation attack performance degrades significantly on realistic timings. For a reliable and informative evaluation, we develop a general scientific methodology for replication and comparison of machine and deep-learning attacks on Tor. Our evaluation indicates high relevance of the multi-proxy data collection setup and the novel dataset.

Keywords: end-to-end correlation attack, traffic analysis, data collection, anonymity, deep learning.

DOI 10.56553/popets-2022-0074
Received 2021-11-30; revised 2022-03-15; accepted 2022-03-16.

***Corresponding Author: Vera Rimmer:** imec-DistriNet, KU Leuven, E-mail: vera.rimmer@kuleuven.be
Theodor Schnitzler: Ruhr-Universität Bochum, E-mail: theodor.schnitzler@rub.de
Tom Van Goethem: imec-DistriNet, KU Leuven, E-mail: tom.vangoethem@kuleuven.be
Abel Rodríguez Romero: imec-DistriNet, KU Leuven, E-mail: abel.rodriguezromero@kuleuven.be
Wouter Joosen: imec-DistriNet, KU Leuven, E-mail: wouter.joosen@kuleuven.be
Katharina Kohls: Radboud University, E-mail: kkohls@cs.ru.nl

1 Introduction

Encrypting traffic is a core security feature of the Internet. An additional layer of protection becomes available with the use of anonymity systems like Tor [42], where multiple layers of encryption and relayed connections conceal the activity of users. In this context, traffic analysis attacks on encrypted traffic are a persisting problem. They circumvent the data encryption by inferring sensitive information from the transmissions metadata.

An important class of attacks in this context are *end-to-end traffic correlation* attacks. The adversary monitors traffic in both connection endpoints (client to entry guard and exit relay to server). Based on similarities between the incoming and the outgoing traffic at both ends, they can de-anonymize Tor connections [10, 24, 27, 28]. End-to-end attacks stand in contrast to *website fingerprinting* [5, 31, 33, 38, 39], where de-anonymization is achieved through classification of entry traffic. The current state of the art estimates considerably high success rates for both types of attacks. However, ensuring *realistic* experimental setups remains challenging for academic work in this context.

Creating a realistic test setup is not trivial due to a diverse set of challenges. For example, as the critical evaluation by Juarez et al. shows [19], the world size for traffic analysis attacks still drastically limits the realism of existing performance benchmarks. The novel attack concepts and improved performance rates ought to be considered together with their experimental limitations.

For the evaluation of end-to-end traffic correlation attacks, the goal of a responsible measurement setup is particularly challenging. The adversary monitors traffic at *both* ends of the connection that makes use of a worldwide infrastructure. While it is trivial to gain access to the entry connection, there is no straightforward and privacy-preserving solution to accurately monitor traffic at generic exit links of the Tor infrastructure. Different workarounds allow us to circumvent this problem and still get access to the exit traffic. In the recent work by Nasr et al. [27], the use of a proxy node between the exit and target server provides access to the exit traffic. Despite the benefits of the proxy-based design, one proxy

at a fixed location creates an *additional intercontinental hop* for a large fraction of connections but not for the others, and thus interferes with the *timing characteristics* of the dataset, which are often utilized in traffic analysis attacks [5, 6, 14, 27, 32]. The impact of such artifacts on the quality of data and, eventually, on the quality of attack performance estimations is unknown.

Another challenge of reliable empirical evaluations of traffic correlation attacks stems from the probabilistic nature of leveraged algorithms. Deep learning produces unprecedented performance in end-to-end side-channel attacks on time-series data, and various traffic analysis attacks against Tor are no exception [5, 27, 33, 38]. However, high success rates come at the cost of increasing complexity and computational costs, performance instability and low interpretability. While exhaustive evaluations are infeasible, the progress of the field depends on our ability to draw reliable conclusions from limited empirical evidence. In particular, correct selection of metrics, proper tuning of complex models and fair comparison between them are all fundamental aspects.

In this work, we explore the impact of the characteristics of the underlying research dataset on effectiveness of end-to-end traffic correlation. Based on the preliminary statistical analysis, we introduce a novel multi-proxy data collection setup that overcomes the supposed key constraint of prior setups. In the second step of our study we replicate a state-of-the-art end-to-end traffic correlation attack DeepCorr [27] on our novel data. To this end, we design a replication methodology for data-driven attacks that utilizes appropriate performance metrics, accounts for various sources of bias and variability in machine and deep learning evaluations and strives for reliable performance comparison. Our extensive experimentation shows consistently lower attack performance in the proposed multi-proxy setup, with high statistical significance. As a result, our work demonstrates a successful reduction of the timing bias introduced by the proxy usage.

In summary, our paper makes three core contributions:

(i) **Novel Multi-Proxy Setup.** We propose a novel data collection design for traffic correlation attacks in research settings that uses multiple proxies instead of a single proxy to capture exit traffic. Our strategic statistical analysis of Tor traffic transmission characteristics shows that the proposed design more closely approximates realistic timing measurements as compared to the state of the art.

(ii) **Attack Replication Methodology.** In order to accurately compare traffic correlation effectiveness in single- and multi-proxy data collection setups, we devise a general methodology for scientific replication and comparison of data-driven attacks on novel or modified data. The proposed algorithm aims to minimize evaluation biases and considers statistical significance of results, which improves over the common practices in the field.

(iii) **Evaluation Results.** Our empirical evaluation of an optimized DeepCorr attack demonstrates an average 7.95 % performance decrease in a more realistic multi-proxy setup. This outcome indicates that the proposed multi-proxy setup reduces the subtle bias introduced by the single-proxy design of prior work. We publish the corresponding dataset and code to facilitate further research[1].

2 Technical Background

In the following, we document the technical background of Tor, provide a general overview of traffic analysis attacks, and discuss existing attacker models.

2.1 Circuits and Connections

Tor circuits usually consist of three relays that transmit information between the client and the server. As such circuits are part of the application-layer security of Tor, they do not directly transport TCP packets, but handle multiple TCP streams of, e. g., a website request. On startup and during runtime, Tor continuously creates new circuits and discards older ones depending on different features, e. g., their lifetime or status flags. A standard circuit consists of an entry guard that connects to the client, a middle relay as the intermediate connector, and the exit relay that connects to the requested destination. The entry guard is picked from the user's guard set, which consists of a static selection of candidate nodes and gets updated after several months [13]. All remaining nodes are selected randomly from the consensus following a bandwidth-oriented preference.

A website visit results in multiple TCP streams that load the (sub)resources of the site. For a new website visit, Tor checks all currently available circuits and determines the one that best suits the current request. It then attaches all streams of that request to the cir-

1 https://distrinet.cs.kuleuven.be/software/tor-tc-dl/

cuit. After a circuit has been used for a connection, it is marked dirty and will not be reused.

2.2 Traffic Analysis Attacks

Traffic analysis attacks enable an adversary to learn sensitive information from the encrypted transmissions of a network. In the context of Tor, the worst-case leak of sensitive information allows the adversary to de-anonymize the connection.

Two primary types of traffic analysis attacks are website fingerprinting (WF) and end-to-end (E2E) traffic correlation, with the latter being the focus of this study. E2E correlation attacks focus on the similarities of transmissions at two endpoints of the connection, i. e., at the *entry* and the *exit* relay of the Tor circuit. In this setting, the entry connection reveals the IP address of the victim and the exit connection reveals the IP address of the server. The de-anonymization is successful if the adversary can find a related pair of entry and exit traffic. The success of an E2E attack is mainly influenced by the access to traffic in the network.

3 Challenges of Data Collection

Accessing Tor exit traffic to evaluate traffic analysis attacks in practice requires the use of a proxy in the connection between the exit relay and the server. Although it is technically possible to achieve this by running an exit node, this poses various ethical concerns (collecting data of other Tor users) and other restrictions (lack of intrinsic traffic patterns of real-world Tor exit nodes). When using a proxy setup, an extra hop is added to the connection, which affects timing characteristics of the monitored traffic. So far it has not been investigated what the consequences of these alterations for the attack and its performance are. Prior data-driven state-of-the-art E2E attacks [27] used a single proxy server located in the US. While the authors document that the additional latency leads to performance impairments *for the attack*, we run a series of experiments to empirically analyze the impact of an additional hops in different proxy setups in practice. We argue that a single proxy introduces different amounts of timing overhead, e. g., depending on how many intercontinental hops are part of the end-to-end route. Such timing differences might affect the distinguishability of traces from different groups of circuits. We further analyze how realistic the resulting overall timings are, given that a real attacker would not use a proxy at all but instead extract the traces directly from the exit connection.

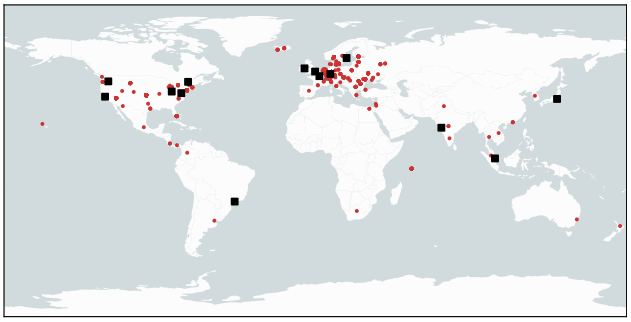


Fig. 1. Overview of exit relay locations (red) and locations of proxies (black) in the preliminary analysis.

3.1 Measurement Setup and Overview

Over a period of one week, we run a client in our institution to measure round-trip times (RTTs) to each of the top 100 websites in a recent Tranco list[2] [20] through 15,935 standard three-hop Tor circuits. In addition, we collect RTTs to 14 geographically distributed servers, located in the most important data center locations in the world (cf. Figure 1). Finally, we measure the RTTs from each of these servers to each website. To measure the RTT to a website, we send an HTTP `HEAD` request and measure the time until our client receives the response. To be able to compare these timings to RTTs between our client and the servers, we run a simple HTTP response server on each server that also responds to HTTP `HEAD` requests by our client.

For each proxy, we combine the time from our client to that proxy with the time from the respective proxy to each website. This way, we are able to simulate the timing overhead introduced by different proxy setups and compare different proxy setups with each other.

3.2 Analysis of RTT Measurements

We now present the results of our preliminary timing measurements. In particular, we conduct the following analyses to evaluate the practical impact of different proxy setups on timing characteristics of Tor traffic:

2 https://tranco-list.eu/list/4L8X

Table 1. Numbers of inter-continental connections within Tor circuits. Since our client is located in Europe, the number of hops is always uneven in the single (US) proxy setup. Only circuits with all relays in Europe or North America (i. e., 96 % of all circuits) are included.

Hops	Single Proxy	Multi-Proxy
0	–	8,653 (54.3 %)
1	10,811 (67.8 %)	2,158 (13.5 %)
2	–	4,131 (25.9 %)
3	4,468 (28.0 %)	337 (2.1 %)

(i) We analyze differences in timings from our client to a single proxy in the U.S. between groups of circuits, that are defined by the number of inter-continental transmissions between two subsequent hops in the connection.
(ii) We compare timings measured with two proxy setups, i. e., a single proxy in the U.S. and 14 geographically diverse proxies to the timings a realistic adversary without a proxy would see.
(iii) We quantify the overhead introduced in the timings depending on the number of proxies in our set of 14 proxies.

3.2.1 Inter-continental Connections in DeepCorr

We argue that relaying all Tor traffic through a single proxy to evaluate the attack affects the end-to-end RTTs of monitored Tor traffic. This may lead to unknown consequences for other timing features that may affect the distinguishablity of traces within the feature learning process. For Tor circuits whose exit is located in Europe, the use of a single proxy in the U.S., as done in state-of-the-art traffic correlation attack evaluations [27] introduces the timing overhead of an inter-continental hop at the end of the connection.

We generalize this assumption and compare the end-to-end timings between our client and websites across groups of circuits that are defined by the number of inter-continental hops in the connection. For the sake of simplicity, we only consider circuits with all relays located in Europe or North America, i. e., 96 % of all circuits in our set of 15,935 standard 3-hop circuits. Table 1 lists all circuits in our set depending on the number of inter-continental hops in the connection for both the single US-proxy and multi-proxy setup. Due to our client being located in Europe, the connection to the US-based proxy includes either 1 or 3 inter-continental hops. Effectively, the numbers of circuits with 0 and 2 inter-continental hops in the multi-proxy setup constitute those circuits, that include an inter-continental hop between exit relay and US proxy in the single proxy setup. That is, for 80 % of circuits, the single proxy setup introduces an *additional* transfer between Europe and North America.

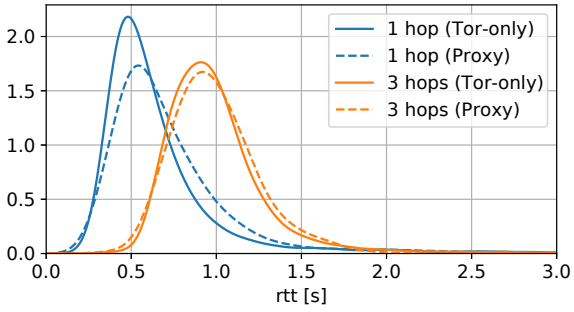


Fig. 2. Distribution of RTT between client and proxy by the number of inter-continental hops in the connection. Tor-only denotes that all hops occur within the Tor circuit, proxy denotes that the last hop is between exit relay and proxy.

Introducing an additional hop in the connection may be a critical issue since it affects round-trip times between client and proxy with potential consequences for the feasibility of traffic analysis. Figure 2 illustrates the distributions of RTTs between client and proxy depending on the number of inter-continental hops in the single proxy setup. We also distinguish whether all hops occur within the Tor circuit (*Tor-only*) or if one hop is introduced by the *Proxy*. Whereas we see differences in timings depending on the number of inter-continental hops (1: $mean = 0.68\,s$, $sd = 0.42\,s$, 3: $mean = 1.01\,s$, $sd = 0.36\,s$), it does not seem to affect the timings at what place in the connection the hop occurs, whether it is in the Tor circuit or between exit and proxy.

3.2.2 Single Proxy vs. Multi-Proxy vs. No Proxy

In the next step, we compare the RTTs to all websites in our dataset through all Tor circuits in our dataset. We use three different proxy setups, (i) a single proxy server located in the US (East), reflecting the setup used for state-of-the-art evaluations of traffic correlation [27], (ii) a set of 14 proxies out of which we select the one that is located nearest to the exit for each circuit, and (iii) no proxy, which reflects timing characteristics of traffic a real adversary would access directly in-transit. For the sake of simplicity, we technically do not measure end-to-end RTTs but collect (i) timings through each Tor circuit to each proxy, (ii) timings through each Tor cir-

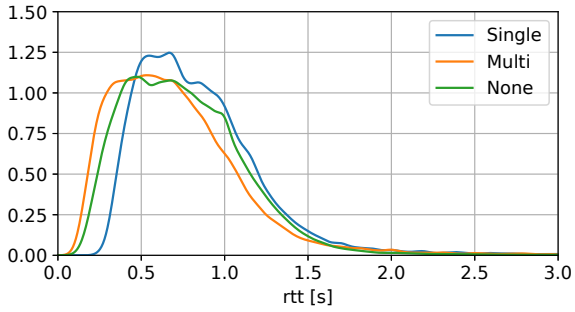


Fig. 3. Empirical distributions of end-to-end RTTs between clients and websites in three proxy setups.

cuit to each website without a proxy, and (iii) timings from each proxy to each website. We then effectively simulate the different proxy setups, by composing respective timing segments to end-to-end RTTs.

Figure 3 shows the distributions (probability density functions) of end-to-end RTTs in each of these setups. We see that the timings in the *multi*-proxy setup ($mean = 0.72\,s$, $sd = 0.43\,s$) are more similar to the setup without proxy (*none*, $mean = 0.76\,s$, $sd = 0.38\,s$) than those in the *single*-proxy setup ($mean = 0.86\,s$, $sd = 0.41\,s$). The overlap of *single* and *none* distributions is 77 %, whereas it is 84 % for *multi* and *none*. We also notice that timings in the multi-proxy setups tend to be even *lower* than in the setup without a proxy. We attribute this to our proxies being located in large data centers and presumably better connected to many websites (also located in data centers) than Tor exits are. Whereas the multi-proxy setup also affects the RTTs (we discuss this limitation in Section 7), the effect is yet smaller in absolute numbers than for the single-proxy setup. Therefore, we argue that the multi-proxy setup is a better approximation of a setup without a proxy w. r. t. their timing characteristics, which is relevant for traffic correlation attacks.

Differences between timing distributions even increase when we compare the timings measured for each website. Table 2 lists the average RTTs for the three proxy setups and the timing distribution overlap for each of the single and multi proxy with the no proxy setup. For most websites, the distributions of measured timings in the setup with multiple proxies are more similar (i. e., higher overlap) to the setup without proxy, than the timing distributions of the single proxy setup.

Table 2. Mean end-to-end RTTs and timing distribution overlap.

Website	Multi [s]	Overlap (M,N)	None [s]	Overlap (N,S)	Single [s]
google.com	0.68	71 %	0.77	51 %	0.95
youtube.com	0.68	70 %	0.78	50 %	0.95
facebook.com	0.75	91 %	0.76	41 %	1.23
netflix.com	0.69	79 %	0.76	58 %	0.90
microsoft.com	0.98	88 %	1.00	41 %	1.23
twitter.com	0.74	86 %	0.77	85 %	0.81
instagram.com	0.66	82 %	0.71	39 %	0.94
apple.com	0.62	76 %	0.71	77 %	0.76
linkedin.com	0.62	82 %	0.67	75 %	0.76
qq.com	1.03	72 %	1.12	75 %	1.22
all (top 100)	0.71	84 %	0.75	77 %	0.85

3.2.3 Number of Proxies

Our next goal is to estimate the required number of proxies to limit the transmission overhead. Whereas we have seen that selecting the nearest one out of a set of 14 server instances better reflects the timing characteristics of Tor traffic without an exit proxy, we now investigate to what extent the transmission overhead depends on the number of proxies to choose from. For each circuit in our dataset, we generate sets of n randomly selected proxies (for each n from 1 to 14) and calculate the distances from the exit relay to the nearest proxy in each set. Our results in Figure 4 show that the average minimum distance of more than 5,700 km when only one proxy server is used, is reduced to approximately 1,000 km when 5 servers are used, and to less than 500 km with 14 server instances in place.

In addition, we also measured the timing overhead per number of server instances used. Our results (cf. Figure 4) show that, while the average RTT is around 0.75 s without a proxy, we measure an average RTT of 0.94 s in a single-proxy setup, effectively introducing 190 ms overhead. When we increase the number of proxies to up to 14, the average RTT gradually approximates the value of the setup without a proxy. In summary, our observations suggest that using a single proxy in the U.S. for monitoring Tor exit traffic introduces a specific timing overhead and, therefore, might also affect timing features that play a role in the evaluation of end-to-end confirmation attacks. We assume that a setup incorporating multiple proxies closer to the exits is a better fit for simulating timing characteristics and, therefore, traffic a realistic adversary would monitor in practice without a proxy.

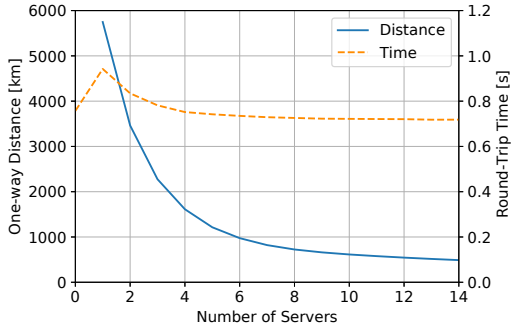


Fig. 4. Distribution of Tor's infrastructure by fractional numbers of nodes and bandwidth shares per country.

4 Data Collection Setup

In the following, we document the experimental setup that we use to record traffic for evaluating end-to-end confirmation attacks.

4.1 Requirements

The setting of E2E attacks requires recording traffic at both endpoints of a connection. In the following, we discuss the implications of designing the experimental setup for an E2E adversary. In general, the dataset can also serve for evaluations of WF attacks (both closed- and open-world) that are conducted with entry traffic only, which is a subset of the data we collect.

4.1.1 Proxy Setup

A recent attack by Nasr et al. [27] uses deep learning for an E2E attack. In their experimental setup, the authors use a `SOCKS5` proxy between the exit relay and server of a connection to be able to record traffic at both ends of the live Tor network. Using an additional proxy has the disadvantage of adding another hop to the connection, which extends the transmission distance and changes the direct connection between the exit relay and the server. However, in contrast to using a dedicated exit relay, it does not introduce any ethical complications and can be adjusted in a way that the additional overhead can be minimized. The following characteristics must be considered when using an additional proxy.

Circuit Diversity. During one browsing session, Tor creates multiple circuits that are used and switched on a regular basis. Despite a primary entry guard that remains the same as long as possible, the remaining middle and exit relays change for different circuits. Because of Tor's worldwide distribution of relays, each circuit has individual performance characteristics such as the circuit length or the use of advertised bandwidth. Using a proxy allows to maintain the original circuit diversity.

Server Locations. Both attacks require the generation of a representative dataset including website requests to a high number of different server locations. Along with the distribution of exit relays, this creates a high diversity in the last hop of the connection. Examples of this are varying connections lengths between the exit and server and individual transmission characteristics, respectively. Using multiple proxies at different strategic locations allows to limit the overhead of the additional hop (cf. Section 3.2.2).

Onion Service Traffic. Using the `Alt-Svc` header, web services can indicate that the website can be accessed over an alternative service. For instance, when the website detects that it is being accessed over a Tor connection, it can refer the user to the relevant `.onion` address. The user agent will then initiate a connection to the onion service and use it to request all subsequent resources. Because of our proxy-based setup, websites receive connections from a cloud provider and not a Tor exit node. Furthermore, as all client traffic is direct to the proxy, no connections to onion services will be made. In a real-world scenario, traffic to the onion services would still appear in the compromised guard node but not in the exit node. Consequently, this minimally affects website fingerprinting attacks, but needs to considered by the adversary in an E2E confirmation attack. We further discuss this limitation in Section 7.

4.2 Setup Overview

The proxy setup consists of a series of clients, a central management entity, and an additional proxy between the exit and server of a connection (cf. Figure 5). The clients run in virtual machines in our institute's cloud infrastructure and consist of the Tor Browser Bundle (TBB), `3proxy`, which allows proxy chaining in addition to Tor's `SOCKS5` proxy, and the core Tor implementation that establishes circuits and uses them to connect to web servers. At each end of the connection, we use `tcpdump` to capture the entry traffic between the client and the entry guard and the exit traffic between the exit relay and the server. The central management scripts organize the website crawling and orchestrate `tcpdump` at both endpoints of the connection.

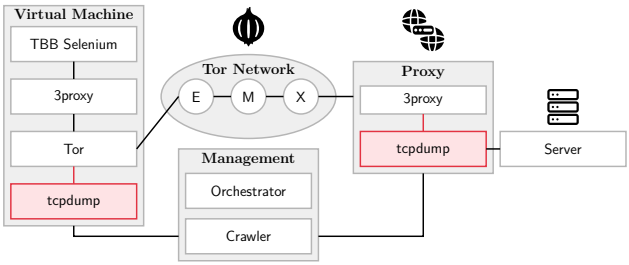


Fig. 5. Experimental setup.

4.2.1 Proxy Chain

3proxy[3] is a lightweight proxy client and server that allows us to put an additional proxy between the TBB and core Tor implementation that connects to the corresponding proxy between the exit and the server. As one goal is to minimize the overhead of the additional proxy hop, we run multiple entities at strategic positions and pick the location with the minimum distance between exit and proxy for each circuit established. To this end, we run 18 proxies in 6 different locations (1 in London, 1 in San Francisco, 3 in New York, 2 in Amsterdam, 1 in Singapore, and 10 in Frankfurt). These proxy locations reflect the distribution of large data center locations around the world. Each proxy has an individual port assigned on the client side. Depending on the circuit that will be used next, we pick the best fitting proxy by deciding for the port that matches the correct server entity. At this point, it is very important to select a proxy before the stream, i. e., the TCP connection, is attached to circuit. Therefore, we must predict the circuit that will be used next and attach it to the fitting port and proxy combination.

4.2.2 Circuit Prediction

If we do not want to interfere with the circuit establishment procedure and selection of relays, we must pick a proxy according to the circuit used in a connection. This is challenging, as Tor builds circuits in advance, and only decides for one of them at the moment the client initiates a connection to a server. However, using an additional proxy between the exit and the server requires using a chain of proxies that adds our own SOCKS5 node behind the initial proxy that connects the Tor client to the entry relay. Technically, we must set up this proxy chain at a specific port that later is used by Tor for the connection establishment. That said, we need to pick the correct candidate from the set of available circuits that Tor will use for the next connection, identify the best (closest) proxy for the exit of this circuit, set up the proxy chain at the port Tor will use, and then wait for the connection to be made.

To pick the correct circuit, we implement a small patch in the Tor client that copies the original circuit selection procedure from Tor, i. e., we apply the same rules as Tor to determine the upcoming circuit. The patch is a control port implementation that iterates all existing circuits and returns the one that is most likely to be picked for the next connection. We use this information for two purposes. First, we estimate the location of the exit relay in the circuit to prepare the *closest* proxy for this connection. Second, we use the IP of the exit relay to allow the incoming connection at the proxy. This allows us to restrict any other connection attempt that is unrelated to our experiments. A side effect of these access rules is that only correctly picked circuits succeed. In case we pick the wrong circuit (and consequently a false proxy location), the incoming connection uses a different IP address and is blocked at the proxy.

4.3 Data Collection Details

In total, we collected three datasets that are used in the experiments presented in the remainder of this paper. Each dataset was collected on a distributed crawler setup using the latest Tor Browser version (9.0.1) which was orchestrated through tbselenium [1]; for the SOCKS proxy, we used 3proxy (version 0.9). For each data collection, the most recently available tor version was used, with the extension that allows us to predict the location of the exit node. The 64 instances of the crawler were distributed over 8 virtual machines that were each provisioned with 4 vCPUs and 8 GB of RAM. The first two datasets, **SD** and **MD** consist of visits to the 15,000 most popular websites according to the Tranco ranking[4] [20]. For the **SD** dataset, only a single proxy run on Amazon's Northern Virginia (us-east-1) datacenter was used. For the **MD** dataset, we leveraged proxies that were globally distributed over six Amazon datacenters, covering America, Europe and Asia. The data for these two datasets was collected between July 28, 2021 and August 20, 2021. In the final largest dataset, $\mathbf{MD}_{1mln}$,

3 3proxy.ru

4 https://tranco-list.eu/list/65VX

we collected traces for the Tranco top 1 million sites[5], also using proxy servers geographically distributed over six locations. The data collection for this dataset started on March 5, 2020 for a duration of 8 days.
All datasets will be made publicly available in a parsed form; the raw PCAPs are available upon request.

5 Evaluation Methodology

In this section, we outline our evaluation methodology. The main goal is to establish whether the choice of a data collection setup indeed impacts the estimated feasibility of traffic correlation attacks. Previously we have shown that the multi-proxy setup is much better at approximating the timing characteristics of traffic in the real Tor infrastructure. However, it is still unclear whether the disturbed timing measurements in the single-proxy setup make a simulated traffic correlation attack easier to deploy. With this evaluation, we aim to empirically compare performance of the state-of-the-art data-driven E2E attack (DeepCorr[27]) on two different datasets: **SD** collected through a single-proxy setup from prior work, and **MD** collected through a multi-proxy setup proposed in our study.

5.1 Validity

Comparing two datasets is not a trivial task due to a number of influential factors that are largely out of control of researchers. Such factors, like the current state of the Tor infrastructure, the amount of users and the dynamically changing web content, may impact the measurements in an unpredictable manner. We call these factors *the impact of time*. In order to minimize the impact of time on our two datasets and thus provide a consistent ground for comparison, we have already taken the following steps in Section 4:

(i) Both datasets are collected within the same time frame: August 2021.
(ii) We fix the Tor browser version to 0.4.6.6.
(iii) We use the same set of 15,000 websites for one-time visits in both datasets. After filtering the failed web visits and traces that may be too short for learning (less than 100 packets), we fix a set of 7948 websites that are present in both datasets.
(iv) The only controlled variable is the amount of proxies used in the setup: one for the single-proxy evaluation and 18 for the multi-proxy evaluation.

With this set of actions, we try to ensure that any differences in data originate from the number and locations of proxies, and not from artifacts or inconsistencies in data collection.

Furthermore, for a valid comparison between two datasets we recognize several influential aspects of developing and evaluating data-driven attacks. These include (i) correctly chosen performance metrics, (ii) a proper attack replication algorithm for evaluating deep learning models on novel data, and (iii) explainability analysis. Careful incorporation of these aspects will ensure that the observed difference in attack performance can be attributed to the actually present semantic differences in data, and not to incorrect interpretation of results, biases or chance. This is especially important for attacks based on deep neural networks, whose learning mechanisms are hardly interpretable and heavily rely on randomness and a multitude of hyperparameters with complex relations. Further in this section, we explain each of these parts of our evaluation methodology.

5.2 Performance Metrics

Scientific empirical evaluations of traffic analysis attacks against Tor aim to estimate the technical threat to privacy of Tor users. However, in lab conditions, these attacks are deployed in simulated scenarios that operate under a set of simplified assumptions, which makes performance estimation technically feasible and systematic, but inevitably limits the attack realism. E.g., related work has focused on perfecting performance estimation for WF attacks, to respect the open-world evaluation challenges [19, 43]. Estimation of actual E2E traffic correlation attack performance for powerful adversaries in the wild is extremely difficult in lab conditions, as it depends on the realism of the threat model in terms of coverage of Tor relays and amount of concurrent connections. However, in this paper, we set the scope on the realism of measurements for E2E attacks affected by *the choice of the data collection setup* as is used in scientific studies. For comparing relative attack performance on similar data with two different data collection setups, a fixed large number of traces is sufficient. Nevertheless, in a simplified research setting the exact choice of metrics and correct interpretation of results are especially important. We look into metrics used by the state-of-the-art and propose improvements.

5 https://tranco-list.eu/list/KW8W

In prior work, a common approach employed to estimate E2E correlation performance is to report and compare the False Positive Rate (FPR), exemplifying the fraction of non-associated pairs that were incorrectly predicted as correlated, and the True Positive Rate (TPR), or Recall, that illustrates the fraction of truly associated pairs that were rightfully predicted as correlated. We argue that using these scores for performance estimation is insufficient and may even be misleading, as they do not provide a complete view over the attacker's capabilities. The reason is the *base rate fallacy* specific to the E2E attack use case. This problem originates from the *inherent imbalance* of positive and negative pairs in the test set: the adversary has to correlate all combinations of traces ($N \times N$), while only 1 out of N intercepted connections is truly associated. In other words, the base rate of positive samples is very low – $\frac{1}{N}$, and decreases with the growth of the volume of traffic. With such a low base rate, even a high TPR and a seemingly low FPR do not make the attack reliable.

To understand this with an example, consider the asserted optimal performance values of DeepCorr: a TPR/Recall of 0.80 and a FPR of 10^{-3} obtained for a number of associated pairs $N = 25,000$. While this may seem effective, when translated to absolute numbers, the actual True Positives amount to $TP = TPR \times \#pos = 0.80 \times 25,000 = 20,000$, while the actual False Positives amount to $FP = FPR \times \#neg = 0.001 \times (25,000 \times 24,999) = 624,975$. This means that for each correctly detected associated pair of flows, the DeepCorr model mistakenly predicts with the same confidence around 30 non-associated pairs as correlated, and is therefore extremely *imprecise*. This issue is not visible from TPR and FPR, or the corresponding ROC-curve, alone. What happens is that precision is ignored during optimization: for this model, precision in fact equals $Pr = TP/(TP + FP) = 20,000/(20,000 + 624,975) = 0.031$, with the maximum possible value of 1.0 and a baseline of $25,000/(625 \times 10^6) = 0.00004$.

DeepCorr performance is decidedly superior to the E2E correlation attacks previously reported in the literature [10, 22, 24, 28], and further improves when using more packets in a flow. However, it is crucial to report all relevant metrics for the sake of transparent and informative estimations. Depending on the attacker goals and the costs of false positives and false negatives, one may choose to optimize the *precision-recall trade-off* to obtain a more trustworthy (precise) detector.

Prior work optimizes the attack based on *loss* – the optimization function used in deep neural networks to learn from data and to detect when to stop learning. However, specifically for DeepCorr, there is no theoretical guarantee that the optimal loss will correspond to optimal precision-recall trade-off (the final goal of the attack). So instead, an adversary may choose to also evaluate a chosen domain-specific metric on every training epoch in addition to the standard loss. In our evaluation, we tackle the inherent disconnect between loss and attack target metrics by optimizing for Average Precision (AP) that summarizes area under the PR-curve.

5.3 Attack Replication

We aim to analyze the performance difference of E2E correlation attack when deployed on a dataset collected with a single proxy versus a dataset collected with multiple proxies. Higher attack performance on the single-proxy dataset would confirm that the deviation of end-to-end timing characteristics from the real-world setup favorably impacts attack simulation. At the same time, lower attack performance on the multi-proxy dataset would indicate that correlation of traffic is a harder problem than it appears from prior research, and including multiple proxies in data collection is a more accurate way to model the problem. The difference in E2E attack performance thus serves as *an indirect signal of quality of the two data collection setups* and is obtained through experimental analysis.

The challenge of experimental analysis is in deriving reliable and generalizable conclusions from limited empirical evidence, which is especially pronounced with data-driven approaches. WF and E2E correlation attacks have long leveraged traditional statistical methods and machine learning algorithms, before at last considering deep learning in 2018 as a novel, powerful approach for both Tor traffic analysis attacks [27, 33]. While relaxing the need for domain knowledge through eliminating manual engineering of salient traffic features, deep learning does increase the complexity and is overall not straightforward in deployment. A common challenge comes from the need of replicating state-of-the-art deep learning attacks on a novel dataset collected in different conditions. In this case, it is not sufficient to reuse the exactly same configuration of a deep neural network reported in prior work. The reason is that the change of any origin in the underlying data distribution may affect the nature and difficulty of the learning problem. To account for that, the deep learning attack needs to be retrained on novel data.

In the context of Tor, many factors may affect the underlying data distribution in traffic and thus change

Algorithm 1. Replication and comparison of a ML/DL attack on two different datasets. Here, with AP as a target metric and t-test for statistical significance.

Input: Datasets D_1 and D_2, ML/DL attack A, hyperparameter search space S.
Output: Dataset with highest mean attack performance; the difference between means; statistical significance.

1 Set number of cross-validation folds K
2 Set maximum number of attack configurations C
3 Set number of randomized training runs n
4 $AP_{D_1} \leftarrow \{\}$, $AP_{D_2} \leftarrow \{\}$
5 **for** *dataset* $d \in \{D_1, D_2\}$ **do**
6 Split d into K equal folds $d_{1,...,K}$
7 **for** $k \in \{1, ..., K\}$ **do**
8 Put aside fold d_k as $test_k$
9 Assign the remaining folds to $train_k$
10 Sample $tune_k \subseteq train_k$
11 $AP_{best} \leftarrow 0$
12 **for** $i \in \{1, ..., C\}$ **do**
13 Sample config c_i from S using the chosen optimization strategy
14 Split $tune_k$ into $tune_{train}$, $tune_{val}$
15 Train A with config c_i on $tune_{train}$
16 Compute AP on $tune_{val}$
17 **if** $AP > AP_{best}$ **then**
18 $c_k = c_i$, $AP_{best} = AP$
19 **repeat** n **times**
20 Change random seed
21 Train A with config c_k on $train_k$
22 Compute AP on $test_k$
23 $AP_d \leftarrow AP_d \cup AP$
24 Compute t-test and p-value on (AP_{D_1}, AP_{D_2})
25 **return** *argmax*($\overline{AP}_{D_1}$, $\overline{AP}_{D_2}$); $|\overline{AP}_{D_1} - \overline{AP}_{D_2}|$; *p-value.*

problem complexity. These could be fundamental updates in the protocol itself, such as defenses against traffic analysis. A deep neural network tuned on unprotected traffic is unlikely to possess enough learning capacity to learn the problem on a harder concept of defended traffic and thus needs to be retuned and then retrained. Moreover, there could be changes in the input representation, such as addition or omission of information or input preprocessing. Finally, as is the case in our study, the impact of time and changes in the data collection methodology may affect the data distribution and the overall problem complexity.

Therefore, to properly deploy and compare DeepCorr on our two new datasets, we develop an *attack replication algorithm* for evaluating and comparing a deep learning or machine learning-based attack *on different datasets*. The general replication algorithm is depicted in Algorithm 1, it is applicable to all the aforementioned cases and trivially extends to more than two datasets. Further on we clarify some of the steps of the algorithm and discuss why they are meant to account for the primary sources of evaluation bias.

Hyperparameter Tuning. There are several important considerations regarding tuning. Firstly, while the attack used for replication (e.g., a state-of-the-art DL approach) is the same for both datasets, it has to be re-tuned for each given dataset separately for fair comparison. This is to ensure that each dataset is assessed with its own best model configuration (architecture and hyperparameters). Not only because that would be the procedure followed by a real-world adversary, but also because with the best configuration, we are able to more accurately assess the expressive power of the dataset.

Secondly, we need to consider resource constraints, as the algorithm involves many intensive iterations through the datasets. There is a trade-off between on one hand a thorough evaluation that addresses possible biases, and on the other hand staying within reasonable computational time. In terms of tuning, the amount of required resources is mainly controlled by the size of a search space S, the maximum number of evaluated attack configurations C, and the chosen size of a subset sampled from the training data for tuning (step 10), which could be a smaller fraction or a full training set.

Nevertheless, it is important to use a consistent optimization strategy that determines which hyperparameter configuration is evaluated at each iteration (step 13). The common options are Grid search (exhaustive search), Random search and Bayesian optimization (probabilistic search). We use Bayesian optimization and elect to start from a small model and gradually increase its complexity. This approach is consistent with the heuristic in Bayesian statistics of preferring simpler models when performance is similar, as they are more likely to represent the underlying distribution of data more accurately [17].

Cross-Validation. To counter selection bias which could arise from evaluation on one fixed data split, the algorithm deploys a procedure similar to the classical k-fold cross-validation: different portions of the dataset are used to train, validate and test the model on different iterations [26]. This way we can assess if the results generalize well, i.e., the attack performs better on one of the datasets over most or even all of the folds. This analysis may increase the confidence in the outcome, or on opposite may show that the result is inconclusive, when the behavior is not consistent across the K folds.

Randomization. We also account for the non-deterministic nature of deep neural networks. Namely, we do several runs (n) for each cross-validation fold with a varied random seed, which impacts random initialization of weights in a deep neural network and random data shuffling between training epochs. Experimenting with more random seeds allows to assess stability of the attack and solidifies the observed trends in performance.

Statistical Significance. The algorithm returns the dataset that yielded the highest average attack performance based on the chosen metric, and the difference between the two averages. It is crucial to compute statistical significance to assess generalization and reliability of the findings, instead of reporting best runs or averages alone, which could be random and hence uninformative or misleading. Our algorithm assesses significance by performing a two-sample t-test, which is applicable to comparing performance of data-driven models with equal variances [11].

For completeness, in Appendix C we provide Algorithm 2 – a version of Algorithm 1 that compares *two different data-driven attacks on the same dataset*, to diminish evaluation biases when proposing new attacks.

5.4 Explainability Analysis

Our data collection approach and attack replication algorithm are meant to ensure validity of the conducted comparison between the single- and multi-proxy setups. The evaluation aims to assert one of the two datasets as the harder one to attack, thus indicating a quality difference between the two data collection setups. However, drawing strong claims from numerical performance metrics is challenging, due to the high complexity and limited interpretability of deep neural networks.

In order to further support the results of the evaluation, we conduct additional explainability analysis of the two attack models. We aim to assess whether the models trained on different datasets utilize the inputs differently, e.g., by prioritizing different features.

We conduct explainability analysis through *visualization of activation maps* – a known technique in computer vision that is most commonly applied to Convolutional Neural Networks (CNNs) for image classification [45]. This approach allows to directly analyze the importance of input features for prediction, because it is applied in the first layer of the network, where the input is still interpretable.

Specifically, we look at the output of the first convolutional layer of the network, which produces a volume of activation maps – two-dimensional arrays of activations. These activations are the inner products of the chunk of the input trace and the convolutional filter (kernel). Because the size of one activation map is linear to the size of the input trace, we can relate the highest activation values to certain areas in the original input. Such areas are considered important by the kernels in the first layer. By repeating this process across all activations maps in the first layer and averaging across many input traces, we reveal which areas in the traffic trace the model finds most relevant for correlation. In our experiments we perform this analysis separately for truly associated and non-associated pairs for each dataset and report the results.

6 Experimental Results

In this section, we apply our proposed methodology aiming to compare performance of the state-of-the-art E2E correlation attack on collected datasets.

First we discuss the input format used for learning, which is the same as is used in prior work which evaluates the attack on the public dataset collected through a single proxy. We do not alter this representation. Our main evaluation methodology is applied as described in Section 5. We perform E2E attack replication on both datasets with single- and multi-proxy designs, which includes hyperparameter tuning and cross-validation with training and testing. Finally we report main performance results, scalability results on the bigger dataset, and the explainability analysis achieved through visualization of activation maps.

6.1 Experimental Setup

All deep learning models used in our experiments are implemented with Tensorflow version 2.2.0 and Keras version 2.3.0-tf. We modified the publicly available implementation of DeepCorr [6] to fit our experiments and implemented the replication procedure on top. The experiments are run on a server with 2 NVIDIA GeForce RTX 2080 Ti GPUs each with 12GB of RAM and an Intel Xeon CPU with 256GB of RAM.

Hyperparameter tuning is implemented with Tune [21] – a research platform for distributed model

6 https://github.com/woodywff/deepcorr

selection on top of Ray [23]. We used the HyperOpt [40] algorithm for model selection with Bayesian optimization [4]. We deployed a cluster on 6 Amazon EC2 P3 instances with 8 NVIDIA Tensor V100 GPUs each with 16GB RAM and an Intel Xeon CPU with 488GB RAM.

We make attack tuning, evaluation and analysis code available on our website provided in footnote 1.

Table 3. Architecture and tuned hyperparameters of the evaluated $\mathcal{TDC}$ models on single- and multi-proxy datasets. Fixed kernel sizes: (2, 30),(4, 10). Fixed strides: (2, 1), (4, 1).

Hyperparam	Search space	$\mathcal{TDC}_{SD}$	$\mathcal{TDC}_{MD}$
kernels	[[64, 128],[64, 64] [128, 128]]	[64, 128]	[64, 128]
pool size	{3, 5}	5	5
dense layers	[[1000],[500, 100] [1000, 100]]	[500, 100]	[500, 100]
optimizer	—	Adam	Adam
lr	{0.001, 0.0001}	0.0001	0.0001
batch size	[32, 64, 128]	32	64
dropout	[0.0, 0.4, 0.6]	0.6	0.6
w_p	[1, 2, 5, 25]	5	5
N_{neg}	[9, 49, 199]	49	49

6.2 Data Input Format

Deep learning-based traffic correlation relies on automated feature extraction from a chosen representation of input traffic. In our experiments, we follow the prior work [27] in representing a bidirectional network flow as a time-series that combines packet sizes, timing and direction features. Therefore, a traffic flow F is formatted as a time-series with 4 features: $F = [T^u, T^d, S^u, S^d]$, where T is inter-packet delays, S is packet sizes, and the u and d refer to the uplink (client to server) and downlink (server to client) directions. For all experiments reported in this study, we use maximum 300 packets in each of the 4 dimensions. Note that attack performance may increase with more packets included, however, to deploy our extensive evaluation methodology for comparing two E2E correlation datasets, one fixed length is considered representative. We also filter out traces with less than 100 packets in the uplink or downlink direction, as the prior work demonstrated that short traces impede the learning capacity of the attack.

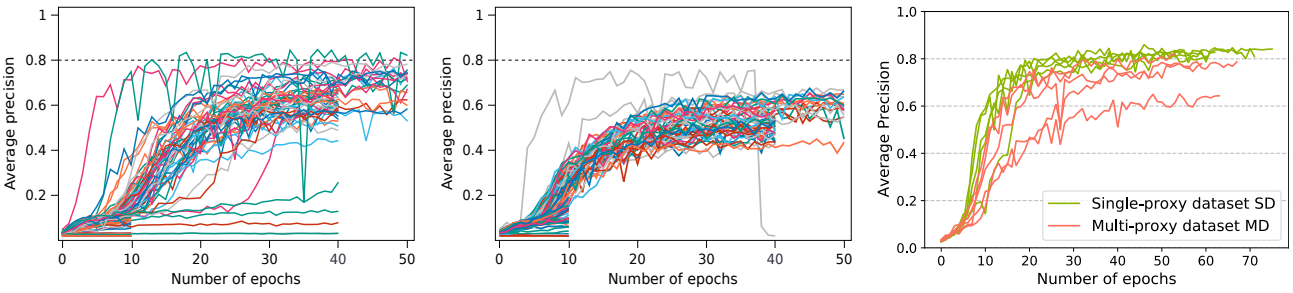
Therefore, a deep learning attack attempts to correlate a pair of traffic segments F_i and F_j, where segment i is captured at a controlled Tor entry guard and segment j is captured at a controlled exit relay (or a proxy server in a research setting). And thus one input instance becomes of shape (8, 300): $F_{i,j} = [T_i^u, T_j^u, T_i^d, T_j^d, S_i^u, S_j^u, S_i^d, S_j^d]$.

6.3 Attack Replication – Tuning

In this part of evaluation we zoom in to the hyperparameter tuning (HT) part of the Algorithm 1 (steps 10-18). We tune a given deep neural network individually on each dataset **SD** and **MD**. We explore two different kinds of search spaces: those informed by the original DeepCorr model $\mathcal{DC}$ (which we found to be visibly prone to overfitting), and those that drastically reduce complexity of the network to optimize evaluation. We call the reduced architecture $\mathcal{TDC}$ - Tiny DeepCorr. The main difference between $\mathcal{TDC}$ and $\mathcal{DC}$ is in the number of kernels used by each convolutional layer. We observe that decreasing the first layer from 2000 to 64 or 128 kernels, and the second layer from 800 to 128 kernels yields a significant multi-fold improvement in speed (9 hours training time down to 20 minutes, 13 hours testing time down to 40 minutes) and easier interpretable activation maps, while the average AP decreases by only $\approx 3\%$. Since we are mostly interested in reliably evaluating the relative performance of the attack on two datasets, we consider this performance drop justified. For transparency, we tune both models and report $\mathcal{DC}$ results in the Appendix A, but we focus the rest of the evaluation on the more optimal design – $\mathcal{TDC}$.

When tuning $\mathcal{TDC}$, apart from reducing the complexity of the network, we also vary the main learning and regularization parameters and the amounts of kernels and nodes for each layer. In order for the neural networks to process this input format in the way intended by DeepCorr, we preserve the original kernel width and the stride values in both convolutional layers. This way the timing and size features are analyzed independently.

Next, we vary the tunable N_{neg} parameter introduced by DeepCorr: it defines the number of non-associated pairs per each truly associated pair provided as negative examples to the network during training. A smaller N_{neg} results in less training data available to the network, but a bigger N_{neg} contributes to a more pronounced imbalance in the data between positive and negative samples, which also has an adverse impact on learning. In order to compensate for this imbalance, we propose to account for the imbalance in training data by using a *weighted* loss function. This is a known mechanism which modifies a learning process by augmenting the loss function with a predefined class weight value to



(a) Validation AP of tested configurations on single-proxy dataset SD.

(b) Validation AP of tested configurations on multi-proxy dataset MD.

(c) Validation AP during training.

Fig. 6. Hyperparameter tuning $\mathcal{TDC}$: average precision (AP) of top-120 configurations over 50 epochs on (a) SD and (b) MD. Note that runs are terminated early when they do not progress well enough. (c) Cross-validation of $\mathcal{TDC}$ on both datasets with seed 10.

penalize the error on the underrepresented class (truly-associated pairs). So we tune the weight of the positive class w_p within the range from 1 (standard non-weighted loss) to N_{neg} (achieves a fully balanced loss).

The resulting search spaces for $\mathcal{TDC}$ are listed in Table 4 columns 1 and 2. We limit the number of experiments to $C = 200$ various configurations (combinations of an architecture and hyperparameters) for each dataset (and to 24 for the much more computationally intensive $\mathcal{DC}$). While exhaustive HT of deep neural networks is out of reach, the automated framework we use adopts Bayesian optimization to guide the search of best parameters. To decrease computational costs of the algorithm, we run HT once on a subset of approximately $5,000$ traces (step 10).

Columns 3 and 4 in Table 4 report the final best configurations for single- and multi-proxy, while the top-5 configurations are listed in Table 6 in Appendix B. As explained in Section 5, we select the optimal configuration based on the top achieved AP.

Figure 6 illustrates the training process for the top-120 runs on **SD** (a) and **MD** (b) in terms of AP on the validation data (convergence in terms of loss can be found in Appendix B). There are several conclusions to derive from the images. First, as expected, the loss and AP are not linearly related: at first AP grows with the minimized validation loss, but then often stabilizies even when the loss starts to increase again (indicating overfitting). The dual-optimization problem is not trivial, and finding the most optimal point in training the attack deserves further exploration. For our experiments, it suffices to select the optimal point in a consistent way for comparing single- to multi-proxy data.

The second and main observation is that in general, tuning on **MD** presents a harder challenge than tuning on **SD** with the same search space. We see that $\mathcal{TDC}_{SD}$ performance is predominantly higher, and the model reaches convergence on earlier epochs. Overfitting is also more present with **SD**. We also notice that some of the best runs for **MD** require more kernels in the first convolutional layer, showing a demand for higher learning capacity. All this is the first indication that the noise produced in the single-proxy setup may indeed increase feasibility of traffic correlation.

We denote the best configuration on the single-proxy data $\mathcal{TDC}_{SD}$ and on the multi-proxy data – $\mathcal{TDC}_{MD}$. While learning from the two datasets presents tasks of varying difficulty, the nature of the problem remains the same such that $\mathcal{TDC}_{SD}$ and $\mathcal{TDC}_{MD}$ are very similar: the only difference is in batch size. It is theoretically possible that further advanced tuning of $\mathcal{TDC}_{MD}$ could potentially improve the attack, e.g., by increasing complexity and employing additional regularization techniques to counteract overfitting. However, our HT experiment succeeds in showing that *the dataset generated in a more realistic multi-proxy setup presents a harder problem for learning.*

6.4 Attack Replication – Training & Test

In the main stage of attack replication (steps 19-23 and 24-25 of Algorithm 1), we estimate and compare attack performance on both datasets by using the whole amount of traffic traces and the best model configurations. Both **SD** and **MD** consist of $N = 7,948$ truly-associated pairs of traces. We perform 5-fold cross-validation ($k = 5$) where for each fold we train and validate on 80 % of correlated traces with $N_{neg} = 49$ negative examples each ($0.8N \times (1 + N_{neg}) = 317,900$) and test on 20 % of positive traces paired with each other ($0.2N \times (0.2N - 1) = 2,526,510$ traces). For each

Table 4. E2E attack performance of the tuned $\mathcal{TDC}$ models on the single- and multi-proxy datasets **SD** and **MD**. AP_{val} is computed on validation data of size $N \times N_{neg}$ ($15,895$ for **SD** and **MD**, and $50,000$ for $\mathbf{MD}_{1mln}$). Other metrics are computed on the test set of $N \times N$ traces (2.5 million for **SD** and **MD**, and 25 million for $\mathbf{MD}_{1mln}$), imitating actual attack. AP indicates the area under PR-curve. TPR, FPR and Pr are first optimized for a low $FPR = 0.001$ then for a high $TPR = 80\,\%$.

Model	$AP_{val}(\%)$	$AP_{test}(\%)$	$TPR_{FPR=0.001}(\%)$	$Pr_{FPR=0.001}(\%)$	$FPR_{TPR=80\,\%}$	$Pr_{TPR=80\,\%}$
$\mathcal{TDC}_{SD}$	79.22 ± 6.80	24.55 ± 6.98	47.76 ± 9.01	21.40 ± 3.27	0.0071 ± 0.003	7.68 ± 2.82
$\mathcal{TDC}_{MD}$	71.27 ± 9.84	13.83 ± 6.60	32.43 ± 10.59	15.02 ± 5.19	0.0154 ± 0.008	4.08 ± 1.98
$p-value$	0.007568	0.000028	–	–	–	–
$\mathcal{TDC}_{MD-1mln}$	95.3 ± 1.2	5.099 ± 1.22	35.93 ± 2.77	39.37 ± 1.75	0.775 ± 1.12	7.09 ± 2.03

fold, we also repeat the evaluation $n = 3$ times with random initialization. The final result for each dataset is therefore averaged over $k \times n = 15$ runs, aiming to account for potential selection bias and non-deterministic behavior of deep learning models. Having more samples should allow to analyze significance of the results.

Same as with HT, we use AP for early stopping, i.e., we obtain the weights of a model that yield the highest AP (area under the PR-curve). The question is then whether these weights generalize well to the unseen data amounting $2,5$ million pairs of traces. Table 4 shows the final performance metrics (extended version can be found in Appendix B). Figures 6a and 6b depict the training process of every run for **SD** and **MD** in terms of AP and validation loss. Overall, E2E attack correlation is more successful on the single-proxy dataset: we observe the difference in AP means of 7.95 %. For the same optimized FPR of 0.001, TPR on **MD** drops by 15.33 %, while for the optimized TPR of 80 %, FPR increases from 0.007 % to 0.015 %, which translates to 100 % more false positives. $\mathcal{TDC}_{\mathcal{SD}}$ also converges faster and appears to be more prone to overfitting.

The difference between means obtained through experiments is not informative on its own, as it could be due to random variations or chance. To show statistical significance of this comparison, we apply the two-sample Welch t-test (for unequal standard deviations) to compute the expected difference between two populations' means. If according to this test the difference is significant (e.g., p-value lower than 0.05), we can assume even on basis of these limited observations that the observed trend would generalize well to other experiments and data splits. Namely, that the attack generally performs worse on **MD**. We apply the test to 40 test set AP values obtained by $\mathcal{TDC}_{\mathcal{SD}}$ and $\mathcal{TDC}_{\mathcal{MD}}$, achieving t-value= 4.866 and p-value= 0.000028, which indicate high statistical significance. In Appendix B, we shed more light on variability of performance on different dataset folds and random seeds.

Scalability. To assess how well the attack scales to a larger dataset, we evaluate $\mathcal{TDC}_{\mathcal{MD}}$ on 25,000 associated pairs from the $\mathbf{MD}_{1mln}$ dataset with $N_{neg} = 5$. The results in Table 4 indicate worsened performance on 25 mln test traces. The experiment illustrates the intrinsic challenge of E2E correlation attacks (cf. Section 5): the growing amount of intercepted traffic traces leads to a quadratic growth of the test set, which in turn causes a sharp decrease in the overall attack performance.

Results. To sum up, based on the extensive tuning and systematic evaluation, we can conclude that the E2E correlation attack performs *significantly better* on **SD** than on **MD**, which corresponds to our initial hypothesis. Namely, the realism issue with the single-proxy data collection setups that was exposed in Section 3 appears to favorably impact estimated performance of data-driven E2E attacks. As this may be due to the noisy timings present in the input, we further explore this connection with explainability analysis.

Explainability. To get the insight behind the observed difference in performance, we compute mean activations maps in the first convolutional layer, separately for correlated and non-correlated inputs, for both datasets. As explained in Section 5, the attacks may assign different levels of importance to various input features or certain areas of input traffic traces, indicating where the signal for traffic correlation is stronger in each dataset.

In Figure 7 we show one example of *the differences between mean activations* obtained from correlated and non-correlated pairs by single and multi-proxy models. Activations produced by $\mathcal{TDC}$'s first layer have shape $(4, 64, 271)$ (features, kernels, activation positions), and we plot the two computed timing features $[T^u_{ij}, T^d_{ij}]$ for positive and negative pairs. As the visual interpretation is limited, we also calculate a sum of the differences for each pair of models - the *total difference in mean activations* (see Table 8 in Appendix D). The first observation to be made here is that in general, uplink timings appear to be more important to correlation than

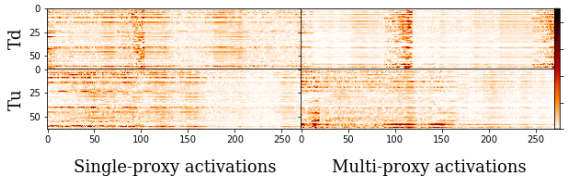


Fig. 7. Differences between mean activation maps computed over 1590 correlated (pos) and non-correlated (neg) traces. Left: single-proxy, right: multi-proxy. Datasets fold: 4, seed: 10. Uplink timings: T^u, downlink: T^d. Dark color indicates higher intensity.

downlink timings. This is surprising, as downlink information contains server responses, i. e. is richer and of greater volume. As there are no studies that look into explainability of DL-based traffic correlation, more in-depth analysis is required to explain the mechanism through understand the exact involvement of kernels. Requests could facilitate E2E correlation if they were indeed more unique than responses. However, that does not have to be the case: DL-based nature of the attack may be causing the attention skew from downlink to uplink data simply because it is easier to analyze. Neural networks are known for simplicity bias [37]: they can exclusively rely on the simplest features and remain invariant to other more complex predictive patterns. To verify this idea, we suggest that future work needs to investigate the inner workings of DL-based traffic correlation attacks, similar to the recent work for WF [9].

Secondly, we compare the activation differences of SD and MD and notice that on the single-proxy data, uplink timings are in general more impactful on the attack. This preliminary analysis indicates stronger timing signal in single-proxy traces; however, more research is needed to study the exact mechanism. We refer the reader to Appendix D for more details.

Overall, the problem of reliable and robust AI evaluation remains an open research question. The guidelines proposed in this paper address common sources of sampling and evaluation bias in experimental setups and can be further improved in future research.

7 Discussion

Here we discuss limitations and future work directions.

Representativeness of Traffic. Our dataset consists of website requests that only consider visits to the main homepage while omitting the pages within. This simplification allows us to create a dataset comparable with prior work on E2E correlation, so the main quality difference is manifested through much more accurate temporal characteristics. However, we did not aim to move the dataset beyond the state-of-the-art in terms of representativeness of simulated human behavior. While our specific analysis of E2E attacks remains largely undisturbed by this limitation and yields novel results, the browsing activity stays within the abstraction of homepage visits, as is common for many other known WF and E2E datasets [27, 29, 33, 38, 39]. There have been notable advances in the area of approximating real Tor user profiles and addressing complex website infrastructures during traffic generation for WF [31]. Using our novel setup to collect an E2E (and WF) dataset that is also more representative of human behavior and assess the impact of traffic analysis attacks would be a challenging but valuable extension.

Proxy-Related Overhead. A proxy-based data collection setup for end-to-end correlation allows to estimate attack success rates in a research setting without compromising safety of Tor users. In this study we show that the novel multi-proxy data collection setup reduces the timing overhead introduced by the state-of-the-art single-proxy design. However, the impact of *any* involvement of proxies on traffic metadata, in comparison to the real-world Tor, remains an open research question. The complex proxy-related overhead and its indirect impact on traffic correlation feasibility can technically only be measured by collecting real exit traffic. As this is associated not only with the loss of node location diversity but also with strong ethical concerns, it would be pertinent to minimize privacy risks by consulting the community for coordination of effort (e. g. through the Tor Research Safety Board[7]).

Ethics Considerations. Work on traffic analysis attacks requires realistic datasets that resemble Tor's daily transmission characteristics. Measurements in the real network must not harm or deanonymize Tor users by any means or affect the overall performance of the system. For all decisions we made when designing our real-world experimental setup, we closely followed Tor's guidelines for ethical research [41]. Although frameworks like the Shadow simulator [16] are able to mimic transmissions through Tor, we collected real-world traffic to allow for comparison with previous research.

Defenses Against Traffic Analysis. The analysis in our paper is focused on unprotected traffic. While this is sufficient for validating our proposed data collec-

7 https://research.torproject.org/safetyboard/

tion setup, evaluation of known protection schemes is a worthwhile future research direction. To this end, we make our dataset on closed-world and open-world WF and E2E attacks publicly available to foster research on a joint evaluation of defensive schemes against both attacks. To the best of our knowledge, our dataset is the first effort in combining both threat models in one large-scale data collection setup, enabling this exploration.

Onion Services. When browsing a typical website, it could be indicating that it can be accessed over an onion service, using the `Alt-Svc` response header. In our setup, the targeted websites observed incoming connections from our proxies, hosted on the network of a cloud provider. Consequently, the websites would not send the `Alt-Svc` header, and thus our Tor Browser client did not attempt to access the them over the onion service. Moreover, because the Tor Browser used a proxy chain, it would not be able to resolve or connect to `.onion` addresses. Consequently, our data collection setup deviates from a real-world data capture for websites that would redirect users to an onion service.

In practice, we find that Cloudflare, a large CDN provider, provides an optional feature for all the websites it hosts that allows them to be accessible via an onion service [34]. However, because of our proxy-based setup, no traffic was sent to onion services, possibly affecting the representativeness of our dataset. More specifically, we captured data at our proxy that might have otherwise been sent to an onion service, and thus would not be captured by an adversary who only controlled the client's guard node and the exit node. In a real-world attack, the adversary needs to take the asymmetry of requests sent to exit nodes and onion services into account. In preliminary tests, we found that onion services are rarely contacted when loading a new website. Nevertheless, an extensive evaluation of the impact of loading resource from onion services on attacks is an important aspect to be addressed in future work.

8 Related Work

Our work builds on DeepCorr, the first traffic correlation attack that leverages deep learning [27]. While the system undoubtedly outperforms existing approaches to flow correlation, we critically replicated its evaluation focusing on a more realistic research setup for modeling the attack. Our study provides additional novel insights on performance evaluation, tuning, scalability and explainability of the DL-based attack. Concurrently to our work, Oh et al. [12] have proposed a novel end-to-end correlation attack by utilizing metric learning techniques. As their learning approach is fundamentally different from the CNN used in DeepCorr, an interesting research direction would be to investigate performance of their attack in our proposed multi-proxy setup.

Previously, critical evaluations of attack techniques in the light of their real-world applicability have been conducted in the context of website fingerprinting [19], with focus on access to traffic as operational requirements for end-to-end confirmation [18, 35], and for de-anonymization attacks targeted at specific users [15].

As a widely adopted technique for end-to-end processing of time-series, DL is favored for unprecedented performance and efficiency. With the analysis of Tor traffic being one of the most prominent applications in privacy [30, 33], DL was also employed, e.g., for protocol detection [44], encrypted video stream analysis [36], and for inferring (mobile) applications from encrypted traffic [2]. Limitations and challenges of DL evaluations in security have been recently explored by Arp et al. [3]. However, most progress on sound practices for DL evaluations are conducted in the AI domain [7, 8, 25].

9 Conclusion

In this work, we introduced new methodologies for data-driven end-to-end correlation attacks on Tor. We proposed a new experimental setup that allows to collect Tor traffic with more realistic timing characteristics by minimizing the additional overhead in proxy-based end-to-end measurements. Furthermore, we introduced a systematic replication strategy along with appropriate evaluation metrics to allow for a fair comparison of data-driven attacks on novel data. Our empirical results demonstrate the relevance of the suggested multi-proxy design: we find that the novel end-to-end correlation dataset that contains more realistic timing measurements also presents a significantly harder learning problem. This indicates that utilizing multiple proxies approximates attacker capabilities more accurately than prior work – a positive step towards realistic attack evaluation. Finally, we open source the research code and the multi-proxy dataset to enable further research on traffic correlation and website fingerprinting attacks.

Acknowledgments

We would like to thank our shepherd, Tobias Pulls, and the anonymous reviewers for their valuable feedback. This research was funded by Research Fund KU Leuven and Flemish Research Programme Cybersecurity. The data collection was supported by the Center for Cyber Security at New York University Abu Dhabi (NYUAD).

References

[1] G. Acar, M. Juarez, and individual contributors, "tor-browser-selenium - Tor browser automation with Selenium," https://github.com/webfp/tor-browser-selenium, 2020.

[2] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning," in *2018 Network traffic measurement and analysis conference (TMA)*. IEEE, 2018, pp. 1–8.

[3] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, C. Wressnegger, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *USENIX Security Symposium*, 2021.

[4] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International conference on machine learning*. PMLR, 2013, pp. 115–123.

[5] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-cnn: A data-efficient website fingerprinting attack based on deep learning," *Privacy Enhancing Technologies Symposium*, vol. 4, pp. 292–310, 2019.

[6] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted HTTP streams," in *Workshop on Privacy Enhancing Technologies*, ser. PET '05. Cavtat, Croatia: Springer, May 2005, pp. 1–11.

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[8] M. F. Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? A worrying analysis of recent neural recommendation approaches," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 101–109.

[9] T. Dahanayaka, G. Jourjon, and S. Seneviratne, "Dissecting traffic fingerprinting CNNs with filter activations," *Computer Networks*, p. 108770, 2022.

[10] G. Danezis, "The traffic analysis of continuous-time mixes," in *Workshop on Privacy Enhancing Technologies*, ser. PET '04. Toronto, Canada: Springer, May 2004, pp. 35–50.

[11] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[12] S. Eun Oh, T. Yang, N. Mathews, J. K. Holland, M. S. Rahman, M. Wright, and N. Hopper, "DeepCoFFEA: Improved flow correlation attacks on Tor via metric learning and amplification," in *43rd IEEE Symposium on Security and Privacy (S&P)*, May 2022.

[13] J. Hayes and G. Danezis, "Guard sets for onion routing," ser. PoPETS '15, vol. 2015, no. 2. De Gruyter, 2015, pp. 65–80.

[14] ——, "k-Fingerprinting: A robust scalable website fingerprinting technique," in *USENIX Security Symposium*, ser. USENIX '16. Washington, DC, USA: USENIX Association, Aug. 2016, pp. 1187–1203.

[15] A. D. Jaggard and P. Syverson, "Onions in the crosshairs: When the man really is out to get you," in *Workshop on Privacy in the Electronic Society*, ser. WPES '17. Dallas, TX, USA: ACM, 2017, pp. 141–151.

[16] R. Jansen and N. Hopper, "Shadow: Running Tor in a box for accurate and efficient experimentation," in *Network and Distributed System Security Symposium*, ser. NDSS '12. San Diego, CA, USA: The Internet Society, Feb. 2012.

[17] W. H. Jefferys and J. O. Berger, "Ockham's razor and bayesian analysis," *American Scientist*, vol. 80, no. 1, pp. 64–72, 1992.

[18] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on Tor by realistic adversaries," in *ACM Conference on Computer and Communications Security*, ser. CCS '13. Berlin, Germany: ACM, Nov. 2013, pp. 337–348.

[19] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *ACM Conference on Computer and Communications Security*, ser. CCS '14. Scottsdale, AZ, USA: ACM, Nov. 2014, pp. 263–274.

[20] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, ser. NDSS 2019, Feb. 2019.

[21] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[22] N. Mathewson and R. Dingledine, "Practical traffic analysis: Extending and resisting statistical disclosure," in *Workshop on Privacy Enhancing Technologies*, ser. PET '04. Toronto, Canada: Springer, May 2004, pp. 17–34.

[23] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging AI applications," in *USENIX Symposium on Operating Systems Design and Implementation*, ser. OSDI'18, 2018, pp. 561–577.

[24] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *IEEE Symposium on Security and Privacy*, ser. SP '05. Oakland, CA, USA: IEEE, May 2005, pp. 183–195.

[25] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," in *European Conference on Computer Vision*. Springer, 2020, pp. 681–699.

[26] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine learning*, vol. 52, no. 3, pp. 239–281, 2003.

[27] M. Nasr, A. Bahramali, and A. Houmansadr, "DeepCorr: Strong flow correlation attacks on tor using deep learning," in *ACM Conference on Computer and Communications Security*, ser. CCS '18. ACM, 2018, pp. 1962–1976.

[28] M. Nasr, A. Houmansadr, and A. Mazumdar, "Compressive traffic analysis: A new paradigm for scalable traffic analysis," in *ACM Conference on Computer and Communications Security*, ser. CCS '17. Dallas, TX, USA: ACM, Oct. 2017, pp. 2053–2069.

[29] S. E. Oh, N. Mathews, M. S. Rahman, M. Wright, and N. Hopper, "GANDaLF: GAN for data-limited fingerprinting," *Privacy Enhancing Technologies Symposium*, vol. 2, pp. 305–322, 2021.

[30] S. E. Oh, S. Sunkam, and N. Hopper, "p1-fp: Extraction, classification, and prediction of website fingerprints with deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 191–209, 2019.

[31] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at internet scale," in *Network and Distributed System Security Symposium*, ser. NDSS '16. San Diego, CA, USA: The Internet Society, Feb. 2018.

[32] 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, pp. 5–24, 2020.

[33] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Network and Distributed System Security Symposium*, ser. NDSS '18. San Diego, CA, USA: The Internet Society, Feb. 2018.

[34] M. Sayrafi, "Introducing the Cloudflare onion service," https://blog.cloudflare.com/cloudflare-onion-service/, Sep. 2019.

[35] T. Schnitzler, C. Pöpper, M. Dürmuth, and K. Kohls, "We built this circuit: Exploring threat vectors in circuit establishment in Tor," in *IEEE European Symposium on Security and Privacy*, ser. EuroSP '21. IEEE, 2021.

[36] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *USENIX Security Symposium*, 2017, pp. 1357–1374.

[37] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli, "The pitfalls of simplicity bias in neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9573–9585, 2020.

[38] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *ACM Conference on Computer and Communications Security*, ser. CCS '18. ACM, 2018, pp. 1928–1943.

[39] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," in *ACM Conference on Computer and Communications Security*, ser. CCS '19. ACM, 2019, pp. 1131–1148.

[40] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[41] The Tor Project, "Ethical Tor research: Guidelines," Jan. 2019, https://blog.torproject.org/blog/ethical-tor-research-guidelines.

[42] ——, "The Onion Router," Feb. 2021, https://www.torproject.org.

[43] T. Wang, "High precision open-world website fingerprinting," in *IEEE Symposium on Security and Privacy*, IEEE. The Internet Society, 2020, pp. 152–167.

[44] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, vol. 24, no. 11, pp. 1–10, 2015.

[45] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

Appendix

A Original DeepCorr Performance

We list the search spaces and chosen hyperparameters for the original DeepCorr architecture in Table 5. Hyperparameter tuning process of $\mathcal{DC}$ in Figure 8 depicts average precision (AP) of different configurations evolving over 30 epochs. Note that the runs are terminated early when they stop progressing. The longest runs take over 8 hours, so we set patience for early stopping to 5 epochs (vs. 10 epochs for $\mathcal{TDC}$). We refer the reader to our website in footnote 1 for more details on reproducing original DeepCorr on public and novel datasets.

Table 5. Hyperparameters of the $\mathcal{DC}$ model on the multi-proxy dataset and final chosen configuration for $\mathcal{TDC}$. Fixed kernel sizes: (2, 30),(4, 10), strides: (2, 1), (4, 1).

Hyperparam	Search space	$\mathcal{TDC}_{SD}$	$\mathcal{TDC}_{MD}$
kernels	[2000,800]	[64, 128]	[64, 128]
pool size	{3, 5}	5	5
dense layers	{[2000],[2000, 500]}	[500, 100]	[500, 100]
optimizer	—	Adam	Adam
lr	{0.001, 0.0001}	0.0001	0.0001
batch size	[32, 64, 128]	32	64
dropout	[0.0, 0.4, 0.6]	0.6	0.6
w_p	[1, 2, 5, 25]	5	5
N_{neg}	[9, 49, 199]	49	49

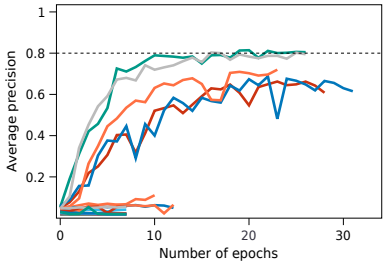


Fig. 8. Validation AP of tuned DeepCorr configurations.

B Tuning & Training Details

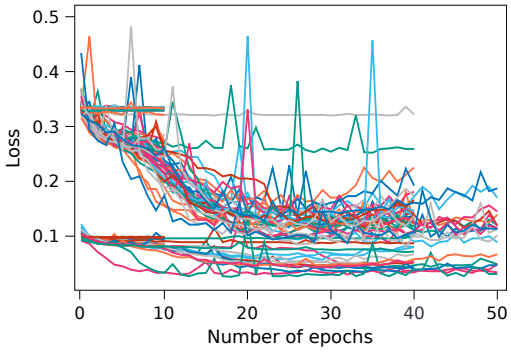
Table 6 lists 5 top tuning experiments for $\mathcal{TDC}$ models on both datasets (based on the search spaces defined earlier in Table 3). While the tuning framework sorts the runs by their final validation AP (i. e., at the last or early stopping epoch), we choose the best configuration based on the maximum validation AP over the whole training process. Note that there are several identical configurations among the top 10. This is due to the framework evaluating some of the most promising hyperparameter configurations repeatedly, in order to account for variability of the training process and verify superiority of given parametrizations.

Figure 9 depicts the complementing part of Figure 6 in Section 6, with the validation loss instead of AP. Table 7 provides information on all individual evaluation runs of $\mathcal{TDC}_{\mathcal{SD}}$ and $\mathcal{TDC}_{\mathcal{MD}}$ on five folds of each dataset over four varied random seeds. These results illustrate high variability of deep neural networks (even on the *exactly same* data and configurations), thus underlining the importance of Algorithm 1.

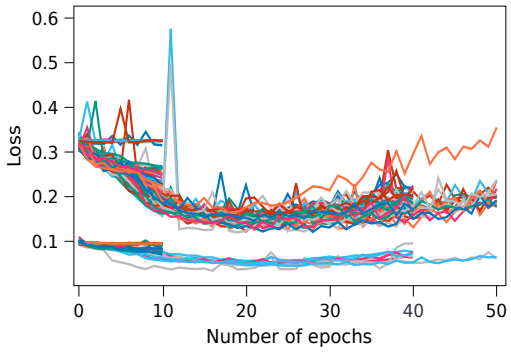
Table 6. Top performing hyperparameter tuning runs for the $\mathcal{TDC}$ model on a subset of the single-proxy dataset SD (top) and multi-proxy dataset MD (bottom). Dropout $= 0.6$. Sorted by the final AP. The best configuration is chosen based on the maximum achieved AP, marked in bold.

	lr	batch size	w_p	dense layers	kernels	loss	final AP
SD	0.001	128	1	[500, 100]	[128, 128]	0.031	0.821
	0.0001	**32**	**5**	**[500, 100]**	**[64, 128]**	0.089	0.748
	—"—	32	1	—"—	—"—	0.034	0.747
	—"—	64	5	—"—	—"—	0.114	0.744
	—"—	64	1	—"—	—"—	0.047	0.740

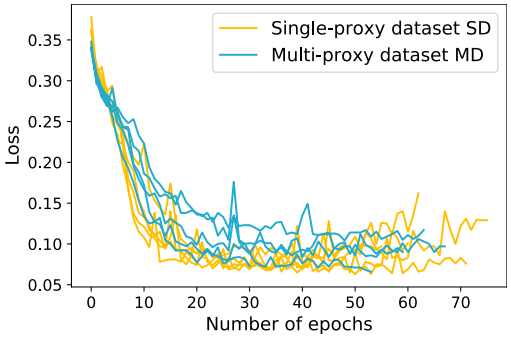
	lr	batch size	w_p	dense layers	kernels	loss	final AP
MD	**0.0001**	**64**	**5**	**[500,100]**	**[64, 128]**	0.184	0.667
	—"—	—"—	5	—"—	[128, 128]	0.192	0.664
	—"—	—"—	1	—"—	[64, 128]	0.065	0.637
	—"—	—"—	1	—"—	—"—	0.061	0.635
	—"—	—"—	5	—"—	—"—	0.178	0.608



(a) Validation loss of tested configurations on single-proxy dataset SD.



(b) Validation loss of tested configurations on multi-proxy dataset MD.



(c) Validation loss during training.

Fig. 9. Hyperparameter tuning $\mathcal{TDC}$: validation loss of top-120 configurations over 50 epochs on (a) SD and (b) MD. Note that runs are terminated early when they do not progress well enough. (c) Cross-validation of $\mathcal{TDC}$ on both datasets with seed 10.

Table 7. Detailed attack results for single-proxy model $\mathcal{TDC}_{SD}$ (top half) and multi-proxy model $\mathcal{TDC}_{MD}$ (bottom half) with cross-validation and varied seeds. Two outlier runs ($\mathcal{TDC}_{MD}$, seed=101, fold=2,4) marked red were excluded from computation of mean and statistical significance.

$\mathcal{TDC}_{SD}$				$FPR = 0.001$		$TPR = 80\,\%$	
	Test fold	$AP_{val}(\%)$	$AP_{test}(\%)$	$TPR(\%)$	$Pr(\%)$	FPR	$Pr(\%)$
Seed = 1	■□□□□	77.67	26.08	48.55	21.83	0.0066	7.05
	□■□□□	77.34	17.98	38.24	18.04	0.0091	5.19
	□□■□□	86.13	27.24	53.02	23.29	0.0046	9.71
	□□□■□	64.87	11.79	31.82	15.42	0.0135	3.57
	□□□□■	81.3	33.27	55.97	24.3	0.0051	8.9
Seed = 10	■□□□□	84.69	29.09	55.47	24.14	0.0039	11.34
	□■□□□	82.57	24.46	47.74	21.51	0.0059	7.75
	□□■□□	85.99	27.99	54.4	23.91	0.0046	9.68
	□□□■□	83.44	34.17	59.12	25.28	0.0036	12.01
	□□□□■	84.65	29.3	55.28	24.1	0.0048	9.48
Seed = 101	■□□□□	67.49	14.38	33.9	16.31	0.0115	4.16
	□■□□□	81.2	29.95	53.52	23.65	0.0054	8.4
	□□■□□	84.76	26.04	52.08	22.99	0.0047	9.56
	□□□■□	71.83	16.05	36.86	17.47	0.0114	4.19
	□□□□■	76.34	17.51	39.87	18.59	0.0109	4.37
Seed = 102	■□□□□	66.05	13.01	31.76	15.47	0.0143	3.36
	□■□□□	83.69	31.74	55.03	23.98	0.0047	9.54
	□□■□□	84.01	27.61	49.87	22.39	0.0059	7.82
	□□□■□	84.04	30.34	56.16	24.35	0.0037	11.86
	□□□□■	76.43	23.1	46.6	21.06	0.0081	5.8
	Mean ± std	**79.22 ± 6.80**	**24.55 ± 6.98**	**47.76 ± 9.01**	**21.40 ± 3.27**	**0.0071 ± 0.003**	**7.68 ± 2.82**

$\mathcal{TDC}_{MD}$				$FPR = 0.001$		$TPR = 80\,\%$	
	Test fold	$AP_{val}(\%)$	$AP_{test}(\%)$	$TPR(\%)$	$Pr(\%)$	FPR	$Pr(\%)$
Seed = 1	■□□□□	61.27	6.73	20.38	10.45	0.0281	1.74
	□■□□□	60.2	6.21	19.87	10.24	0.0246	1.98
	□□■□□	62.2	9.05	26.04	13.07	0.0196	2.48
	□□□■□	84.75	24.23	47.92	21.62	0.0063	7.27
	□□□□■	75.38	19.37	40.94	18.99	0.0091	5.18
Seed = 10	■□□□□	81.85	14.41	35.03	16.73	0.01	4.74
	□■□□□	75.43	15.83	34.84	16.65	0.0104	4.56
	□□■□□	78.44	21.45	46.42	21.01	0.0065	7.12
	□□□■□	65.13	7.7	22.58	11.44	0.0237	2.06
	□□□□■	78.75	16.59	37.99	17.86	0.009	5.25
Seed = 101	■□□□□	79.44	18.33	41.13	19.11	0.0084	5.58
	□■□□□	22.35	0.96	4.97	2.78	0.1313	0.38
	□□■□□	51.77	4.84	15.66	8.35	0.0343	1.43
	□□□■□	13.19	0.53	3.02	1.71	0.2103	0.24
	□□□□■	76.96	18.37	37.99	17.93	0.0101	4.71
Seed = 102	■□□□□	60.97	6.26	21.57	11.1	0.0243	2
	□■□□□	68.64	9.81	26.92	13.35	0.0173	2.8
	□□■□□	60.83	6.93	21.19	10.83	0.0216	2.25
	□□□■□	80.4	19.8	42.83	19.7	0.0077	6.04
	□□□□■	80.51	23.03	44.47	20.28	0.0074	6.32
	Mean ± std	**71.27 ± 9.84**	**13.83 ± 6.60**	**32.43 ± 10.59**	**15.02 ± 5.19**	**0.0154 ± 0.008**	**4.08 ± 1.98**

C Comparison of Attacks

With Algorithm 2, we present our proposal for replication and scientific comparison of DL-based attacks against Tor, including E2E traffic correlation and website fingerprinting. The main intended usage is a systematic evaluation of novel attacks against the state-of-the-art. This algorithm follows the general logic of Algorithm 1 and has similar motivation. Namely, in order to aim for reliable conclusions on superiority of one data-driven approach over the other based on limited empirical evidence, it is essential to account for major sources of potential bias and variability. While cross-validation is usually used to diminish possible sampling bias, other aspects are not very often considered in the literature: (1) re-tuning on data that was not used for the original tuning of the state-of-the-art attack; (2) coordination of tuning processes for both attacks in terms of time and memory resources, available search spaces, and conditions and metrics for early stopping; (3) randomization in training of neural networks, which is manifested e. g. in random weight initialization and random data shuffling during stochastic gradient descent. And finally, the algorithm suggests a Welch t-test for statistical significance in case of unequal standard deviations.

As already discussed in Section 5, exhaustive evaluations are unattainable. The ideal evaluation may become too computationally intensive and tedious to perform, especially with large datasets and models. This often prompts researchers to compromise on some aspects of a thorough evaluation, to obtain a desirable cost/benefit trade-off in terms of computational costs and reliability of results. In such situations, we encourage to explicitly acknowledge which parts of the algorithm have been omitted or reduced for a computationally feasible comparison. For example, compared attacks might have had inconsistent tuning conditions, or performance samples of one attack were collected over a bigger range of random seeds because of a higher variance. Reporting these experimental limitations along with the results is instrumental for transparency and for estimation of certainty in a given outcome.

Algorithm 2. Replication and comparison of two ML/DL attacks on the same dataset. Here, with AP as a target metric and t-test for statistical significance.

Input: Dataset D; ML/DL attacks A_1, A_2; hypeparameter search spaces S_{A_1}, S_{A_2}.
Output: Attack with the highest mean performance on D; diff between means; statistical significance.

Set number of cross-validation folds K
Set maximum numbers of attack configurations C_{A_1}, C_{A_2} (or consistent time and memory resources)
Set number of randomized training runs n
Split D into K equal folds $D_{1,...,K}$
$AP_{A_1} \leftarrow \{\}$, $AP_{A_2} \leftarrow \{\}$
for $k \in \{1, ..., K\}$ **do**
    Put aside fold D_k as $test_k$
    Assign the remaining folds to $train_k$
    Sample $tune_k \subseteq train_k$
    for *attack* $A \in \{A_1, A_2\}$ **do**
        $AP_{best} \leftarrow 0$
        for $i \in \{1, ..., C_A\}$ **do**
            Sample config c_i from S_A using the chosen optimization strategy
            Split $tune_k$ into $tune_{train}$, $tune_{val}$
            Train A with config c_i on $tune_{train}$
            Compute AP on $tune_{val}$
            if $AP > AP_{best}$ **then**
                $c_k = c_i$, $AP_{best} = AP$
        repeat n **times**
            Change random seed
            Train A with config c_k on $train_k$
            Compute AP on $test_k$
            $AP_A \leftarrow AP_A \cup AP$
Compute t-test and p-value on (AP_{A_1}, AP_{A_2})
return *argmax*($\overline{AP}_{A_1}$, $\overline{AP}_{A_2}$); $|\overline{AP}_{A_1} - \overline{AP}_{A_2}|$; *p-value.*

D Explainability Graphs

In Section 6, we perform explainability analysis to find out whether certain timing characteristics are more influential for one data collection setup than the other. Here we give more detailed results of the explainability experiments, for models created with random seeds 1 and 10 and trained on all 5 folds. These experiment provides preliminary ideas on the presence of potential timing bias that may be simplifying traffic correlation in the single-proxy setup. Figure 10 depicts differences in mean activations between truly-associated and non-associated traces. Since the visual analysis is challenging, Table 8 contains total (summed up) differences between these activations maps for both uplink and downlink timings. We observe that uplink timings mostly have much larger impact on single-proxy models than on multi-proxy models, except the two scenarios. This preliminary result could be an indication of subtle timing bias being present in uplink timings of traffic traces collected over the single-proxy setup. To obtain conclusive results, we propose extending this investigation in future research beyond analysis of activation maps.

Table 8. Comparison of the impact of uplink timings vs. downlink timings on the traffic correlation attack by single- and multi-proxy models created with given seeds and folds. For both models, we first show the total difference in mean activations for uplink timings (A_{T^u}), as well as for downlink timings (A_{T^d}), followed by their subtraction ($A_{T^u} - A_{T^d}$). Positive results of subtraction show that uplink values are always larger. Then, the delta column shows how much larger the relative impact of uplink timings is on single-proxy models than it is on multi-proxy models. For reference, the right side of the table lists the corresponding attack performance metrics.

		$\mathcal{TDC}_{SD}$			$\mathcal{TDC}_{MD}$				$\mathcal{TDC}_{SD}$		$\mathcal{TDC}_{MD}$	
	Test fold	A_{T^u}	A_{T^d}	A_{T^u}-A_{T^d}	A_{T^u}	A_{T^d}	A_{T^u}-A_{T^d}	Delta	$AP_{val}(\%)$	$AP_{test}(\%)$	$AP_{val}(\%)$	$AP_{test}(\%)$
Seed = 10	■□□□□	226720.5	101278.8	125441.7	247898.9	179573.3	68325.5	46 %	84.69	29.09	81.85	14.41
	□■□□□	226106.2	112463.7	113642.6	148192.4	76552.0	71640.4	37 %	82.57	24.46	75.43	15.83
	□□■□□	183858.9	85385.1	98473.8	167102.9	113232.8	53870.1	45 %	85.99	27.99	78.44	21.45
	□□□■□	207299.5	80356.2	126943.3	200730.1	124901.6	75828.4	40 %	83.44	34.17	65.13	7.7
	□□□□■	207509.1	90500.0	117009.1	188045.7	146269.1	41776.6	64 %	84.65	29.3	78.75	16.59
Seed = 1	■□□□□	180377.9	98670.1	81707.8	246522.6	154175.9	92346.7	−13 %	77.67	26.08	61.27	6.73
	□■□□□	193831.3	92423.1	101408.2	213124.6	100228.8	112895.8	−11 %	77.34	17.98	60.2	6.21
	□□■□□	177800.0	86152.4	91647.7	155957.5	102699.1	53258.3	42 %	86.13	27.24	62.2	9.05
	□□□■□	191768.9	79840.0	111928.9	190277.7	117061.6	73216.1	35 %	64.87	11.79	84.75	24.23
	□□□□■	170874.1	76537.1	94337.0	176942.0	151082.6	25859.5	73 %	81.3	33.27	75.38	19.37

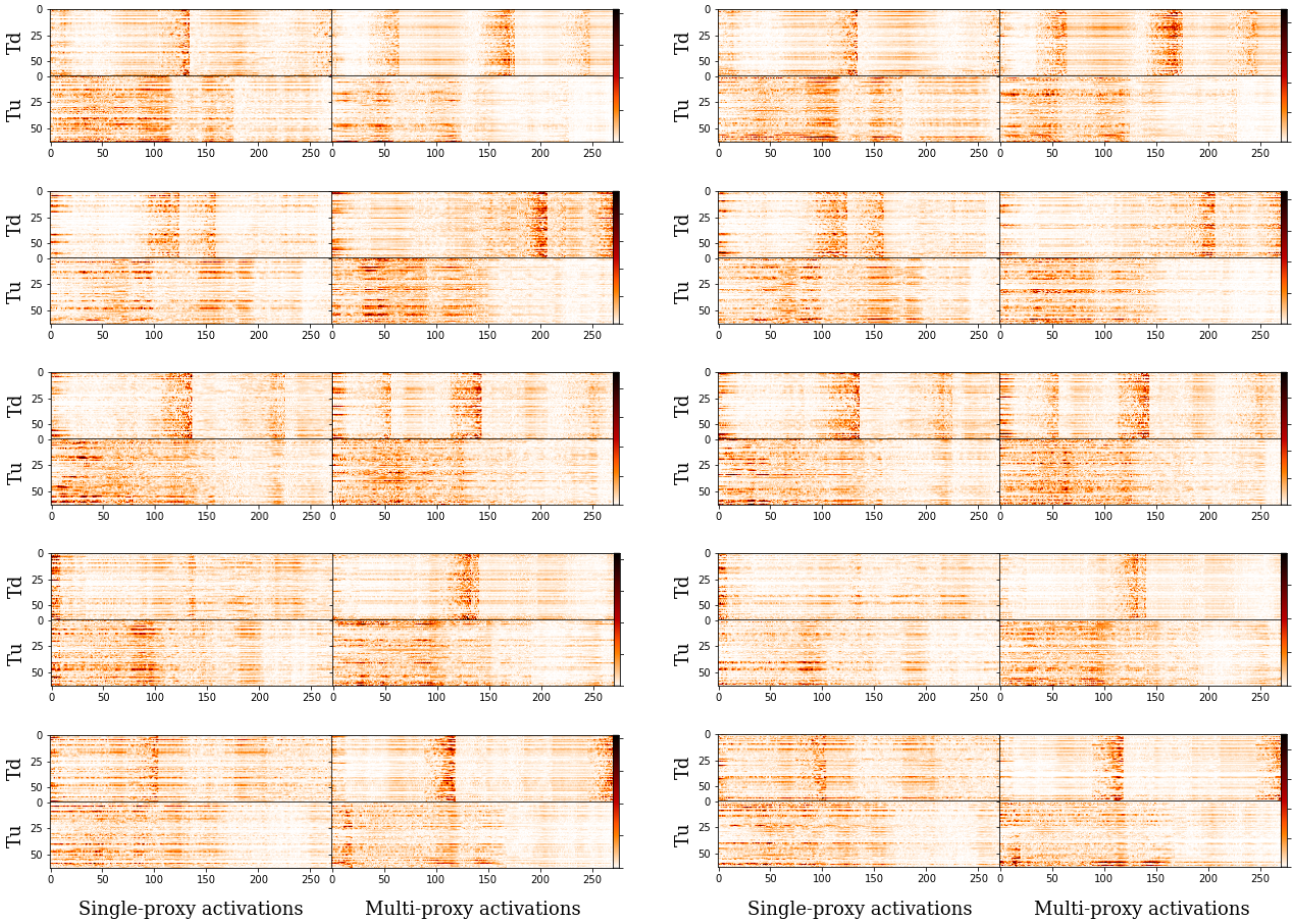


Fig. 10. Differences between mean activation maps computed over 1590 truly-associated (pos) and non-associated (neg) traces. Individual heatmaps display on the horizontal axis single-proxy (left) vs. multi-proxy (right), and on the vertical axis differences between mean activations generated from uplink timings (T^u) vs. downlink (T^d). Darker color indicates higher intensity. All maps in the left column were produced with seed 1 vs. seed 10 for those in the right column. Folds 1 to 5 are arranged from top to bottom.