Proceedings on Privacy Enhancing Technologies ..; .. (..):1–20

Sébastien Henri*, Gines Garcia-Aviles, Pablo Serrano, Albert Banchs, and Patrick Thiran

# Protecting against Website Fingerprinting with Multihoming

**Abstract:** Anonymous communication tools, such as Tor, are extensively employed by users who want to keep their web activity private. But recent works have shown that when a local, passive adversary observes nothing more than the timestamp, size and direction (incoming or outgoing) of the packets, it can still identify with high accuracy the website accessed by a user. Several defenses against these *website fingerprinting* attacks have been proposed but they come at the cost of a significant overhead in traffic and/or website loading time. We propose a defense against website fingerprinting which exploits multihoming, where a user can access the Internet by sending the traffic through multiple networks. With multihoming, it is possible to protect against website fingerprinting by splitting traffic among the networks, i.e., by *removing* packets from one network and sending them through another, whereas current defenses can only add packets. This enables us to design a defense with no traffic overhead that, as we show through extensive experimentation against state-of-the-art attacks, reaches the same level of privacy as the best existing practical defenses. We describe and evaluate a proof-of-concept implementation of our defense and show that is does not add significant loading-time overhead. Our solution is compatible with other state-of-the-art defenses, and we show that combining it with another defense further improves privacy.

**Keywords:** website fingerprinting defense, multihoming, multipath

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

***Corresponding Author: Sébastien Henri:** Cisco Meraki, USA. E-mail: sebastien.henri@meraki.net. Work done primarily at EPFL, Switzerland
**Gines Garcia-Aviles:** University Carlos III of Madrid, Spain. E-mail: gigarcia@it.uc3m.es
**Pablo Serrano:** University Carlos III of Madrid, Spain. E-mail: pablo@it.uc3m.es
**Albert Banchs:** IMDEA Networks Institute & University Carlos III of Madrid, Spain. E-mail: banchs@it.uc3m.es
**Patrick Thiran:** EPFL, Switzerland. E-mail: patrick.thiran@epfl.ch

## 1 Introduction

The web activity of users, i.e., which websites they visit, is known to be sensitive data as it discloses a large amount of information. Such information can be used by repressive states against citizens who try to go against country-level censorship and by marketers (e.g., it has been revealed that in the USA, Internet service providers (ISPs) sell the Internet browsing records to marketers [4]). Being able to keep a person's web activity private is thus of the utmost importance. Typical solutions include the use of encryption protocols that hide only the content of the packet, such as TLS (used by HTTPS), and of anonymous communication tools that also hide the destination of the packets from a local adversary (e.g., a curious or state-controlled ISP, or the curious administrator of a public WiFi access-point), such as Tor. Anonymous communication tools like Tor, which we study more particularly, are supposed to make it impossible for a local adversary to detect which website a client visits.

Recent works [18, 30, 32, 48, 49, 66, 68], however, have shown that traffic analysis techniques, such as *website fingerprinting* (denoted in this paper by WF), enable a local adversary to identify with high accuracy which website is visited (more than 90% accuracy in a list of 100 monitored websites), even if the client uses an anonymous communication tool (i.e., she hides the final destination and content of the packets from the local adversary). The adversary can successfully identify the website (carry out a WF attack) by looking at only packet metadata; with Tor, the packets are fragmented in so-called *Tor cells* of fixed size, which means that only the timestamp and direction (incoming or outgoing) of the packets are available to the adversary, and not the size of the packets. WF attacks typically extract features on the observable metadata (such as total number of packets, timings of packets, ratio between outgoing packets and incoming packets). These attacks are formulated as a classification problem and rely on supervised machine-learning techniques to learn associations between features and websites.

A large number of defenses has been proposed to defeat these attacks. They rely on two main techniques: $(i)$ *link padding*, which takes advantage of the inability of the adversary to see the content of the packets and inserts dummy packets to the packet flow to confuse the adversary; and $(ii)$ *packet delaying*, which delays packets to modify the trace. Both of these techniques incur a performance overhead: Link padding causes a traffic overhead, because more packets are sent; and packet delaying causes a loading-time overhead, because delaying packets makes the website loading time larger. This yields a trade-off between privacy and performance.

In this paper, we consider a client that is multihomed, i.e., she is connected to the Internet through multiple networks. Multihoming has been studied for quite some years as a solution for improving reliability and performance. It has long been used by enterprises [7] but, until recently, rarely by individual clients due to the lack of natively multihomed devices, and because it relied on multipath solutions, such as SCTP [33, 64], which are difficult to use in today's Internet. Multihoming has recently gained popularity for individual clients for two reasons. First, multihomed devices have become omnipresent in the last few years, in particular due to the emergence of hybrid networks (networks with two or more technologies). Today, virtually all smartphones are natively multihomed and support both WiFi and cellular. All laptops have a WiFi interface and can easily be made multihomed by the addition of a lightweight and low-cost component (e.g., 3G/4G USB dongles) or with a smartphone sharing its connection through Bluetooth or USB. Dual-SIM cellphones have also been launched onto the market and are gaining popularity. Second, multipath solutions compatible with the protocols dominant in today's Internet have been recently developed to exploit hybrid networks. The most popular solution is multipath TCP (MPTCP) [27, 28]: It has been shown to bring significant performance gains [16, 29, 42, 53] and it is now widely deployed [11]. In particular, all Apple operating systems (for phones and computers) now support MPTCP [1], and an implementation exists for Linux-based devices [46].

As exposed above, multihoming and multipath are becoming increasingly present in today's networks. Here, we study their implications on WF attacks. We consider the scenario of a multihomed client with an adversary that can only observe the traffic sent through one of the networks—which is very likely in several use cases. For instance, one of the most relevant use cases for Tor is web access in countries under censorship, where users might be fingerprinted by their local ISP under government regulations. In this case, multihoming does not only offer better performance and reliability; it can also be a means to circumvent fingerprinting by a local ISP, because it is very complex and highly unlikely that different ISPs combine their traces to fingerprint a user, as we explain later.

The availability of multiple networks makes it possible to choose, for each packet, through which one of these networks it should be sent, which is referred hereafter as *splitting*. Splitting traffic among the networks makes it possible to *remove* packets from one network by sending them through another—whereas current defenses can only add and/or delay packets, which creates performance overhead, as explained above. This enables us to design a defense with no traffic overhead. Inserting dummy packets and/or delaying packets remains of course possible in multihoming and can be combined with splitting to further improve privacy, as we also show.

To provide security and privacy guarantees, exploiting the existence of several non-colluding entities is a standard and successful technique used in other contexts (e.g., secret sharing [58]). But splitting traffic through multihoming is not straightforward and raises several challenges. The main challenge that needs to be addressed is to know how to split the traffic between the networks to improve the resistance against WF. We show that this is non-trivial, as deterministic schemes (e.g., round-robin) do not significantly improve privacy. This makes necessary the development of a specific multipath scheduler for increasing the resistance against WF. Our first contribution is the design of a novel multipath scheduler for protecting against WF. Because one of the main use cases of multihoming relies on hybrid networks, we call this multipath scheduler HyWF. We extensively evaluate HyWF against state-of-the-art WF attacks and we show that it achieves the same level of accuracy as the best existing practical defenses, but does so without adding any traffic overhead. HyWF is compatible with other defenses that rely on link padding or packet delaying, and combining HyWF with another defense further improves privacy by combining the gains brought by the two defenses. Our second contribution is to demonstrate this with the description and evaluation of two novel defenses that combine HyWF with two state-of-the-art defenses: HyWF-AP, an extension of HyWF with adaptive padding [36, 60], and HyWF-WT, an extension of HyWF with Walkie-Talkie [69]. Our third contribution is a proof-of-concept implementation of HyWF. Our implementation does not require modifying Tor or the application. It enables us to evaluate the performance of our splitting scheme. We show that having two paths instead of one has no cost in terms of performance, as HyWF does not add significant loading-time overhead.

This paper is structured as follows. We discuss related work in Section 2, present our system and adversarial model in Section 3, and describe our methodology in Section 4. In Section 5, we introduce HyWF, a defense with no traffic overhead. We evaluate it through extensive simulations in Section 6. In Section 7, we describe the implementation of HyWF and show that it does not add significant loading-time overhead. In Section 8, we introduce and evaluate HyWF-AP and HyWF-WT, two extensions of HyWF with two state-of-the-art WF defenses: adaptive padding and WalkieTalkie, respectively. We conclude in Section 9.

## 2 Related Work

### 2.1 Traffic Analysis

Traffic analysis has been extensively studied in the last 20 years. In 1996, it was shown that valuable information could be extracted from encrypted data [65]; and in 1998, that the website accessed by clients could be successfully identified when observing encrypted data [18]. In 2009, Herrmann et al. showed that WF attacks could be successfully applied to anonymous communication tools [32]. Since then, many WF attacks and defenses have been proposed and several attacks have been shown to be efficient for Tor, as detailed below.

#### 2.1.1 Existing Attacks

**WF Attacks against Tor**: After the first work by Herrmann et al., many other attacks were proposed. They are all formulated as a classification problem: They first extract meaningful features from the observable metadata; then, to associate these features to the websites, they train a supervised machine-learning model. The first attack against Tor that proved to be successful relied on support vector machines (SVM) [49]. Since then, many attacks that employ different features and machine-learning models have been proposed [15, 26, 30, 34, 48, 59, 66, 67, 71]. Recently, attacks employing deep learning have been proposed and proved to be successful [9, 44, 55, 62]. In this paper, we evaluate our defenses against the most relevant attacks, considered to be the most advanced and effective WF attacks [19, 45]. They are very general and not bound to any specific defenses.

- $k$-NN [66]: Wang et al. use a $k$-nearest neighbors model with $N = 1{,}225$ features (total transmission time, number of packets, concentration of outgoing packets, etc.).
- CUMUL [48]: Panchenko et al. propose an attack that uses an SVM model with, as features, the cumulative sum of the packet sizes (negative for outgoing packets, positive for incoming packets). With Tor, the absolute value of the packet sizes is constant.
- $k$-fingerprinting [30]: Hayes et al. propose an attack that extracts $N = 175$ features out of the traces (e.g., total number of packets, number of packets per second, concentration of incoming/outgoing packets, etc.) and uses a random forest classifier.
- Deep Fingerprinting [62]: Sirinam et al. propose to use deep learning on the sequences of incoming and outgoing packets to predict the visited website. It is denoted hereafter by DF.

Juarez et al. [35] and Wang and Goldberg [68] study the practicality of WF attacks. In particular, Wang and Goldberg show that by carefully building the datasets on which the attacks train their model, it is possible to achieve a practical WF attack against Tor.

**Other WF Attacks**: Attacks in contexts other than Tor are also studied. Danezis [21] and Miller et al. [43] study attacks against HTTPS traffic, where the adversary has information about the website that is accessed (the IP address of the server is sent in clear). They show that it can with very good accuracy infer which webpage of the website the client visits. Website fingerprinting is also possible by looking at the effects of contention on the CPU's cache of the client's computer [61]. Recently, a new attack based on similarity digest has been proposed [51]. As opposed to all other existing attacks, it is not formulated as a machine learning problem, and it requires fewer samples for training. This attack is shown to be effective against VPN traffic and to detect malware activity, but not as an attack against Tor traffic.

**End-to-end Traffic Analysis**: Finally, end-to-end traffic analysis attacks, introduced by the seminal work of Feamster and Dingledine [25], are performed when the adversary (typically, an ISP) sees traffic on both ends of the path and can identify the website visited by a client by correlating the traffic sent before and after an anonymity network (e.g., Tor). Similarly to most WF defenses [14, 19, 36, 69], we assume here that the adversary is local, i.e., it is unable to perform end-to-end traffic analysis attacks. With a non-local adversary, the multihoming solution presented in this paper has two contradicting consequences: On the one hand, by sending traffic to different ISPs at the client side, multihoming increases the probability that the same ISP is on the two ends of the path and is able to perform an end-to-end traffic analysis attack. On the other hand, because this ISP would see only part of the packets at the client side and all the packets at the server side, correlation would be looser, which would very likely decrease the accuracy of the attack. Studying the precise effect of multihoming on end-to-end traffic analysis attacks is out of the scope of this paper.

#### 2.1.2 Existing Defenses

In parallel, mechanisms to protect against these WF attacks are studied. The large majority relies on link padding (inserting dummy packets[1] to confuse the adversary that is unable to distinguish them from real packets) and/or on packet delaying, and incurs traffic and/or loading-time overhead.

**1** Note that breaking packets in Tor cells also requires some padding so that all cells have the same size. Because this padding is done with Tor with or without another specific defense against WF attacks, we only consider the insertion of dummy packets to compute the traffic overhead.

The most secure defense consists in ensuring that all traces look exactly the same; this is the basis for constant-rate padding, such as BuFLO [24]. With BuFLO, real packets are delayed and dummy packets are inserted so that the inter-arrival times (time interval between two consecutive packets) stay constant. But this might still leak some information as the total loading time might be different for different websites (short traces will end before long traces). CS-BuFLO [13] and Tamaraw [14] are proposed to improve the performance and to reduce the overhead of BuFLO. With these improved techniques, traces for different websites look the same, and the adversary is virtually unable to distinguish between them. However, sending packets at a constant rate requires both inserting dummy packets and delaying real packets, which causes very high traffic and loading-time overhead. Tamaraw offers the best performance, but the authors still report a loading-time overhead of 320%. For this reason, several other defenses are proposed.

One defense is proposed that does not rely on link padding or on packet delaying: randomized pipelining (RP) [50]. It enables HTTP pipelining (i.e., sending multiple HTTP requests in parallel) and it randomizes the number and order of parallel HTTP requests. But this technique is shown to be ineffective in practice [15, 67]. LLaMA [19] is a defense that improves RP by combining it with link padding and packet delaying, but this incurs traffic and loading-time overhead. Another defense loads a decoy page each time the client wants to access a website [49], but this defense performs worse than other defenses such as adaptive padding (described below), with smaller privacy and larger overhead [30]. WalkieTalkie [69] improves this idea by simulating the loading of sensitive and non-sensitive pages altogether in a half-duplex mode. Other defenses are proposed at the application layer. ALPaCA [19] is a server-side defense that pads the content of a webpage to alter its characteristics (in particular its size) without modifying how it looks to the client. ALPaCA significantly improves privacy (the accuracy of the attack is reduced to about 15%), but it incurs a traffic overhead of about 90% and a loading-time overhead of more than 50%. In addition, as this defense needs to be implemented at the server side, it might be difficult to deploy in practice.

Other defenses are proposed: They do not try to make all traces look exactly the same (by sending packets at constant rate), rather aim at making the traces be statistically similar. Traffic morphing [70] relies on padding and ensures that the packet-size distribution is similar for all traces. This method is not effective with Tor, which already sends packets with a fixed size, and it is shown to be ineffective even when the packet sizes are different [24, 30]. Adaptive padding (denoted in this paper by AP) [60] employs a similar idea, but it works on the inter-arrival times rather than on the packet sizes. It relies on padding and tries to make all traces statistically similar and undistinguishable from each other by ensuring that the distribution of the inter-arrival times is the same for all traces. WTF-PAD [36] is an implementation of AP for WF attacks.

The above-mentioned defenses can be implemented along with HyWF, the defense that we propose and that does not add overhead, as shown in Section 8.

## 2.2 Privacy-Protecting Splitting Schemes

Splitting data between several non-colluding entities has been proven to be a successful technique. Shamir [58] and Blakley [10] first invented secret sharing in 1979, as a way to protect information by sharing it between several participants. Since then, data-splitting schemes have been used in many other contexts that include cloud oblivious storage [63], privacy-protecting cloud computing [37], secure data deduplication [39], vehicular ad-hoc networks [54], and secure sharing of personal health records [40], to name a few. Tor itself relies on non-colluding entities (the guard and exit relays) to maintain privacy. Message splitting between different paths is theoretically studied in the context of mix networks [57]. Finally, traffic splitting in Tor is briefly studied by De la Cadena et al. [23], in particular when the number of networks is greater than two.

## 2.3 Multipath Routing

Multipath routing makes it possible to split the traffic among several paths. It has recently gained popularity as a way to improve performance [31, 52], including in Tor [8, 38, 56]. The most popular solution is multipath TCP (MPTCP) [27, 28] that we use for our implementation. Our solution can also be implemented with other multipath solutions, e.g., multipath QUIC [22]. In hybrid networks with WiFi and cellular networks, MPTCP has been shown to improve throughput [16, 53], reliability [42], and latency [29]. MPTCP is now widely deployed [11]; in particular, all Apple operating systems (for phones and computers) now support MPTCP, and an implementation exists for Linux-based devices [46].

# 3 System and Adversarial Model

## 3.1 System Model

We consider a Tor client who is multihomed, i.e., she has access to multiple networks. In this paper, we consider that the client uses two networks, which is by far the most frequent

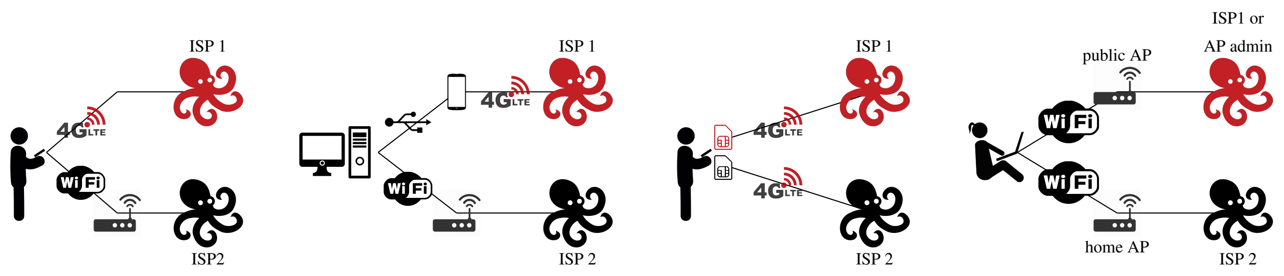

**Fig. 1.** Different real-world examples of a multihomed client. The adversary can be a curious ISP (*ISP1* and *ISP2* in the figure) or the curious administrator of a public WiFi access point (*AP admin* in the figure). Left: client with a smartphone connected to a WiFi access-point (home or public) and to a cellular network. Center left: desktop computer connected to its home access-point with WiFi and to a cellular network through a smartphone sharing its connection with USB. Center right: client with a dual-SIM smartphone connected to two cellular networks with different ISPs. Right: client with a laptop connected to two WiFi access-points (home and public).

case in practice; our defense can easily be extended to more networks which would further improve the protection against WF attacks [23].

A widely available real-world example of a multi-homed client is the case illustrated in Fig. 1 (left) of a client connected to a WiFi access-point and to a cellular network with a smartphone. Other real-world examples of a multihomed client are presented in Fig. 1: A client with a desktop computer or a laptop connected to a home network with WiFi and to a cellular network through a smartphone sharing its connection with USB (center left); a client with a dual-SIM smartphone connected to two cellular networks with different ISPs (center right); and a client with a laptop connected to two access-points, home and public (right). Even though multihoming might come at the price of having to pay for the services of two ISPs, these real-world examples already cover a large amount of users that have multihomed devices (e.g., tablets or smartphones) and have contracts with two ISPs (e.g., cellular and fixed access). Such users already exploit multihoming for enhanced performance and ubiquitous connectivity, and improved privacy is one additional benefit that they can enjoy.

In this paper, we assume that both networks are cost-equivalent, i.e., that the client does not wish to send less traffic on one network. This may be the case in many practical situations (e.g., cellular access with flat fee and home access). When this is not the case, our defense can still be applied with a trade-off between cost and privacy. We briefly discuss this trade-off in Section A.1 in Appendix. We also assume that the delays incurred by the networks are of the same order of magnitude, which is the dominant case (in particular, WiFi and LTE have been shown to have similar latencies, with LTE incurring delays only 10% higher than WiFi on average [29]).

We assume that it is possible to decide on a packet-per-packet basis, in both directions, through which network traffic is sent. The client connects to Tor through a multipath-compatible Tor bridge, as depicted in Fig. 2 (MP bridge). In our evaluation, the client and the Tor bridge use MPTCP, the most popular multipath solution. This approach with a Tor bridge has been widely employed by previous WF defenses [14, 36, 69]. With this architecture, the client defends against all adversaries located before the first Tor node, e.g., the client's local ISP.[2] It does not require modifying Tor nor the application, and MPTCP or any other out-of-the-box multipath solution can be employed between the client and the bridge. The multipath solution can be implemented as a pluggable transport (PT) [2]. In addition to sending traffic on multiple paths, the PT mechanism must also hide that the traffic is using a multipath solution by encrypting the MPTCP packet and encapsulating it in a single-path TCP packet,[3] thus hiding the MPTCP fields (e.g., sequence number).

The above architecture adds some complexity to the system implementation, as multipath needs to be used until the multipath bridge, but such complexity is transparent to the users and the applications. Indeed, multihoming is implemented at the transport layer (e.g., with MPTCP) through a dedicated multipath scheduler (implemented in the client's phone or laptop and in the bridge). It does not disrupt the operation of the higher layers (i.e., the application).

## 3.2 Adversarial Model

We consider a scenario where the client wants to protect against curious adversaries snooping on her traffic in order to

2 The client can also use multiple paths in Tor, similarly as Conflux [8] that uses multipath to improve the performance of Tor. This enables the client to defend in addition against adversaries located within the Tor network, such as curious Tor guard nodes. This would require modifying the Tor software at the client and exit node.

3 The detailed implementation of the encapsulation and encryption scheme is left as an engineering issue that is not addressed by this paper.

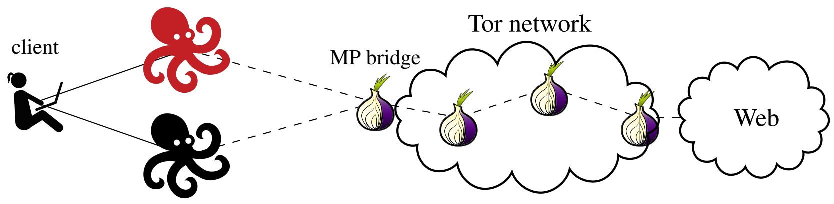

**Fig. 2.** Model where the client is multihomed, with two different adversaries, and connects to a multipath-compatible Tor bridge (MP bridge), typically located in another country.

know which website she accesses. Similarly as with the vast majority of WF defenses [14, 19, 36, 69], we assume that the curious adversaries are local. They can be an ISP, an entity that controls the ISP (e.g., a state) or the curious administrator of a public WiFi access-point. The adversaries are passive, i.e., they do not modify the transmissions. Because the client uses Tor and a PT mechanism as described in Section 3.1, the adversaries are able to observe only the timestamp and direction of the packets. The observable metadata for one website access is called a *trace*.

The key assumption that we make is that the adversaries are not able to correlate the traffic sent through one network and the traffic sent at the same time through the other network, i.e., they cannot reconstruct the original trace. In particular, a single adversary cannot snoop on the two networks at the same time. This happens when the client connects to the Internet by using two technologies and an adversary can snoop on a single technology (e.g., when the adversary is the curious administrator of a WiFi access-point). This also happens when the adversaries are different ISPs, which is an important use case: For example, it has been revealed that in the USA, ISPs sell the Internet browsing records to marketers [4], which is cited as one reason for using the Tor network [3]. Browsing records can typically be inferred by WF attacks. To be protected against such attacks, the client can use two different ISPs, e.g., she uses two technologies (wired or wireless) with two different ISPs, or she connects to two ISPs with the same technology (see the examples of Fig. 1).

We consequently assume that there is one adversary per network. In most cases, the two adversaries do not collude and are consequently unable to reconstruct the original trace. Even if the client uses the same ISP for her connections to the two networks with different medium access technologies (e.g., mobile and WiFi),[4] or if an external entity (e.g., a state or a state-controlled entity) can make the two adversaries (e.g., two local ISPs) collude, for example, in a repressive country under censorship, it would also prove difficult and challenging to reconstruct the original trace from the two different traces, for the following reasons. First, when a device is multihomed, it has two different identifiers for the two ISPs (different IP and MAC addresses), and it can be non-trivial to associate the two identifiers with a same device.[5] Second, the infrastructures for the two different ISPs or for the two medium-access technologies of the same ISP are necessarily different, which means that the paths taken by the packets are distinct: Even if the colluding ISPs are able to associate the two identifiers of a device, the reconstruction of the original trace cannot be achieved synchronously but is necessarily asynchronous. Consequently, the traces for each ISP would need to be stored to be reconstructed later, i.e., the two ISPs would need to store all the per-packet metadata sent through their network for all users they want to attack, until they can reconstruct the complete traces. This would incur significant practical difficulties and storage overhead. Third, the two ISPs should be tightly synchronized in order to reconstruct the trace,[6] which would also be challenging in practice; also, different paths would necessarily mean different latencies and packet ordering, which would make the attack much more difficult to perform. In contrast, when the client uses a single ISP, the attack can be performed on-the-fly along the single path, hence there is no need for storage and synchronization. Hence, in repressive countries where the state is known to spy on the users and can force ISPs to collude, spying on a client who uses two different ISPs would be much more difficult, if at all possible, as specific tools would need to be devised to correlate the different traces.

**4** Note that the client must be connected to two different networks. If she is connected to the same home router, even with two different technologies (e.g., WiFi and Ethernet, WiFi and power-line communication, two WiFi connections in the 2.4 GHz and 5 GHz band, etc.), the ISP will see a single trace and will be able to apply existing efficient attacks.

**5** It is trivial if the client is registered with the ISPs under the same name, but this is not always the case, e.g., if she uses a friend's Internet access.

**6** To prevent the ISPs from using the TCP timestamp field, which would enable them to reconstruct the traces without being synchronized, the TCP timestamp option should be disabled; because the TCP timestamp option is negotiated during the TCP handshake between the hosts [12], it is enough if the PT disables it.

In a direct consequence of our hypothesis, an adversary can only use the partial trace sent through a single network to infer the website. In the following, we consequently consider a single adversary that tries to infer the website that a client visits by observing the traffic sent through a single network.

We also make the following assumptions that, extensively made in the literature [15, 32, 49, 59, 66, 67], all favor the adversary:

**Page Load Parsing**: The adversary is able to detect the beginning and the end of different website accesses. We discuss further this question in Section 7.

**No Background Traffic**: The adversary is able to distinguish one trace for a specific website access from other packets sent by other applications or for the access of another website. This can prove challenging in practice, because multiple applications might be used simultaneously.

**Replicability**: The adversary is able to train its machine-learning model under the same conditions as the client. In practice, the trace that an adversary wants to classify is not obtained with the same method as the traces used for training the machine-learning model (e.g., they do not use the same device, the device is not at the same distance of the WiFi access-point and the cellular base-station, etc.).

Failing to verify one of these three assumptions has been shown to have negative effect on the attack [20, 35]. This means that the reported accuracy of the attacks against the defenses that we propose, HyWF and HyWF-AP, is a conservative value and that in a practical scenario, it is very likely that our defenses would achieve better privacy.

## 4 Data and Methodology

### 4.1 Datasets

**`Wang`**: The primary dataset are traces gathered in 2014 by Wang et al. [66]. This dataset, denoted by `Wang`, has been extensively used in the literature [6, 30, 48, 66, 68]. It contains 90 different traces for each of 100 *monitored* websites (the total number of monitored traces is $n_{\text{mon}} = 9{,}000$), and one trace for each of $n_{\text{unmon}} = 9{,}000$ *unmonitored* websites. The monitored websites come from a list of websites blocked in China, the United Kingdom, and Saudi Arabia. The unmonitored websites are drawn from Alexa's top 10,000 list. There is no intersection between the monitored and unmonitored websites. 6,000 of the monitored and 6,000 of the unmonitored websites are randomly chosen as the training set, i.e., the set of traces used by the adversary to train the machine-learning model; the remaining 3,000 monitored and 3,000 unmonitored are used as our testing set, i.e., the set of traces used to test the accuracy of the attack.

**`Hayes`**: To verify that our conclusions can be generalized to different datasets, we also use the dataset gathered in 2016 by Hayes et al. [30]. This dataset is denoted by `Hayes`. It contains 100 different traces for 85 monitored websites (i.e., $n_{\text{mon}} = 8{,}500$), and one trace for 100,207 unmonitored websites. 55 of the monitored websites are the 55 top Alexa websites and the remaining 30 monitored websites are 30 popular Tor hidden services. The unmonitored websites are drawn from Alexa's top list, excluding the top 55. There is no intersection between the monitored and unmonitored websites. The training set always contains two-thirds of the monitored traces, i.e., 5,667 monitored traces, and the testing set contains the remaining third of the monitored traces. For the unmonitored traces, the Hayes dataset is tried in two scenarios: with $n_{\text{unmon}} = n_{\text{mon}} = 8{,}500$ (because we have $n_{\text{unmon}} = n_{\text{mon}}$ in the `Wang` dataset), and two-thirds of them in the training set and one third in the testing set; and, similarly as what is done by Hayes et al. [30], with all the unmonitored websites ($n_{\text{unmon}} = 100{,}207$), and 5% of them in the training set (i.e., 5,010) and 95% of them in the testing set.

**`DF`**: Finally, to evaluate our defense against the DF attack [62], we use the dataset provided by the authors, denoted by `DF`. It consists of 1,000 traces for 95 monitored websites of the top Alexa's top 100 list and 40,716 unmonitored websites of the Alexa's top 50,000 list (excluding the top 100). The authors of DF have kept only the first 5000 packets of each trace, which means that some traces are truncated. The truncated traces represent however only 8% of the traces, and we do not expect this to impact significantly the results. Two-thirds of the traces are used as the training set, one-sixth as the validation set (used by the DF algorithm) and one-sixth as the testing set.

The raw traces of these datasets are hereafter called *original* traces. When a defense is applied on an original trace (packets are split between the two networks and/or dummy packets are inserted), the resulting trace is called *protected*.

### 4.2 Methodology and Experiments

We assume that the adversary knows the defense (i.e., it knows the splitting scheme used by the client and by the proxy) and is able to train a machine learning model on protected traces. In practice, an adversary does not know whether a specific trace is protected (sent through two networks) or not (sent through a single network). Hence we assume that it knows that the client has the possibility to use two networks, but that it does not know if the client is actually using one or two networks. We will show that this assumption does not weaken the adversary, as the performance of the WF attacks remains extremely close to that obtained if the adversary knows beforehand that the trace is protected (in other words, the adversary is able to train a model that distinguishes protected traces from original traces).

To evaluate the attacks, we use the true positive rate (TPR) as one accuracy measure, defined as the probability that a monitored website is classified as the correct monitored website. The lower the TPR is, the more secure the scheme is. We do both closed-world and open-world experiments.

**Closed-world**: In the closed-world experiments, the adversary tries to predict which website is visited out of the different monitored websites. Only the monitored websites are used for both training and testing.

**Open-world**: In the more practical open-world experiments, the adversary tries to predict whether the client visits a monitored or an unmonitored website, and if it is a monitored website, which one it is. It is important to measure if the adversary makes a prediction error when observing the trace of an unmonitored website: In addition to using the TPR, we use the false positive rate (FPR) as a second accuracy measure, defined as the probability that an unmonitored website is incorrectly classified as a monitored website. The higher the FPR is, the more secure the scheme is.

# 5 Designing HyWF, a Defense without Overhead

We first study defenses with no overhead: No packet is added to the trace (i.e., no link padding) and no packet is delayed by the client. The only choice that the client and the proxy make is about how packets are split between the two networks. In this section, we evaluate our design decisions in the closed-world experiment with the primary dataset (`Wang`) and against the $k$-fingerprinting attack. This attack is chosen because it is found to be the most efficient attack among $k$-NN, CUMUL, and $k$-fingerprinting, three attacks that run on commodity machines. We will evaluate our defense more broadly in Section 6, in particular against the DF attack that is more complex to carry out. For all attacks, we try different hyper-parameter values (e.g., number of trees for $k$-fingerprinting) and report the best results. In this section, we want to compare our defense to AP [36, 60] because it is the defense that, found to offer good trade-off between privacy and performance [30], is currently being considered for addition to the Tor project [5]. In Section 6, we compare HyWF with WalkieTalkie [69], another state-of-the-art defense.

As opposed to AP, the schemes described in this section do not incur any traffic overhead and do not require statistics on the traces. With the `Wang` dataset in the closed-world experiment, AP reduces the TPR of the $k$-fingerprinting attack to around 40% (see Section 8 for details on AP). Our goal is to achieve at least the same accuracy. We first show that off-the-shelf schedulers do not achieve this goal. We then show that a random splitting scheme can achieve a level of privacy equivalent to that of AP.

## 5.1 Off-the-Shelf Schedulers

### 5.1.1 Split Outgoing and Incoming Traffic

The first solution, appealing due to the simplicity of its implementation, is to divide outgoing and incoming traffic, and to send the former through one network, and the latter through the other network. With the `Wang` dataset, this means that 90% of the traffic (the incoming traffic) is sent through one network, and 10% (the outgoing traffic) through the other. Such a scheme reduces to 67% the TPR of the attack against the network through which the incoming traffic is sent, and to 55% against the network through which the outgoing traffic is sent. Even though the privacy improvement is significant, this does not achieve our goal to reach the level of privacy offered by AP. Consequently, we move to schemes where both networks can be used for both incoming and outgoing traffic.

### 5.1.2 Round-Robin Scheduler

The simplest multipath scheduler is a round-robin scheduler (a scheduler that sends packets alternatively through the two networks). This scheduler is, for example, proposed in the default Linux kernel MPTCP implementation [47]. The number of consecutive packets sent through one network can be tuned and is denoted by $n_{\text{cons}} \in \mathbb{N}$. But for all values of $n_{\text{cons}}$, the TPR of the attack is almost not reduced, compared to the baseline, going from 91% to at best 85% (the results are shown in Fig. 13 in Appendix). This is because with the round-robin scheduler, the splitting scheme is deterministic, and the adversary is able to learn the characteristics of the protected traces. To increase the level of privacy to the desired goal, off-the-shelf deterministic schedulers are not sufficient. We now study the use of a random splitting strategy.

## 5.2 Fixed Splitting Probability

As mentioned earlier, we assume that the client wants the same level of privacy in the two networks, i.e., she sends the same average amount of traffic through both networks. In Appendix, we discuss the case where she sends a smaller fraction of her traffic through one of the networks, e.g., because this network is more costly (this naturally degrades privacy in the other network).

We first study the simplest random splitting strategy: For each packet, the client and the proxy randomly send it through Network 1 (with probability $p = 0.5$) or through Network 2 (with probability $1 - p = 0.5$). The accuracy of this very simple strategy depends heavily on the assumption made for the adversary. If the adversary does not know that the client is using a second network (i.e., it learns a model only on original traces), the TPR of the attack is reduced to virtually 0 (around 3%, whereas random guessing gives an accuracy of 1%). But this corresponds to a very weak adversary, hence is too optimistic. If the adversary knows that a second network is used (i.e., it learns a model on protected traces), then the TPR of the attack is again high (around 80%). More importantly, if the adversary learns a model with both protected and original traces (it only knows that the client has the possibility to use two networks, but it does not know if the client is actually using one or two networks), the accuracy is also around 80% for protected traces and around 90% for original traces. This means that the adversary is able to learn if the traces are protected or not. Clearly, this simple random strategy is insufficient.

## 5.3 One Probability per Website Access

Alternatively, the client and the proxy can employ the following more complex defense: At the beginning of a website access, they choose a probability $p$ uniformly at random in $[0, 1]$, and for this website access, they send packets with probability $p$ through Network 1 and with probability $1 - p$ through Network 2. This means that for each access of a website, the splitting probability is different. In particular, two different accesses of a same website will look different. On average along all website accesses, 50% of the traffic is sent through each network.

The probability used by the client and by the proxy are different, because they are chosen independently; they are respectively denoted by $p_c$ and $p_p$. The adversary has no access to $p_c$ or $p_p$, chosen locally for each website access. We assume, however, that it knows the strategy (i.e., that $p$ is chosen uniformly in $[0, 1]$) and it is able to train a model on traces protected with this strategy, i.e., to devise an attack that specifically targets a multipath defense with random splitting probability: With the $n_{\text{or}} = 6{,}000$ traces in the training set of original traces, the adversary can build a larger training set of $n_{\text{pr}}$ protected traces by computing several times the protected trace of any original trace (the protected trace is different each time because $p_c$ and $p_p$ are different each time). Because we assume that the adversary does not know if a trace is protected or not, it must train a model with a training set that consists

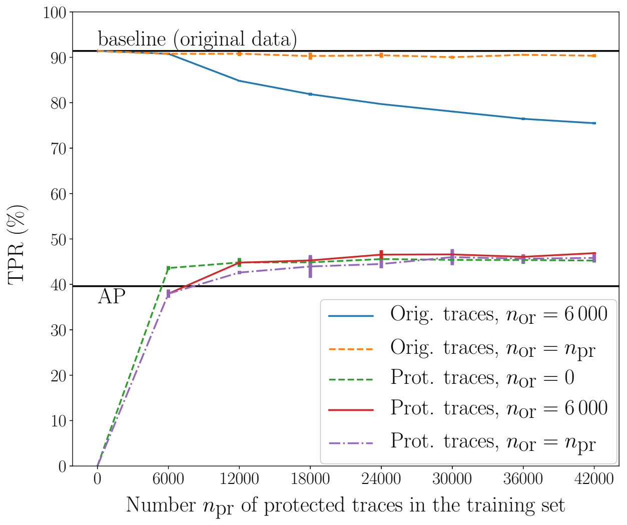

**Fig. 3.** Performance of the attack on original traces and traces protected with the splitting scheme described in Section 5.3, for different sizes of the training set. Closed-world experiment, `Wang` dataset, $n_{\text{cons}} = 1$.

of both protected and original traces. We evaluate the effect of the defense when the adversary tries to attack original traces and when it tries to attack protected traces.

**Effect of the Scheme on Original Traces**: We start by studying the effect of the defense for an attack against original traces. The results are shown in Fig. 3 (*Orig. traces*) for different values of $n_{\text{pr}}$; for every $n_{\text{pr}}$, the attack is repeated five times: The results might be different for each attack because the splitting probability $p$ are different. Consequently, the traces in both the training set and the testing set are different. We present averaged results with a bar indicating the standard deviation. Adding more protected traces in the training set and using only the $n_{\text{or}} = 6{,}000$ original traces tends to decrease the accuracy (plain blue line). However, this comes only from the fact that adding more protected traces biases the training set towards the protected traces, hence reduces the accuracy for the original traces. If the training set contains as many original traces as the number of protected traces ($n_{\text{or}} = n_{\text{pr}}$, dashed orange line), then the accuracy stays very close to the baseline. Note that, as opposed to the protected traces for which protecting the original trace gives a different result each time, adding each original trace several times does not add any information (as the trace is always the same) but only removes the bias towards protected traces. Because we assume that the adversary does not know whether a trace is protected or not and needs to be able to attack both protected and original traces, we use (unless specified otherwise) $n_{\text{or}} = n_{\text{pr}}$.

**Effect of the Scheme on Protected Traces**: We then study the effect of the defense on protected traces. The results are shown

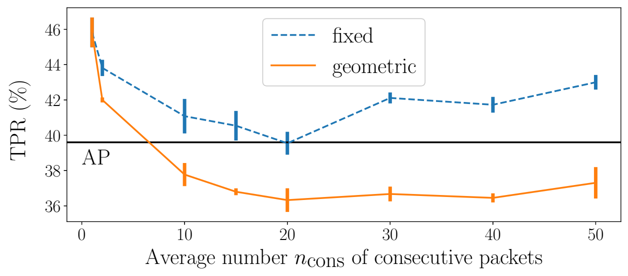

**Fig. 4.** Performance of the attack on traces protected with the splitting scheme described in Section 5.4, for different values of the average number of consecutive packets $n_{\text{cons}}$. Closed-world experiment, `Wang` dataset.

in Fig. 3 (*Prot. traces*) in three scenarios: when only protected traces are used to train the model ($n_{\text{or}} = 0$); when the 6,000 original traces are used once to train the model along with the protected traces ($n_{\text{or}} =$ 6,000); and, to remove the bias towards protected traces, when the number of original traces is the same as the number of protected traces ($n_{\text{or}} = n_{\text{pr}}$). As expected, adding several protected traces for each original trace in the training set helps the adversary: The TPR increases from 37.9±0.8% when they are not repeated to 45.8±0.8% when they are repeated seven times. This is because adding several protected traces per original trace enables the attack to implicitly infer the splitting probability $p$ used for one website access, because it becomes more likely that a protected trace for the same website exists in the training set with a splitting probability close to that used for the website access. But the TPR plateaus once $n_{\text{or}}$ reaches 36,000/42,000. We also note again that the TPR of the attack is similar when the adversary knows beforehand that the trace is protected ($n_{\text{or}} = 0$) and when the adversary does not know it ($n_{\text{or}} = n_{\text{pr}}$): The adversary is able to distinguish protected traces from original traces.

## 5.4 Consecutive Packets to One Network

Fig. 3 also shows that this random splitting scheme does not attain our goal of reaching a TPR of the attack below 40%. We next show that the number of consecutive packets sent through one network has an impact on the TPR of the attack. We try two different settings. In the first setting, a number $n_{\text{cons}} \in \mathbb{N}$ is fixed in advance for all traces; when one network is chosen randomly for sending one packet, the source sends $n_{\text{cons}}$ packets through this network, before choosing again randomly one of the two networks. In the second setting, the average number of consecutive packets $n_{\text{cons}} \in \mathbb{N}$ is fixed in advance for all traces; when one network is chosen randomly, the source draws randomly a number $c \in \mathbb{N}$ from a geometric distribution with average $n_{\text{cons}}$, and sends $c$ packets through this network, before choosing again randomly one of the two networks and drawing a new value $c$. Intuitively, choosing a geometric distribution makes it much more difficult for an adversary to predict when the flow switches path (the geometric distribution is the only memoryless discrete-valued distribution). Note that $n_{\text{cons}}$ is the average number of consecutive packets sent *each time a network is chosen*. The average number of consecutive packets per network for an entire trace is different and depends on $p$: If $p$ is close to 1, then the probability that Network 1 is chosen several times in a row will be high, and the average number of consecutive packets in Network 1 will be larger than in Network 2. The results are shown in Fig. 4. Sending consecutive packets through each network improves the privacy of the defense scheme (it decreases the TPR of the attack). As expected, choosing randomly the number of consecutive packets to send (second scheme, with a geometric distribution) further improves privacy. In Section 6.2, we evaluate further the effect of $n_{\text{cons}}$.

Repeating several times the same trace with different splitting probabilities enables the attacker to *implicitly* estimate the splitting probability $p$. We also verify that our scheme is robust against an attack that would specifically target our defense by trying to *explicitly* estimate $p$. This attack relies on the fact that sending consecutive packets through one network causes longer inter-packet delays on the other network. If long delays happen only during the utilization of the other network, then it is possible to estimate $p$ by counting the average number $n$ of consecutive packets for which the inter-packet delay $d$ is below some threshold $T$: The expectation of $n$ is $n_{\text{cons}}/(1-p)$, hence an estimation of $p$ is $(n - n_{\text{cons}})/n$. We add to the features of $k$-fingerprinting the estimated value of $p$ for different values of $T$. Specifically, $T = \alpha * d_{\text{avg}}$ where $d_{\text{avg}}$ is the average inter-packet delay. In Table 1, we show the TPR of $k$-fingerprinting for different values of $\alpha$ ($\alpha = 1/3$ and $\alpha = 3$ were also tried with similar results, not shown for clarity's sake); when all $p$ for all $\alpha$ are added in the feature set; and when no $p$ is added to the feature set (same attack as for Figure 4). The attack is repeated five times and we show the average TPR. The results show that our defense remains robust against such an attack; the TPR for all values of $\alpha$ is within the standard deviation of the value without the estimated $p$ (36.9% ± 0.8%).

**Table 1.** Performance (TPR) of $k$-fingerprinting where an estimation of the splitting probability $p$ is added to the feature set, for different values of $\alpha$. Closed-world experiment, `Wang` dataset, $n_{\text{pr}} =$ 42,000.

| $\alpha$ | **1/4** | **1/2** | **1** | **2** | **4** | **all** | **none** |
|---|---|---|---|---|---|---|---|
| | **37.2%** | **37.4%** | **36.4%** | **37.2%** | **36.4%** | **36.9%** | **36.9%** |

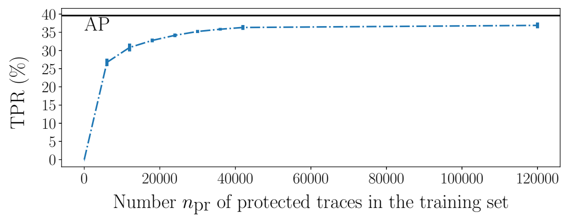

**Fig. 5.** Performance of the $k$-fingerprinting attack against HyWF, for different sizes of the training set. Closed-world experiment, `Wang` dataset.

## 5.5 Definition of HyWF

The scheme defined in Section 5.4 achieves our goal of reaching a level of privacy equivalent to that of AP. It choses a splitting probability uniformly at random for each website access and uses a number of consecutive packets drawn from a geometric distribution with average $n_{\text{cons}} = 20$. The TPR of the attack against this scheme is shown in Figure 5 as a function of $n_{\text{pr}}$. With $n_{\text{pr}} = 120{,}000$, it is decreased to 36.9±0.8%, i.e., it performs even a bit better than AP. We denote this scheme by HyWF. Algorithm 1 presents the pseudo-code for HyWF.

**Algorithm 1** Pseudo-code of HyWF. $\mathrm{G}_0$, Unif and Bern denote, respectively, the geometric, uniform and Bernoulli distributions.

```
$n_{\text{cons}} = 20$
for each website access do
    Draw $p$ from Unif $([0, 1])$
    $n \leftarrow 0, c \leftarrow 0$
    for each packet do
        $n \leftarrow n + 1$
        if $n > c$ then
            Draw $c$ from $\mathrm{G}_0\,(1/n_{\text{cons}})$ and $i$ from Bern $(p)$
            $n \leftarrow 0$
        end if
        Send packet through Network $i$
    end for
end for
```

# 6 Evaluation of HyWF

We now broadly evaluate HyWF with different WF attacks and different datasets, and with open-world experiments. We also compare HyWF with other state-of-the-art defenses.

## 6.1 Different WF Attacks

In the results presented above, we show only the highest accuracy among the attacks $k$-NN [66], CUMUL [48], and $k$-fingerprinting [30]. The best results against protected traces among these three attacks are always obtained with $k$-fingerprinting. In Table 2, we also show the results obtained with the two other attacks, $k$-NN and CUMUL. Each attack is repeated five times and Table 2 also presents the standard deviation. The first line of the table (*Orig.*) corresponds to the case $n_{\text{or}} = 6{,}000$ and $n_{\text{pr}} = 0$ (the model is trained only on the original traces). We also show the results with a model trained on original and protected traces for an attack against original traces (second line) and protected traces (third line); we compute the TPR for different values $n_{\text{or}} = n_{\text{pr}}$ between 6,000 and 42,000 and keep the best result. For HyWF, $k$-fingerprinting yields results significantly better than the other three attacks. Note in particular that with $k$-NN and CUMUL, the performance of the model that works for both protected and original traces (an assumption that is necessary for a practical attack) is decreased even when attacking original traces. Because it uses only the cumulative sum of the packet sizes as features, CUMUL has already been shown to perform less well than $k$-fingerprinting (that uses more diverse features) when attacking protected traces [30]. $k$-NN has already been shown to perform less well than $k$-fingerprinting and better than CUMUL against protected traces [30], which is consistent with our results.

We then evaluate our defense against DF [62], one of the most recent WF attack. It uses deep learning to break WF defenses. This attack is shown to present better performance against state-of-the-art defenses than the other three attacks $k$-fingerprinting, $k$-NN, and CUMUL; in particular, it achieves around 90% accuracy against AP. Using deep learning makes it also much more computationally intensive than the other defenses; in particular, DF requires the use of GPUs and cannot run on commodity machines. We use a 48-core machine with 4 GPUs and 256 GB of RAM. Fig. 6 shows the performance of DF against HyWF, where $n_{\text{pr}}$ is increased by applying HyWF several times, as explained in Section 5.3. We first note that DF performs better than $k$-fingerprinting and reaches

**Table 2.** Performance (TPR) of different attacks with a model learned on original traces (*Orig. traces*), and with a model learned on original traces and traces protected with HyWF. Closed-world experiment, `Wang` dataset.

| | **$k$-fingerprinting [30]** | **$k$-NN [66]** | **CUMUL [48]** |
|---|---|---|---|
| **Orig.** | **90.7±0.3%** | **91.0±0.3%** | **90.2±0.6%** |
| **HyWF, orig.** | **90.8±0.3%** | **82.6±1.9%** | **75.9±1.3%** |
| **HyWF, prot.** | **36.3±0.6%** | **15.3±1.7%** | **12.4±0.4%** |

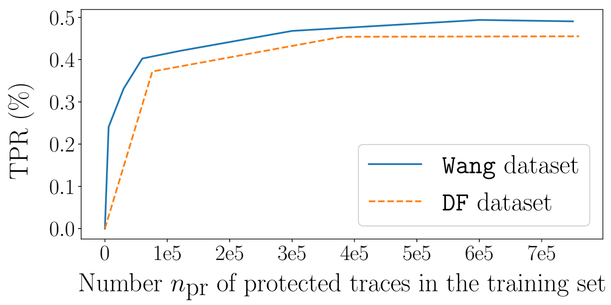

**Fig. 6.** Performance against HyWF of the DF attack. Closed-world experiment, `Wang` and `DF` datasets.

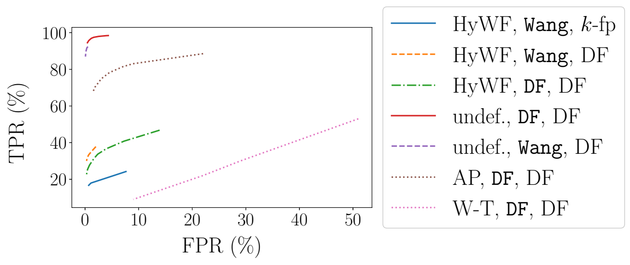

**Fig. 7.** Performance against HyWF, AP and undefended data of the DF attack. Open-world experiment, `Wang` and `DF` datasets.

about 49% of accuracy. HyWF however continues to provide a much better privacy than AP, against which DF achieves 90% accuracy [62]. We also note that DF requires several hundred thousand of protected traces in the training set, i.e., many more than $k$-fingerprinting and the other attacks. This means that DF is much more complex to perform: In addition to requiring a much more powerful machine, it runs in about 14 hours against the `knn` dataset, whereas $k$-fingerprinting requires about an hour.

$k$-fingerprinting was used when we needed to evaluate a wide range of design parameters. In the remainder of the paper, we show results for DF (the most efficient attack), unless specified otherwise.

## 6.2 Different Experiments and Datasets

**Impact of $n_{\text{cons}}$**: In Section 5.4, we evaluated the impact of the average number of consecutive packets $n_{\text{cons}}$ with the `Wang` dataset in the closed-world experiment and with $k$-fingerprinting. Similarly, with other datasets, in both the closed- and open-world experiments, and against DF, the effect of variations of $n_{\text{cons}}$ is small as long as $n_{\text{cons}}$ is in the range 10-40 (see Fig. 16 and Fig. 17 in Appendix). This is an important feature for the generalization of HyWF, as it means that the parameter $n_{\text{cons}}$ does not need to be fine tuned for a specific dataset.

**HyWF in the Open-world Experiment**: We obtained the above results with the `Wang` dataset in the closed-world experiment. We now study open-world experiments (a scenario that is more practical than closed-world experiments) with both the `Wang` and `DF` datasets. The two datasets are described in Section 4.1. The performance achieved by HyWF is reported in Fig. 7, where we show the ROC curve for HyWF and undefended data for `Wang` and `DF` datasets with the DF attack ($k$-fingerprinting against HyWF is also shown for comparison). Protecting the traces with HyWF decreases significantly the TPR and/or increases the FPR. Overall, HyWF significantly improves privacy in the open-world experiment without adding any traffic overhead.

## 6.3 Comparison with other Defenses

We also compare HyWF against two other defenses, AP, already used as a comparison earlier, and WalkieTalkie [69], a defense that incurs both loading-time and traffic overhead, but that has been shown to provide significant privacy improvements [62].

**Closed-world**: For the closed-world setting, we show the traffic overhead (T.ov.), the loading-time overhead (LT.ov.) and the TPR of the DF attack in (Table 3). We see that HyWF achieves a performance similar to or even slightly better than WalkieTalkie in terms of TPR. But WalkieTalkie incurs a much higher overhead in terms of traffic and loading-time. We also study the top-$N$ prediction, in which the attack is successful if the correct website is among the $N$ highest probabilities. Using only the top-1 prediction ignores cases where the adversary confuses pages with only a small set of others [41]. WalkieTalkie has been shown to perform badly against top-2 prediction [62], which we also show in Figure 8 (the real website is virtually always among the top-2 websites predicted by DF). In contrast, HyWF performs much better against top-$N$ prediction. This means the HyWF confuses the adversary significantly more than WalkieTalkie, with which the adversary is able to know with very high probability that the visited website is one of only two websites.

**Open-world**: For the open-world setting, we compare the ROC curves of HyWF, AP and WalkieTalkie (W-T) in Fig. 7. We see that HyWF significantly outperforms AP, and has a

**Table 3.** Performance of HyWF, AP, and WalkieTalkie. Closed-world experiment, DF, `DF` dataset.

| | T.ov. | LT.ov. | TPR |
|---|---|---|---|
| HyWF | 0% | 0% | 48.6% |
| AP | 64% | 0% | 90.7% |
| WalkieTalkie | 31% | 34% | 49.7% |

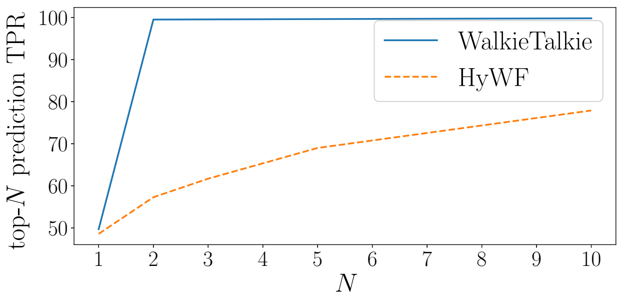

**Fig. 8.** Top-N prediction accuracy of WalkieTalkie and HyWF, DF attack, closed-world experiment, `DF` dataset.

TPR similar to that of WalkieTalkie. WalkieTalkie, however, has a higher FPR, because it simulates loading one sensitive and one non-sensitive page at the same time, which means that the adversary easily confuses sensitive and non-sensitive pages. However, similarly as what we showed for the closed-world experiment, WalkieTalkie is less efficient against top-2 prediction: In the open-world setting, we compute the number of times for which the adversary, when attacking a true sensitive page, tagged the correct sensitive webpage as the most probable one (even when the attack decides for a non-sensitive page for the top-1 prediction). For WalkieTalkie, this number is 67.2%; for HyWF, it is only 36.0%.

Note in addition that, as opposed to AP and WalkieTalkie, HyWF achieves such a performance without adding any traffic and loading-time overhead, or requiring any a priori knowledge (traces statistics for AP, database of sensitive/non-sensitive pages for WalkieTalkie). In addition, as we show in Section 8, HyWF can be combined with any existing defense relying on padding and packet delaying, which can only further improve privacy.

### 6.4 Complexity of the Attacks against HyWF

To increase the accuracy of the attack against HyWF by implicitly inferring the splitting probability, the protected traces are repeated several times in the training set. Adding more traces increases the TPR of the attack, but it also significantly increases the complexity of the learning phase of the model.

***k*-fingerprinting**: In the closed-world experiment, the running time necessary to carry the $k$-fingerprinting attack increases linearly and reaches more than one hour (see Fig. 14 in Appendix). In contrast, when attacking only original traces ($n_{\text{pr}} = 0$, i.e., when the client uses a single network), performing the attack requires only 8 minutes. In the open-world experiment, there are twice more original and protected traces, and the running time of the attack is further increased. It grows linearly from 45 minutes with $n_{\text{pr}} = 0$ to 589 minutes with $n_{\text{pr}} = 60{,}000$ (each trace is repeated five times). When $n_{\text{pr}} > 36{,}000$, a commodity machine with 8 GB of RAM cannot learn the model anymore, because it requires too much memory.

**DF**: As already mentioned, DF is more computationally intensive than $k$-fingerprinting, and this is even more true against HyWF, as the dataset needs to be much larger. Against undefended data, DF reaches an accuracy of more than 90% in about 30 minutes; against HyWF, it requires 14 hours to reach an accuracy of less than 50%.

## 7 Implementation of HyWF

In this section, we describe a proof-of-concept implementation of HyWF and evaluate its performance in terms of privacy and potential loading-time overhead due to the use of multipath—by design, HyWF adds no traffic overhead.

### 7.1 Architecture and Setup

To protect against a local adversary with an easily deployable solution, we use an architecture with a Tor bridge, as the one already presented in Fig. 2. This approach is easy to deploy by installing standard (i.e., unmodified) Tor solutions, along with modified MPTCP modules that implement HyWF scheduling.

Our testbed (illustrated in Fig. 12 in the Appendix) consists of two machines, one acting as the client and one as the bridge, both of them physically connected to the DMZ of one of our institutions to ensure a proper connectivity to the Tor network. Both machines are Intel-based PCs equipped with an i7-6700 (3.4 GHz) CPU and 16 GB of RAM, running Ubuntu 18.04. They are connected using two independent paths, P1 and P2, each path consisting on a different WiFi 802.11g network operating on a different and non-overlapping channel. The WiFi interfaces are Asus USB-n10 nano devices, with the bridge also acting as the WiFi Access Point for both networks. To emulate realistic scenarios where the bridge can be located "far away" from the client, we introduce an extra delay between the communication interfaces of the server and the bridge application, by using the Linux `tc` command. More specifically, following typical round trip time (RTT) figures,[7] we added an extra random RTT, uniformly distributed between 40 ms and 80 ms. The distribution is similar for both paths, i.e., we assume that both networks have similar delays. The bridge

7 See e.g., https://wondernetwork.com/pings.

is connected to the Tor network via an independent Fast Ethernet interface, and both the client and the bridge are provided with an additional control interface to control the execution of the experiments.

To ensure repeatability and ease of scripting together with realistic web browsing, all the resources were requested via a Splash lightweight web browser. The Tor software provides an HTTP proxy in order for Splash to request resources through the Tor network. Both the client and the bridge run MPTCP v.0.9.1 and Tor v.0.3.3. To implement HyWF, we leave unmodified the Tor software and implement a novel MPTCP scheduler that follows the splitting scheme described in Section 5.5 (details on the implementation can be found in Section A.2 in the Appendix). Our proof-of-concept implementation chooses a new splitting probability $p$ for each new MPTCP connection, which requires to keep some per-connection state (as part of our future work we plan to test other mechanisms, e.g., it has been shown that it is possible to efficiently split different website accesses by using a time-based splitting with a fixed threshold of 1 second [68]).

## 7.2 Experimental Evaluation

### 7.2.1 Privacy

We gather traces from 100 different websites and use them to carry out a closed-world evaluation of HyWF. These traces are obtained in different scenarios: with single-path TCP (shown as reference), MPTCP with round-robin scheduler (denoted by RR), MPTCP with default scheduler (denoted by DEF) and HyWF scheduler. This dataset is denoted by `exp`.[8] In Table 4, we show the TPR and the top-2 prediction (as defined in Section 6.3) of the DF attack for the different scenarios. For multipath algorithms, we show the maximum TPR over the two paths.

**Table 4.** Performance of HyWF with `exp` dataset. Closed-world experiment, DF attack, `exp` dataset.

| scenario | TPR | top-2 TPR |
|---|---|---|
| **single-path TCP** | **93.4%** | **97.4%** |
| **MPTCP RR** | **89.6%** | **94.7%** |
| **MPTCP DEF** | **84.2%** | **90.9%** |
| **HyWF, $n_{pr,r} = 10{,}000$** | **44.2%** | **50.8%** |
| **HyWF, $n_{pr,r} = 20{,}000$** | **49.2%** | **54.4%** |
| **HyWF, $n_{pr,r} = 30{,}000$** | **48.7%** | **53.8%** |
| **HyWF, $n_{pr,r} = 30{,}000$, $n_{pr,s} = 1e6$** | **40.5%** | **46.9%** |

8 Dataset available at https://github.com/sebhenri/HyWF_dataset.git.

For HyWF, we present results for different numbers $n_{pr,r}$ of experimentally-obtained traces. The traces for each scenario are obtained in batches of 10,000 traces: when $n_{pr,r} = 30{,}000$, the traces are obtained in three batches. First, note that off-the-shelf MPTCP schedulers do not bring any significant gain in terms of privacy. For RR, this is because packets are split deterministically between the two paths (the accuracy of the attack against the other path is similar). For the default MPTCP scheduler, this is because the scheduler aims at minimizing the RTT and typically sends most of the traffic on a single path, the path with the lowest RTT (the accuracy of the attack against the other path is extremely low, around 2%).

The results also confirm that HyWF, in contrast, offers a significant privacy gain. As opposed to the simulations of Sections 5 and 6, increasing the number of traces from $n_{pr,r} = 20{,}000$ to $n_{pr,r} = 30{,}000$ does not improve the accuracy of the attack. This is confirmed with $k$-fingerprinting, with which the TPR stays constant from $n_{pr,r} = 10{,}000$ to $n_{pr,r} = 30{,}000$, at around 36%. This is because, as opposed to simulated data, network conditions change between batches, which reduces the accuracy of the attack. This data staleness was already observed before [35] and is confirmed in our case when we try to attack data from one batch by training on another batch gathered at two weeks interval: the TPR is only 12.6%. Note that we also observe data staleness for single-path TCP, but it is much less significant (85.3% when attacking one batch with data trained on another batch, vs. 93.4%). To increase the number of instances without having to run new experiments that are prone to data staleness, we evaluate a scenario where $n_{pr,s} = 1e6$ traces are added in the training set by simulating HyWF from the single-path TCP experiments; the testing set always consists only of real multipath traces. This does not help improving the accuracy of the attack, and even degrades it.

Overall, real experiments are close to the simulation results and confirm that HyWF significantly improves privacy. In addition, we have shown in Sections 5 and 6 that to be effective, the attack requires training on a dataset much larger than with single-path. But getting this large dataset is challenging in practice because of data staleness [35].

### 7.2.2 Performance

We now analyze whether such gains in privacy come at some cost in terms of performance. By design, HyWF does not add any traffic overhead, as it only splits packets between two paths. We consider two performance metrics to express loading-time overhead and user experience: the time-to-last-byte (TTLB) and the time-to-first-byte (TTFB). The former gives the loading time of the website, and the latter corre-

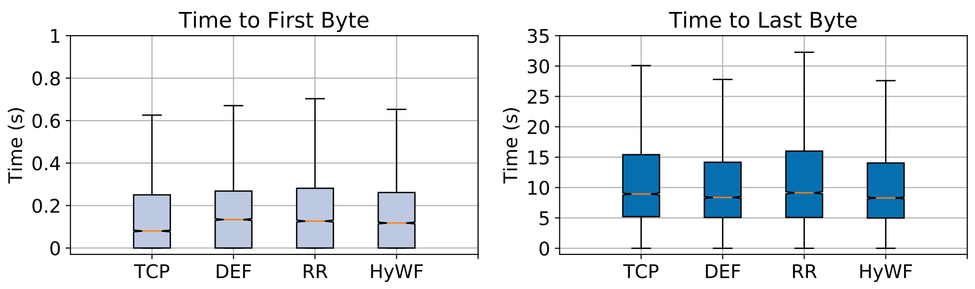

**Fig. 9.** Box-and-whisker plots of the TTFB (left) and TTLB (right) for single-path TCP (TCP) and different configurations of MPTCP: default scheduler (DEF), round-robin (RR) and our proposal (HyWF). `exp` dataset with 10,000 traces per scenario.

sponds to the time elapsed between the first `SYN` of the initial TCP connection and the first incoming TCP segment with user-plane data.

As benchmarks for comparison, we consider the following four transport schemes: $(i)$ single-path TCP, $(ii)$ MPTCP over P1 and P2 with the default MPTCP scheduler (DEF), and $(iii)$ MPTCP over P1 and P2 with a round-robin MPTCP scheduler (RR). We analyze performance for each of these schemes and compare it against our approach (MPTCP over P1 and P2 with our scheduler HyWF).

In Fig. 9 we show the performance results using box-and-whisker plots to represent all values collected for each scheme. We show the TTFB on the left and the TTLB on the right. We conclude from these results that multipath and HyWF have no impact on the loading-time performance: the TTFB and TTLB values are very close for all schemes. Since HyWF does not add any traffic overhead, but simply splits the packets between two paths, we conclude that it is possible to devise an efficient defense against WF attacks, without incurring any performance overhead.

## 8 Combining HyWF with other Defenses

We now show that HyWF is compatible with other defenses against WF. To improve privacy, the client can employ link padding, i.e., insert dummy packets to confuse the adversary. Because it does not see the content of the packets, the adversary is not able to distinguish real packets from dummy packets. She can also delay some packets, which cause loading-time overhead but enables her to shape the traffic to confuse the adversary. Link padding and packet delaying are the methods used by the virtually all existing defenses (see Section 2.1.2). Here, we evaluate HyWF with AP [36, 60], because it is the defense that is under consideration for addition to Tor if link padding were to be implemented [5], and with WalkieTalkie [69], that has been shown to provide significant privacy improvements [62]. Despite their limitations (traffic and/or loading-time overhead, requirement to know the distribution of inter-arrival times or a database of sensitive/non-sensitive webpages), AP and Walkie-Talkie are able to improve privacy when only one network is available, and we want to evaluate the privacy gains offered when they are used along with HyWF, the defense that we describe in Section 5.5.

### 8.1 Designing HyWF-AP and HyWF-WT

#### 8.1.1 HyWF-AP

As mentioned in Section 2.1.2, AP works on ensuring that the distribution of the inter-arrival times is the same for all traces. The packets are never delayed, i.e., there is no loading-time overhead. The goal of AP is to disrupt statistically unlikely delays between packets. In particular, if an unusually large gap between two packets is found, AP adds dummy packets to hide this large gap and to prevent it from being used as a distinguishing feature. Because the authors notice that bursts of packets play an important role in identifying websites, AP mimics bursts of packets when filling the gaps. Details on AP can be found in the related literature [36, 60]. In this paper, we use the implementation of AP for WF, provided by Juarez et al. [36]. With AP, both the client and the proxy use as a parameter some probability distribution for the inter-arrival times. Here, we use the default normal distribution provided by Juarez et al. with their code. Packets are split between the two networks (by following the strategy described in Section 5.5) after the AP is added to the original trace.

#### 8.1.2 HyWF-WT

WalkieTalkie [69] is based on two main mechanisms: half-duplex mode, which means that the client sends requests only when all previous requests have been satisfied; and decoy

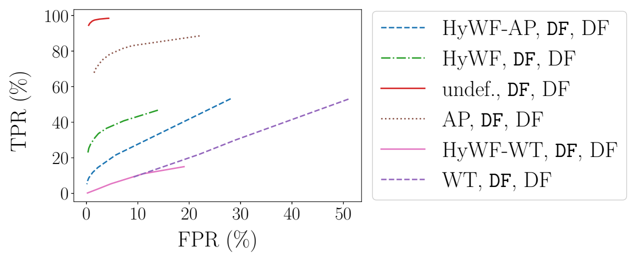

**Fig. 10.** Performance against HyWF, HyWF-AP, HyWF-WT, AP, WalkieTalkie, and undefended data of the DF attack. Open-world experiment, `DF` dataset.

page, in which two pages are loaded at the same time to hide which page is accessed. To reduce overhead, WalkieTalkie only simulates loading two pages at the same time by grouping bursts of packets together. In the closed-world scenario, it loads two pages of the 100 websites under study; in the open-world scenario, it loads one sensitive and one non-sensitive page. The decoy page is chosen to minimize the overhead. This defense consequently requires access to a database of websites. Packets are split between the two networks after WalkieTalkie is added to the original trace.

## 8.2 Evaluation of HyWF-AP and HyWF-WT

In this section, we evaluate HyWF-AP and HyWF-WT, for both closed-world and open-world, with the DF attack and the `DF` dataset. The number of protected traces in the training set is always $n_{\text{pr}} = 800{,}000$.

### 8.2.1 Closed World

In Table 5, we show the TPR and the top-2 TPR of HyWF-AP and HyWF-WT. For reference, we also show the performance of HyWF, AP and WalkieTalkie alone.

For the same overhead, combining HyWF with state-of-the-art defenses significantly improves privacy: The TPR goes from around 50% with HyWF and WalkieTalkie (and more than 90% with AP) to 30% or less. It is interesting to note that HyWF-WT has a lower TPR than HyWF-AP, which is in line with the fact that WalkieTalkie performs better than AP; however, we have seen in Section 6.3 that WalkieTalkie performs badly against top-2 prediction, and we see in Table 5 that for this metric, HyWF-AP performs better than HyWF-WT.

**Table 5.** Performance of HyWF combined with other defenses. Closed-world experiment, DF, `DF` dataset.

| | T.ov. | LT.ov. | TPR | top-2 TPR |
|---|---|---|---|---|
| **HyWF** | 0% | 0% | 48.6% | 57.3% |
| **AP** | 64% | 0% | 90.7% | 94.1% |
| **WalkieTalkie** | 31% | 34% | 49.7% | 99.5% |
| **HyWF-AP** | 64% | 0% | 30.6% | 40.0% |
| **HyWF-WT** | 31% | 34% | 27.6% | 54.7% |

### 8.2.2 Open World

Finally, we evaluate HyWF-AP and HyWF-WT against the DF attack in the open-world setting. The ROC curve for these two novel defenses as well as for HyWF, AP, WalkieTalkie and undefended data is shown in Fig. 10. The results show that combining AP and WalkieTalkie with HyWF significantly increases privacy without incurring a performance cost. This is further confirmed by looking at the top-2 prediction, as defined in Section 6.3 in the open-world case: AP and WalkieTalkie both perform quite badly for this measure (respectively, 88.5% and 67.2%). In contrast, this measure is reduced to 14.4% for HyWF-WT, and to 25.6% for HyWF-AP. This shows that by splitting traffic between two networks, HyWF can be combined with other state-of-the-art defenses, and that doing so further improves privacy.

# 9 Conclusion

We have presented HyWF, a novel defense against website fingerprinting attacks. HyWF exploits multihoming and multipath to split the traffic between two networks. We have shown that a high level of privacy cannot be reached with off-the-shelf multipath schedulers. He have designed an algorithm based on random splitting that achieves a privacy similar to that of state-of-the-art defenses—without any traffic overhead. We have presented a proof-of-concept implementation of HyWF and showed that it does not add any significant loading-time overhead. HyWF is compatible with other defenses that rely on link padding or randomized pipelining. Combining HyWF with another defense further improves privacy. We have illustrated this by introducing and evaluating HyWF-AP and HyWF-WT, two extensions of HyWF with, respectively, adaptive padding and WalkieTalkie defenses. HyWF-AP and HyWF-WT decrease significantly the accuracy of the website-fingerprinting attacks, and they do better than all state-of-the-art defenses.

## Acknowledgements

We would like to thank the anonymous reviewers and in particular our shepherd Nick Hopper for their very helpful comments and feedback. This work was supported in part by the 5G-City Project under Grant TEC2016-76795-C6-3-R through the Spanish Ministry of Economy and Competitiveness.

## References

[1] Improving Network Reliability Using Multipath TCP. https://developer.apple.com/documentation/foundation/urlsessionconfiguration/improving_network_reliability_using_multipath_tcp, accessed Nov. 2019.

[2] The Tor project. Pluggable Transports. https://2019.www.torproject.org/docs/pluggable-transports, accessed Nov. 2019.

[3] Tor: Inception. https://www.torproject.org/about/torusers.html.en, accessed Nov. 2019.

[4] ISPs Sell Clickstreams For $5 A Month, 2007. https://seekingalpha.com/article/29449-compete-ceo-isps-sell-clickstreams-for-5-a-month, accessed Nov. 2019.

[5] Padding Negotiation. *Tor Proposal 254*, 2015. github.com/torproject/torspec/blob/master/proposals/254-padding-negotiation.txt, accessed Nov. 2019.

[6] K. Abe and S. Goto. Fingerprinting Attack on Tor Anonymity using Deep Learning. *Proceedings of the Asia-Pacific Advanced Network*, 2016.

[7] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A Measurement-Based Analysis of Multihoming. In *ACM SIGCOMM Conference*, 2003.

[8] M. AlSabah, K. Bauer, T. Elahi, and I. Goldberg. The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting. *Proceedings on Privacy Enhancing Technologies*, 2013.

[9] S. Bhat, D. Lu, A. Kwon, and S. Devadas. Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning. *Proceedings on Privacy Enhancing Technologies*, 2019.

[10] G. Blakley. Safeguarding Cryptographic Keys. In *National Computer Conference*, 1979.

[11] O. Bonaventure and S. Seo. Multipath TCP Deployments. https://www.ietfjournal.org/multipath-tcp-deployments, accessed Nov. 2019.

[12] B. B. J. V. Borman, D. and E. R. Scheffenegger. TCP Extensions for High Performance. *RFC 7323*, 2014.

[13] X. Cai, R. Nithyanand, and R. Johnson. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *ACM Workshop on Privacy in the Electronic Society*, 2014.

[14] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *ACM Conference on Computer and Communications Security*, 2014.

[15] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security*, 2012.

[16] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley. A Measurement-based Study of Multipath TCP Performance over Wireless Networks. In *ACM Internet Measurement Conference*, 2013.

[17] Y.-C. Chen and D. Towsley. On Bufferbloat and Delay Analysis of Multipath TCP in Wireless Networks. In *IFIP Networking Conference*, 2014.

[18] H. Cheng and R. Avnur. Traffic Analysis of SSL Encrypted Web Browsing, 1998. https://pdfs.semanticscholar.org/1a98/7c4fe65fa347a863dece665955ee7e01791b.pdf, accessed Nov. 2019.

[19] G. Cherubin, J. Hayes, and M. Juarez. Website Fingerprinting Defenses at the Application Layer. *Proceedings on Privacy Enhancing Technologies*, 2017.

[20] W. Cui, T. Chen, C. Fields, J. Chen, A. Sierra, and E. Chan-Tin. Revisiting Assumptions for Website Fingerprinting Attacks. In *ACM Asia Conference on Computer and Communications Security*, 2019.

[21] G. Danezis. Traffic Analysis of the HTTP Protocol over TLS, 2010.

[22] Q. De Coninck and O. Bonaventure. Multipath QUIC: Design and Evaluation. In *ACM International Conference on emerging Networking EXperiments and Technologies*, 2017.

[23] W. De la Cadena, A. Mitseva, J. Pennekamp, J. Hiller, F. Lanze, T. Engel, K. Wehrle, and A. Panchenko. Traffic Splitting to Counter Website Fingerprinting. In *ACM Conference on Computer and Communications Security*, 2019.

[24] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *IEEE Symposium on Security and Privacy*, 2012.

[25] N. Feamster and R. Dingledine. Location Diversity in Anonymity Networks. In *ACM Workshop on Privacy in the Electronic Society*, 2004.

[26] S. Feghhi and D. J. Leith. A Web Traffic Analysis Attack Using Only Timing Information. *IEEE Transactions on Information Forensics and Security*, 2016.

[27] A. Ford, C. Raiciu, M. Handley, S. Barré, and J. Iyengar. Architectural Guidelines for Multipath TCP Development. *RFC 6182*, 2011.

[28] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. *RFC 6824*, 2013.

[29] A. Frommgen, T. Erbshäußer, A. Buchmann, T. Zimmermann, and K. Wehrle. ReMPTCP: Low Latency Multipath TCP. In *IEEE International Conference on Communications*, 2016.

[30] J. Hayes and G. Danezis. $k$-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *USENIX Security Symposium*, 2016.

[31] S. Henri, C. Vlachou, J. Herzen, and P. Thiran. EMPoWER Hybrid Networks: Exploiting Multiple Paths over Wireless and ElectRical Mediums. In *ACM International Conference on emerging Networking EXperiments and Technologies*, 2016.

[32] D. Herrmann, R. Wendolsky, and H. Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Tech-

nologies with the Multinomial Naïve-Bayes Classifier. In *ACM Workshop on Cloud Computing Security*, 2009.

[33] J. R. Iyengar, P. D. Amer, and R. Stewart. Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking*, 2006.

[34] R. Jansen, M. Juarez, R. Galvez, T. Elahi, and C. Diaz. Inside Job: Applying Traffic Analysis to Measure Tor from Within. In *Network and Distributed System Security Symposium*, 2018.

[35] 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*, 2014.

[36] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright. Toward an Efficient Website Fingerprinting Defense. In *European Symposium on Research in Computer Security*, 2016.

[37] T. Jung, X.-Y. Li, Z. Wan, and M. Wan. Privacy Preserving Cloud Data Access with Multi-Authorities. In *IEEE INFOCOM*, 2013.

[38] H. T. Karaoglu, M. B. Akgun, M. H. Gunes, and M. Yuksel. Multi Path Considerations for Anonymized Routing: Challenges and Opportunities. In *Conference on New Technologies, Mobility and Security*, 2012.

[39] J. Li, X. Chen, M. Li, J. Li, P. P. Lee, and W. Lou. Secure Deduplication with Efficient and Reliable Convergent Key Management. *IEEE Transactions on Parallel and Distributed Systems*, 2014.

[40] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou. Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. *IEEE Transactions on Parallel and Distributed Systems*, 2013.

[41] S. Li, H. Guo, and N. Hopper. Measuring Information Leakage in Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security*, 2018.

[42] I. Lopez, M. Aguado, C. Pinedo, and E. Jacob. SCADA Systems in the Railway Domain: Enhancing Reliability Through Redundant Multipath TCP. In *IEEE International Conference on Intelligent Transportation Systems*, 2015.

[43] B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar. I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis. *Proceedings on Privacy Enhancing Technologies*, 2014.

[44] S. E. Oh, S. Sunkam, and N. Hopper. $p$-FP: Extraction, Classification, and Prediction of Website Fingerprints with Deep Learning. *Proceedings on Privacy Enhancing Technologies*, 2019.

[45] R. Overdorf, M. Juarez, G. Acar, R. Greenstadt, and C. Diaz. How Unique is Your .onion?: An Analysis of the Fingerprintability of Tor Onion Services. In *ACM Conference on Computer and Communications Security*, 2017.

[46] C. Paasch and S. Barré. Multipath TCP in the Linux Kernel. https://www.multipath-tcp.org, accessed Nov. 2019.

[47] C. Paasch and S. Barré. Multipath TCP in the Linux Kernel – Configure MPTCP. https://multipath-tcp.org/pmwiki.php/Users/ConfigureMPTCP, accessed Nov. 2019.

[48] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle. Website Fingerprinting at Internet Scale. In *Network and Distributed System Security Symposium*, 2016.

[49] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *ACM Workshop on Privacy in the Electronic Society*, 2011.

[50] M. Perry. Experimental Defense for Website Traffic Fingerprinting. *Tor project Blog. https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting*, 2011.

[51] A. Qasem, S. Zhioua, and K. Makhlouf. Finding a Needle in a Haystack: The Traffic Analysis Version. *Proceedings on Privacy Enhancing Technologies*, 2019.

[52] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *ACM SIGCOMM Conference*, 2011.

[53] C. Raiciu, C. Paasch, S. Barré, A. Ford, M. Honda, F. Duchêne, O. Bonaventure, and M. Handley. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[54] M. Raya and J.-P. Hubaux. Securing Vehicular Ad-Hoc Networks. *Journal of Computer Security*, 2007.

[55] 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*, 2018.

[56] F. Rochet, O. Pereira, and O. Bonaventure. Moving Tor Circuits Towards Multiple-Path: Anonymity and Performance Considerations. Technical report, UC Louvain, 2015. https://pdfs.semanticscholar.org/aa94/7dd4762bd0f6531bacfeac9d29ef1e1d4cd6.pdf, accessed Nov. 2019.

[57] A. Serjantov and S. J. Murdoch. Message Splitting Against the Partial Adversary. In *International Workshop on Privacy Enhancing Technologies*, 2005.

[58] A. Shamir. How to Share a Secret. *Communications of the ACM*, 1979.

[59] Y. Shi and K. Matsuura. Fingerprinting Attack on the Tor Anonymity System. In *International Conference on Information and Communications Security*, 2009.

[60] V. Shmatikov and M.-H. Wang. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In *European Symposium on Research in Computer Security*, 2006.

[61] A. Shusterman, L. Kang, Y. Haskal, Y. Meltser, P. Mittal, Y. Oren, and Y. Yarom. Robust Website Fingerprinting through the Cache Occupancy Channel. In *USENIX Security Symposium*, 2019.

[62] 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*, 2018.

[63] E. Stefanov and E. Shi. Multi-Cloud Oblivious Storage. In *ACM Conference on Computer and Communications Security*, 2013.

[64] R. Stewart. Stream Control Transmission Protocol. *RFC 6824*, 4960.

[65] D. Wagner and B. Schneier. Analysis of the SSL 3.0 Protocol. In *USENIX Workshop on Electronic Commerce*, 1996.

[66] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective Attacks and Provable Defenses for Website

Fingerprinting. In *USENIX Security Symposium*, 2014.
[67] T. Wang and I. Goldberg. Improved Website Fingerprinting on Tor. In *ACM Workshop on Privacy in the Electronic Society*, 2013.
[68] T. Wang and I. Goldberg. On Realistically Attacking Tor with Website Fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016.
[69] T. Wang and I. Goldberg. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *USENIX Security Symposium*, 2017.
[70] C. Wright, S. Coull, and F. Monrose. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *Network and Distributed System Security Symposium*, 2009.
[71] J. Yan and J. Kaur. Feature Selection for Website Fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2018.
[72] K. Yedugundla, S. Ferlin, T. Dreibholz, Ö. Alay, N. Kuhn, P. Hurtig, and A. Brunstrom. Is Multi-path Transport Suitable for Latency Sensitive Traffic? *Computer Networks*, 2016.

## A Appendix

### A.1 Asymmetric Networks

We evaluate the scenario where the client wants to send a smaller fraction $p_s < 0.5$ of her traffic through one of the networks (denoted by Network 1), for example because this network is more costly (e.g., WiFi is typically cheaper to use than cellular). We evaluate the same splitting scheme as HyWF, except that $p$ is chosen uniformly at random in $[0, 2p_s]$, so that the average value of $p$ along all traces is $p_s$. Fig. 11 shows the TPR of the attack against Network 1 and Network 2 as a function of $p_s$. We observe that the relationship between cost (i.e., amount of data sent on the costly network) and privacy is linear, which makes it simple for a client to decide how to make the trade-off between the two.

### A.2 HyWF Scheduler

Our HyWF scheduler builds on the round-robin scheduler provided with the MPTCP implementation and configured with the `full-mesh` mode of operation. With this mode, MPTCP generates one subflow for each IP address pair (source, destination). Therefore, we set up some `iptables` rules to ensure that only the two subflows corresponding to the two links of Fig. 12 are available (this is queried with the `mptcp_rr_is_available` method). Instead of operating in "bursts" of segments, i.e., picking one of the paths at random with probabilities $p$ for Link 1 and $(1-p)$ for Link 2, then drawing the consecutive numbers of segments from a geometric distribution with average $n_{\text{cons}}$, we operate for simplicity on a segment-by-segment basis and use the following algorithm, equivalent to that described in Algorithm 1. When $c$ is drawn from a geometric distribution, this algorithm forms a two-state Markov chain, and it is equivalent to do the following for each packet: When the last packet was sent through Network 1, the packet is sent through Network 2 with probability $(1-p)/n_{\text{cons}}$ and through Network 1 with probability $1-(1-p)/n_{\text{cons}}$; when the last packet was sent through Network 2, the packet is sent through Network 1 with probability $p/n_{\text{cons}}$ and through Network 2 with probability $1-p/n_{\text{cons}}$. We code this algorithm inside the `mptcp_write_xmit` method. More specifically, the `next_segment` method returns a pointer to the next link to use, which is determined based on the last link used (stored in a `*sock` pointer inside an `mptcp_cb` structure) as follows:

- If the last segment was transmitted over Link 1, the next segment is transmitted over the same link with probability $1-(1-p)/n_{\text{cons}}$, and over Link 2 with probability $(1-p)/n_{\text{cons}}$
- If the last segment was transmitted over Link 2, the next segment is transmitted over the same link with probability $1-p/n_{\text{cons}}$, and over Link 1 with probability $p/n_{\text{cons}}$.

The parameters $n_{\text{cons}}$ and $p$ are also stored in the `mptcp_cb` structure, with $p$ randomly chosen on a per-download basis. Due to the kernel programming constraints (in particular, the lack of `float` variables), $p$ is chosen as a random integer in the range $[0, 255]$ using the `get_random_bytes()` function with an `unsigned`

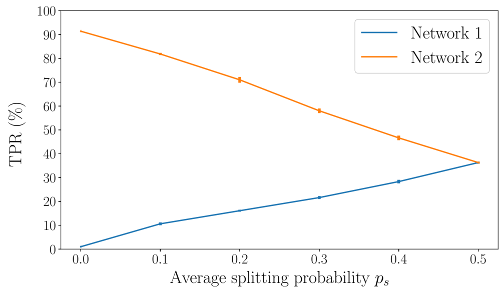

**Fig. 11.** Performance of the attack when the client sends a fraction $p_s$ of its traffic through Network 1, for different values of $p_s$. Closed-world experiment, `Wang` dataset.

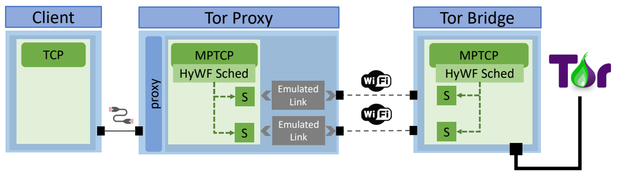

**Fig. 12.** Testbed deployed, with two wireless networks connecting the client (Tor proxy) and the Tor bridge.

char. The algorithm is then updated accordingly. Finally, we follow recommendations to improve latency with MPTCP [17, 72] and we disable the idle restart functionality and the Nagle algorithm (net.ipv4.tcp_low_latency= 1 and net.ipv4.tcp_slow_start_after_idle= 0).

## A.3 Additional Tables and Figures

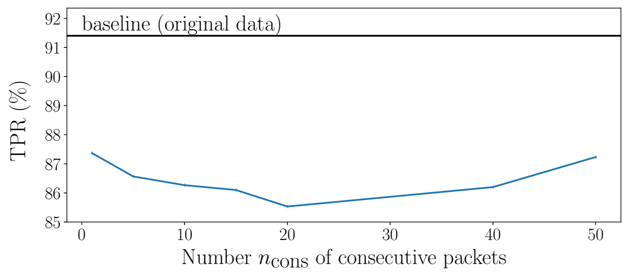

**Fig. 13.** Performance of the attack on traces protected with a round-robin scheme, for different values of the number of consecutive packets $n_{\text{cons}}$. Closed-world experiment, Wang dataset.

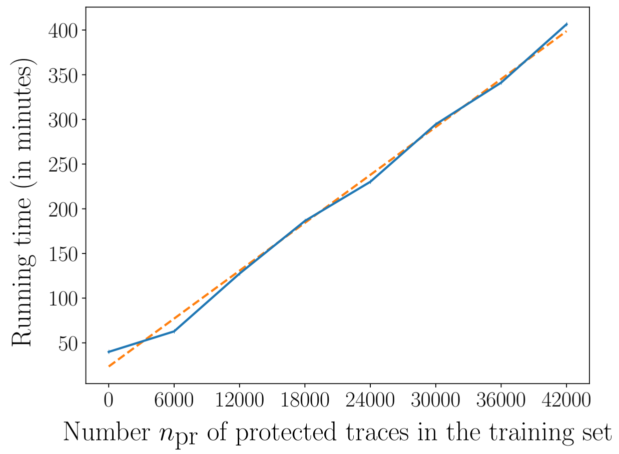

**Fig. 14.** Running time to perform the attack, for different sizes of the training set. Experimental values are in blue; the orange dashed line is the linear fit. Closed-world experiment, $k$-fingerprinting attack against HyWF with Wang dataset.

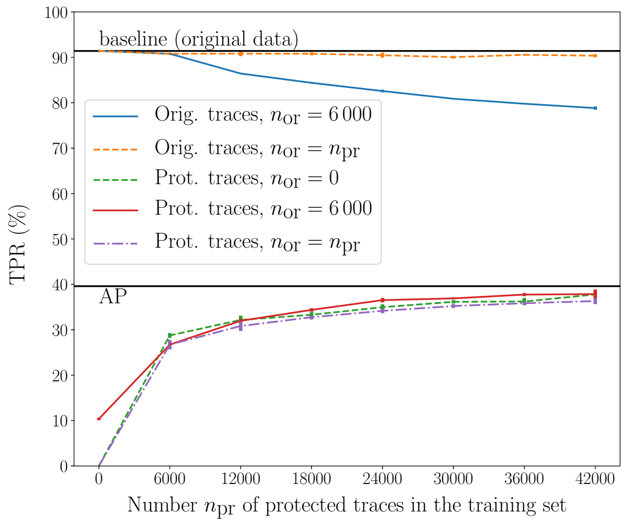

**Fig. 15.** Performance of the attack on original traces and traces protected with HyWF, for different sizes of the training set. Closed-world experiment, Wang dataset.

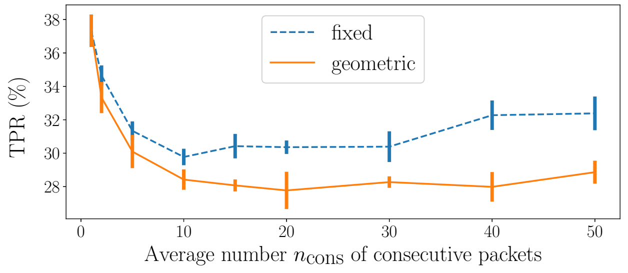

**Fig. 16.** Performance of the attack on traces protected with the splitting scheme described in Section 5.4, for different values of the average number of consecutive packets $n_{\text{cons}}$. Open-world experiment, Hayes dataset.

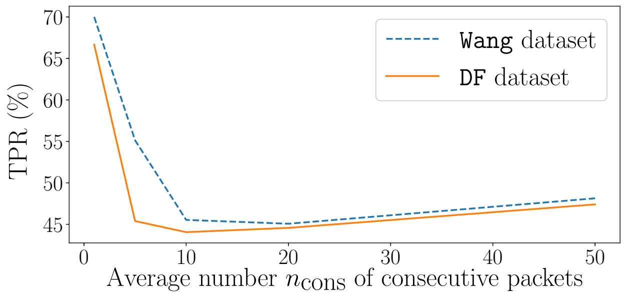

**Fig. 17.** Performance of the attack on traces protected with the splitting scheme described in Section 5.4, for different values of the average number of consecutive packets $n_{\text{cons}}$ and a geometric distribution. Closed-world experiment, Wang and DF datasets, DF attack ($n_{\text{pr}} = 300{,}000$).