# Circuit Fingerprinting Attacks:
# Passive Deanonymization of Tor Hidden Services

Albert Kwon[†], Mashael AlSabah[‡§†*], David Lazar[†], Marc Dacier[‡], and Srinivas Devadas[†]

[†]*Massachusetts Institute of Technology,* {kwonal, lazard, devadas}@mit.edu
[‡]*Qatar Computing Research Institute,* mdacier@qf.org.qa
[§]*Qatar University,* malsabah@qu.edu.qa

This paper sheds light on crucial weaknesses in the design of hidden services that allow us to break the anonymity of hidden service clients and operators passively. In particular, we show that the *circuits*, paths established through the Tor network, used to communicate with hidden services exhibit a very different behavior compared to a general circuit. We propose two attacks, under two slightly different threat models, that could identify a hidden service client or operator using these weaknesses. We found that we can identify the users' involvement with hidden services with more than 98% true positive rate and less than 0.1% false positive rate with the first attack, and 99% true positive rate and 0.07% false positive rate with the second. We then revisit the threat model of previous website fingerprinting attacks, and show that previous results are directly applicable, with greater efficiency, in the realm of hidden services. Indeed, we show that we can correctly determine which of the 50 monitored pages the client is visiting with 88% true positive rate and false positive rate as low as 2.9%, and correctly deanonymize 50 monitored hidden service servers with true positive rate of 88% and false positive rate of 7.8% in an open world setting.

## 1 Introduction

In today's online world where gathering users' personal data has become a business trend, Tor [14] has emerged as an important privacy-enhancing technology allowing Internet users to maintain their anonymity online. Today, Tor is considered to be the most popular anonymous communication network, serving millions of clients using approximately 6000 volunteer-operated relays, which are run from all around the world [3].

In addition to sender anonymity, Tor's hidden services allow for receiver anonymity. This provides people with a free haven to host and serve content without the fear of being targeted, arrested or forced to shut down [11]. As a result, many sensitive services are only accessible through Tor. Prominent examples include human rights and whistleblowing organizations such as Wikileaks and Globalleaks, tools for anonymous messaging such as TorChat and Bitmessage, and black markets like Silkroad and Black Market Reloaded. Even many non-hidden services, like Facebook and DuckDuckGo, recently have started providing hidden versions of their websites to provide stronger anonymity guarantees.

That said, over the past few years, hidden services have witnessed various active attacks in the wild [12, 28], resulting in several takedowns [28]. To examine the security of the design of hidden services, a handful of attacks have been proposed against them. While they have shown their effectiveness, they all assume an active attacker model. The attacker sends crafted signals [6] to speed up discovery of *entry guards*, which are first-hop routers on circuits, or use congestion attacks to bias entry guard selection towards colluding entry guards [22]. Furthermore, all previous attacks require a malicious client to continuously attempt to connect to the hidden service.

In this paper, we present the first practical *passive* attack against hidden services and their users called *circuit fingerprinting* attack. Using our attack, an attacker can identify the presence of (client or server) hidden service activity in the network with high accuracy. This detection reduces the anonymity set of a user from millions of Tor users to just the users of hidden services. Once the activity is detected, we show that the attacker can perform website fingerprinting (WF) attacks to deanonymize the hidden service clients and servers. While the threat of WF attacks has been recently criticized by Juarez *et al.* [24], we revisit their findings and demonstrate that the world of hidden services is the ideal setting to wage WF attacks. Finally, since the attack is passive, it is undetectable until the nodes have been deanonymized, and can target thousands of hosts retroactively just by having access to clients' old network traffic.

[*] Joint first author.

**Approach.** We start by studying the behavior of Tor circuits on the live Tor network (for our own Tor clients and hidden services) when a client connects to a Tor hidden service. Our key insight is that during the circuit construction and communication phase between a client and a hidden service, Tor exhibits fingerprintable traffic patterns that allow an adversary to efficiently and accurately identify, and correlate circuits involved in the communication with hidden services. Therefore, instead of monitoring every circuit, which may be costly, the first step in the attacker's strategy is to identify suspicious circuits with high confidence to reduce the problem space to just hidden services. Next, the attacker applies the WF attack [10, 36, 35] to identify the clients' hidden service activity or deanonymize the hidden service server.

**Contributions.** This paper offers the following contributions:

1. We present key observations regarding the communication and interaction pattern in the hidden services design in Tor.

2. We identify distinguishing features that allow a passive adversary to easily detect the presence of hidden service clients or servers in the local network. We evaluate our detection approach and show that we can classify hidden service circuits (from the client- and the hidden service-side) with more than 98% accuracy.

3. For a stronger attacker who sees a majority of the clients' Tor circuits, we propose a novel circuit correlation attack that is able to quickly and efficiently detect the presence of hidden service activity using a sequence of only the first 20 cells with accuracy of 99%.

4. Based on our observations and results, we argue that the WF attacker model is *significantly more realistic and less costly* in the domain of hidden services as opposed to the general web. We evaluate WF attacks on the identified circuits (from client and hidden service side), and we are able to classify hidden services in both open and closed world settings.

5. We propose defenses that aim to reduce the detection rate of the presence of hidden service communication in the network.

**Roadmap.** We first provide the reader with a background on Tor, its hidden service design, and WF attacks in Section 2. We next present, in Section 3, our observations regarding different characteristics of hidden services. In Section 4, we discuss our model and assumptions, and in Sections 5 and 6, we present our attacks and evaluation. In Section 7, we demonstrate the effectiveness of WF attacks on hidden services. We then discuss possible future countermeasures in Section 8. Finally, we overview related works in Section 9, and conclude in Section 10.

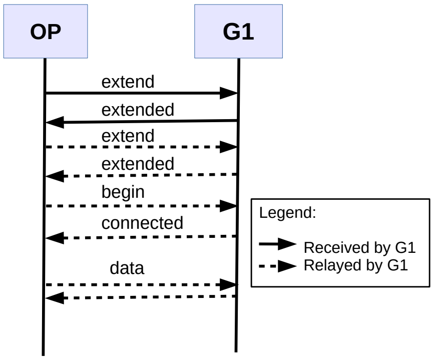


**Figure 1:** Cells exchanged between the client and the entry guard to build a general circuit for non-hidden streams after the circuit to G1 has been created.

## 2 Background

We will now provide the necessary background on Tor and its hidden services. Next, we provide an overview of WF attacks.

### 2.1 Tor and Hidden Services

Alice uses the Tor network simply by installing the Tor browser bundle, which includes a modified Firefox browser and the *Onion Proxy* (OP). The OP acts as an interface between Alice's applications and the Tor network. The OP learns about Tor's relays, *Onion Routers* (ORs), by downloading the network *consensus* document from *directory servers*. Before Alice can send her traffic through the network, the OP builds *circuits* interactively and incrementally using 3 ORs: an entry guard, middle, and exit node. Tor uses 512-byte fixed-sized *cells* as its communication data unit for exchanging control information between ORs and for relaying users' data.

The details of the circuit construction process in Tor proceeds as follows. The OP sends a `create_fast` cell to establish the circuit with the entry guard, which responds with a `created_fast`. Next, the OP sends an `extend` command cell to the entry guard, which causes it to send a `create` cell to the middle OR to establish the circuit on behalf of the user. Finally, the OP sends another `extend` to the middle OR to cause it to create the circuit at exit. Once done, the OP will receive an `extended` message from the middle OR, relayed by the entry guard. By the end of this operation, the OP will have shared keys used for layered encryption, with every hop on the circuit.[1] The exit node peels the last layer of the encryption and establishes the TCP connection to Alice's destination. Figure 1 shows the cells exchanged between OP and the entry guard for regular Tor connections, after the exchange of the `create_fast` and `created_fast` messages.

Tor uses TCP secured with TLS to maintain the OP-to-OR and the OR-to-OR connections, and multiplexes circuits within a single TCP connection. An OR-to-OR connection multiplexes circuits from various users, whereas an OP-to-OR connection multiplexes circuits from the same user. An observer watching the OP-to-OR TCP connection should not be able to tell apart which TCP segment belongs to which circuit (unless only one circuit is active). However, an entry guard is able to differentiate the traffic of different circuits (though the contents of the cells are encrypted).

Tor also allows receiver anonymity through hidden services. Bob can run a server behind his OP to serve content without revealing his identity or location. The overview of creation and usage of hidden services is depicted in Figure 2. In order to be reachable by clients, Bob's OP will generate a hidden service *descriptor*, and execute the following steps. First, Bob's OP chooses a random OR to serve as his *Introduction Point* (IP), and creates a circuit to it as described above. Bob then sends an `establish_intro` message that contains Bob's public key (the client can select more than one IP). If the OR accepts, it sends back an `intro_established` to Bob's OP. Bob now creates a signed descriptor (containing a timestamp, information about the IP, and its public key), and computes a descriptor-id based on the public key hash and validity duration. The descriptor is then published to the hash ring formed by the hidden service directories, which are the ORs that have been flagged by the network as "HSDir". Finally, Bob advertises his hidden service URL `z.onion` out of band, which is derived from the public key. This sequence of exchanged cells to create a hidden service is shown in Figure 3.

In Figure 4, we show how Alice can connect to Bob. Using the descriptor from the hidden service directories, The exchange of cells goes as follows. First, Alice's OP selects a random OR to serve as a *Rendezvous Point* (RP) for its connection to Bob's service, and sends an `establish_rendezvous` cell (through a Tor circuit). If the OR accepts, it responds with a `rendezvous_established` cell. In the meantime, Alice's OP builds another circuit to one of Bob's IPs, and sends an `introduce1` cell along with the address of RP and a cookie (one-time secret) encrypted under Bob's public key. The IP then relays that information to Bob and an `introduce2` cell, and sends an `introduce_ack` towards Alice. At this point, Bob's OP builds a circuit towards Alice's RP and sends it a `rendezvous1`, which causes the RP to send a `rendezvous2` towards Alice. By the end of this operation, Alice and Bob will have shared keys established through the cookie, and can exchange data through the 6 hops between them.

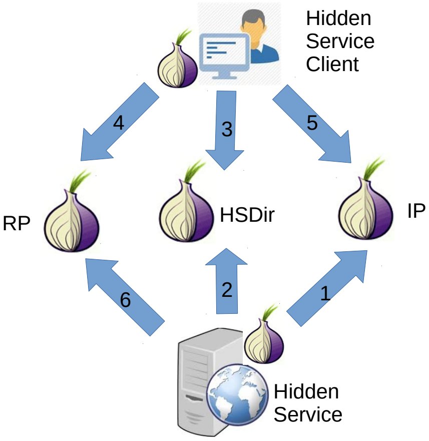


**Figure 2:** Circuit construction for Hidden Services.

## 2.2 Website Fingerprinting

One class of traffic analysis attacks that has gained research popularity over the past few years is the website fingerprinting (WF) attack [10, 36, 35, 9]. This attack demonstrates that a *local passive* adversary observing the (SSH, IPsec, or Tor) encrypted traffic is able, under certain conditions, to identify the website being visited by the user.

In the context of Tor, the strategy of the attacker is as follows. The attacker tries to emulate the network conditions of the monitored clients by deploying his own client who visits websites that he is interested in classifying through the live network. During this process, the attacker collects the network traces of the clients. Then, he trains a supervised classifier with many identifying features of a network traffic of a website, such as the sequences of packets, size of the packets, and inter-packet timings. Using the model built from the samples, the attacker then attempts to classify the network traces of users on the live network.

WF attacks come in two settings: open- or closed-world. In the closed-world setting, the attacker assumes that the websites visited are among a list of *k* known websites, and the goal of the attacker is to identify which one. The open-world setting is more realistic in that it assumes that the client will visit a larger set of websites

[1] We have omitted the details of the Diffie-Hellman handshakes (and the Tor Authentication Protocol (TAP) in general), as our goal is to demonstrate the data flow only during the circuit construction process.

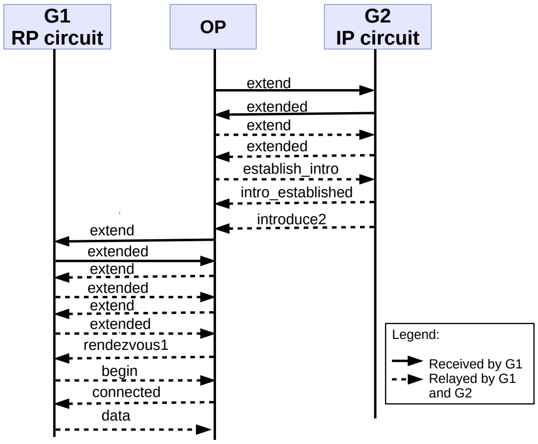


**Figure 3:** Cells exchanged in the circuit between the entry guards and the hidden service operator after the circuits to G1 and G2 have been created. Note that both G1 and G2 might be the same OR, and that entry guards can only view the first `extend` cell they receive.

*n*, and the goal of the attacker is to identify if the client is visiting a monitored website from a list of *k* websites, where $k \ll n$.

Hermann *et al.* [20] were the first to test this attack against Tor using a multinomial Naive Bayes classifier, which only achieved 3% success rate since it relied on packet sizes which are fixed in Tor. Panchenko *et al.* [33] improved the results by using a Support Vector Machine (SVM) classifier, using features that are mainly based on the volume, time, and direction of the traffic, and achieved more than 50% accuracy in a *closed-world* experiment of 775 URLs. Several subsequent papers have worked on WF in open-world settings, improved on the classification accuracy, and proposed defenses [10, 36, 35, 9].

## 3 Observations on Hidden Service Circuits

To better understand different circuit behaviors, we carried out a series of experiments, which were designed to show different properties of the circuits used in the communication between a client and a Hidden Service (HS), such as the Duration of Activity (DoA), incoming and outgoing cells, presence of multiplexing, and other potentially distinguishing features. DoA is the period of time during which a circuit sends or receives cells. The expected lifetime of a circuit is around 10 minutes, but circuits may be alive for more or less time depending on their activities.

For the remainder of this paper, we use the following terminology to denote circuits:

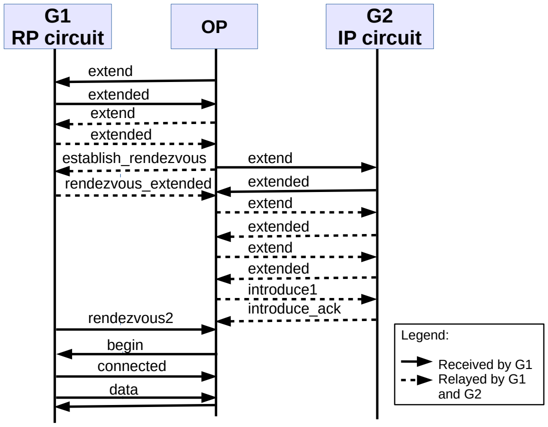


**Figure 4:** Cells exchanged in the circuit between the entry guards and the client attempting to access a hidden service after the circuits to G1 and G2 have been created.

- HS-IP: This is the circuit established between the Hidden Service (HS) and its Introduction Point (IP). The purpose of this circuit is to listen for incoming client connections. This circuit corresponds to arrow 1 in Figure 2.
- Client-RP: This is the circuit that a client builds to a randomly chosen Rendezvous Point (RP) to eventually receive a connection from the HS after he has expressed interest in establishing a communication through the creation of a Client-IP circuit. This circuit corresponds to arrow 4 in Figure 2.
- Client-IP: This is the circuit that a client interested in connecting to a HS builds to one of the IPs of the HS to inform the service of its interest in waiting for a connection on its RP circuit. This circuit corresponds to arrow 5 in Figure 2.
- HS-RP: This is the circuit that the HS builds to the RP OR chosen by the client to establish the communication with the interested client. Both this circuit and the Client-RP connect the HS and the client together over Tor. This circuit corresponds to arrow 6 in Figure 2.

For our hidden service experiments, we used more than 1000 hidden services that are compiled in `ahmia.fi` [2], an open source search engine for Tor hidden service websites. We base our observations on the logs we obtained after running all experiments for a three month period from January to March, 2015. This is important in order to realistically model steady-state Tor processes, since Tor's circuit building decisions are influenced by the circuit build time distributions. Furthermore, we configured our Tor clients so that they do not use fixed entry guards (by setting UseEntryGuards to 0). By doing so, we increase variety in our data collection, and do not limit ourselves to observations that are only obtained by using a handful of entry guards.

### 3.1 Multiplexing Experiment

To understand how stream multiplexing works for Client-RP and Client-IP circuits, we deployed a single Tor process on a local machine which is used by two applications: `firefox` and `wget`. Both automate hidden services browsing by picking a random `.onion` domain from our list of hidden services described above. While the `firefox` application paused between fetches to model user think times [19], the `wget` application accessed pages sequentially without pausing to model a more aggressive use. Note that the distribution of user think times we used has a median of 13 seconds, and a long tail that ranges between 152 to 3656 seconds for 10% of user think times. Since both applications are using the same Tor process, our intention is to understand how Tor multiplexes streams trying to access different `.onion` domains. We logged for every `.onion` incoming stream, the circuit on which it is attached. We next describe our observations.

**Streams for different `.onion` domains are not multiplexed in the same circuit.** When the Tor process receives a stream to connect to a `.onion` domain, it checks if it already has an existing RP circuit connected to it. If it does, it attaches the stream to the same circuit. If not, it will build a new RP circuit. We verified this by examining a 7-hour log from the experiment described above. We found that around 560 RP circuits were created, and each was used to connect to a different `.onion` domain.

**Tor does not use IP or RP circuits for general streams.** Tor assigns different purposes to circuits when they are established. For streams accessing non-hidden servers, they use general purpose circuits. These circuits can carry multiple logical connections; i.e., Tor multiplexes multiple non-hidden service streams into one circuit. On the other hand, streams accessing a `.onion` domain are assigned to circuits that have a rendezvous-related purpose, which differ from general circuits. We verified the behavior through our experiments, and also by reviewing Tor's specification and the source code.

### 3.2 Hidden Service Traffic Experiment

The goal of this experiment is to understand the usage of IP and RP circuits from the hidden server and from the client points of view. We deployed a hidden service on the live Tor network through which a client could visit a cached version of any hidden service from our list above, which we had previously crawled and downloaded. Our hidden service was simultaneously accessed by our five separate Tor instances, four of which use `wget`, while one uses `firefox`. Every client chooses a random page from our list of previously crawled hidden pages and requests it from our HS. Again, all clients pause between fetches for a duration that is drawn from a distribution of user think times. During the whole hour, we logged the usage of the IP and RP circuits observed from our hidden server, and we logged the RP and IP circuits from our 5 clients. We ran this experiment more than 20 times over two months before analyzing the results.

In addition, to get client-side traffic from live hidden services, we also deployed our five clients described above to access our list of real Tor HSs, rather than our deployed HS.

Similarly, to understand the usage of general circuits, and to compare their usage to IP, and RP circuits, we also ran clients as described above, with the exception that the clients accessed general (non-hidden) websites using Alexa's top 1000 URL [1]. From our experiments, we generated the cumulative distribution function (CDF) of the DoA, the number of outgoing and incoming cells, which are shown in Figure 5a, 5b, and 5c. We present our observations below.

**IP circuits are unique.** Figure 5a shows the CDF of the DoA for different circuit types. Interestingly, we observe that IP circuits from the hidden service side (i.e., HS-IP) are long lived compared to other circuit types. We observe that the DoA of IP circuits showed an age of around 3600 seconds (i.e., an hour), which happens to be the duration of each experiment. This seems quite logical as these have to be long living connections to ensure a continuous reachability of the HS through its IP. Another unique aspect of the hidden services' IP circuits, shown in Figure 5b, was that they had exactly 3 outgoing cells (coming from the HS): 2 `extend` cells and one `establish_intro` cell. The number of incoming cells (from the IP to the HS) differ however, depending on how many clients connect to them. Intuitively, one understands that any entry guard could, possibly, identify an OP acting on behalf of an HS by seeing that this OP establishes with him long-lived connections in which it only sends 3 cells at the very beginning. Furthermore, from the number of incoming client cells, an entry guard can also evaluate the popularity of that HS.

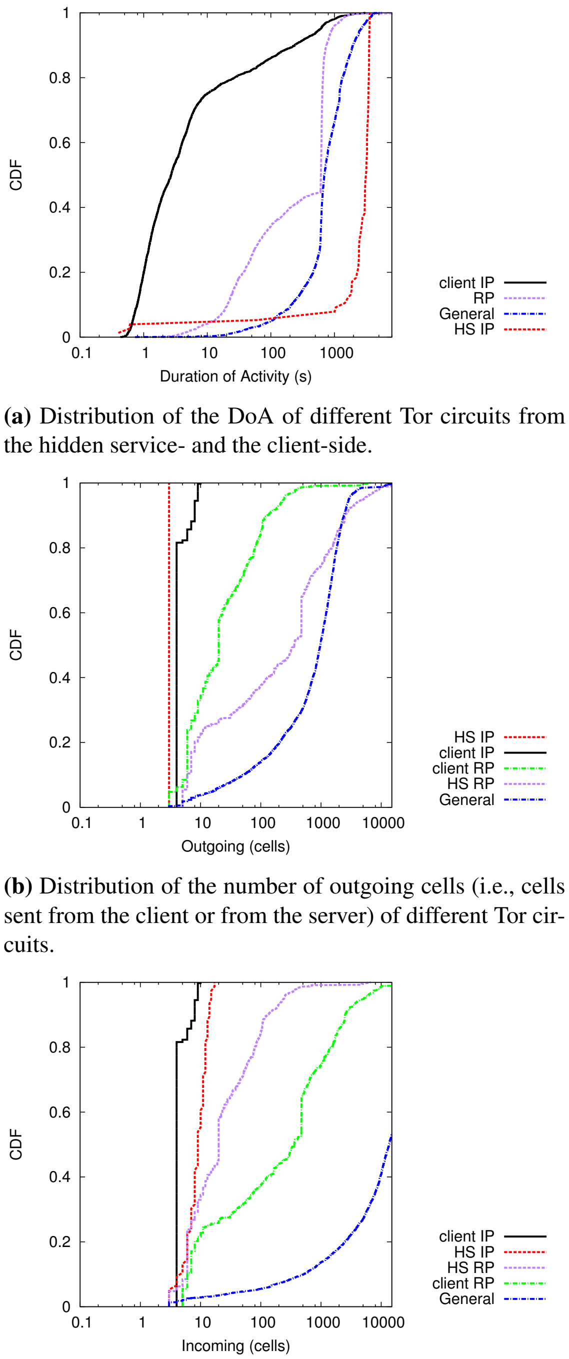
Client-IP circuits are also unique because they have the same number of incoming and outgoing cells. This is evidenced by the identical distributions of the number of incoming and outgoing cells shown in Figures 5b and 5c. For most cases, they had 4 outgoing and 4 incoming cells. The OP sends 3 `extend` and 1 `introduce1` cells, and receives 3 `extended` and 1 `introduce_ack` cells. Some conditions, such as RP failure, occasionally resulted in more exchanged cells, but IP circuits still had the same number of incoming and outgoing cells. Another unique feature was that, contrary to the HS-IP circuits, the Client-IP circuits are very short lived – their median DoA is around a second, as shown in Figure 5a, and around 80% of Client-IP circuits have a DoA that is less than or equal to 10 seconds. We expect this behavior as Client-IP circuits are not used at all once the connection to the service is established.

Active RP circuits, like general circuits, had a median DoA of 600 seconds, which is the expected lifetime of a Tor circuit. This was in particular observed with the clients which accessed our HS (the same RP circuit is reused to fetch different previously crawled pages). On the other hand, when the clients access live Tor hidden services, they have significantly lower DoA. Indeed, we observe (Figure 5a) that general circuits tend to have a larger DoA than RP circuits. The reason for this is that the same RP circuit is not used to access more than one hidden domain. Once the access is over, the circuit is not used again. On the other hand, general circuits can be used to access multiple general domains as long as they have not been used for more than 600 seconds.

**HS-RP circuits have more outgoing cells than incoming cells.** This is quite normal and expected since that circuit corresponds to the fetching of web pages on a server by a client. Typically, the client sends a few requests for each object to be retrieved in the page whereas the server sends the objects themselves which are normally much larger than the requests. There can be exceptions to this observation when, for instance, the client is uploading documents on the server or writing a blog, among other reasons.

Similarly, because RP circuits do not multiplex streams for different hidden domains, they are also expected to have a smaller number of outgoing and incoming cells throughout their DoA compared to active general circuits. As can be seen in Figures 5b, and 5c, one may distinguish between Client-RP and HS-RP circuits by observing the total number of incoming and outgoing cells. (Note that, as expected, the incoming distributions for the client and for the hidden service RP circuits from Figure 5c are the same as the outgoing distribution for hidden service and client RP, respectively, from Figure 5b.)

The incoming and outgoing distributions of RP circuits are based on fetching a hidden page, so the distributions we see in the figures might represent baseline distributions, and in the real network, they may have more incoming and outgoing cells based on users' activity. Although the exact distributions of the total number of incoming and outgoing cells for RP circuits is based on our models and may not reflect the models of users on the live network, we believe that the general trends are realistic. It is expected that clients mostly send small requests, while hidden services send larger pages.



**(a)** Distribution of the DoA of different Tor circuits from the hidden service- and the client-side.

**(b)** Distribution of the number of outgoing cells (i.e., cells sent from the client or from the server) of different Tor circuits.

**(c)** Distribution of the number of incoming cells (i.e., cells sent to the client or from the server) of different Tor circuits.

**Figure 5:** Cumulative distribution functions showing our observations from the experiments. Note that the X-axis scales exponentially.

**Table 1:** Edit distances of hidden pages across several weeks.

| Edit distance | 1 week | 2 weeks | 3 weeks | 8 weeks |
|---|---|---|---|---|
| Q1 | 1 | 0.997 | 0.994 | 0.980 |
| Median | 1 | 1 | 1 | 1 |
| Q3 | 1 | 1 | 1 | 1 |
| Mean | 0.96 | 0.97 | 0.96 | 0.927 |

**Table 2:** Edit distances of Alexa pages across several weeks.

| Edit distance | 1 week | 2 weeks | 3 weeks | 8 weeks |
|---|---|---|---|---|
| Q1 | 0.864 | 0.846 | 0.81 | 0.71 |
| Median | 0.95 | 0.94 | 0.92 | 0.88 |
| Q3 | 0.995 | 0.990 | 0.98 | 0.96 |
| Mean | 0.90 | 0.88 | 0.86 | 0.8 |

## 3.3 Variability in Hidden Pages

Over a period of four weeks, we downloaded the pages of more than 1000 hidden services once per week. We then computed the edit distance, which is the number of insertions, deletions, and substitutions of characters needed to transform the page retrieved at time $T$ with the ones retrieved at time $T+k$ weeks (with $k \in [1..8]$). Table 1 shows the three quartiles and the mean for the distribution of edit distances computed, which demonstrates that the pages remained almost identical. For comparison, we also downloaded the pages of Alexa's top 1000 URLs, and computed the edit distances in Table 2. This is not surprising since the sources of variations in the pages are mostly due to dynamism, personalized advertisements, or different locations. None of these sources is applicable to hidden services since clients are anonymous when they initiate the connections. Note that hidden services may implement personalized pages for a user after he or she logs into his or her account; however in the context of this paper, we are mainly concerned with the retrieval of the very first page.

## 4 Threat Model

Alice's anonymity is maintained in Tor as long as no single entity can link her to her destination. If an attacker controls the entry and the exit of Alice's circuit, her anonymity can be compromised, as the attacker is able to perform traffic or timing analysis to link Alice's traffic to the destination [5, 23, 25, 32]. For hidden services, this implies that the attacker needs to control the two entry guards used for the communication between the client and the hidden service. This significantly limits the attacker, as the probability that both the client and the hidden service select a malicious entry guard is much lower than the probability that only one of them makes a bad choice.

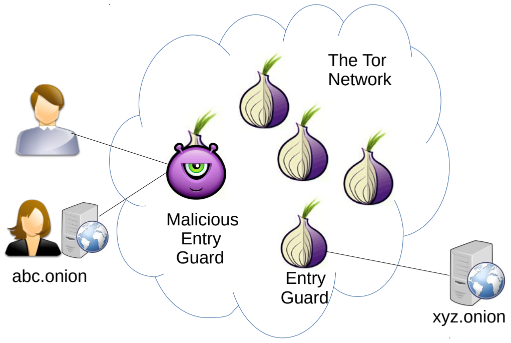


**Figure 6:** Our adversary can be a malicious entry guard that is able to watch all circuits

Our goal is to show that it is possible for a local passive adversary to deanonymize users with hidden service activities without the need to perform end-to-end traffic analysis. We assume that the attacker is able to monitor the traffic between the user and the Tor network. The attacker's goal is to identify that a user is either operating or connected to a hidden service. In addition, the attacker then aims to identify the hidden service associated with the user.

In order for our attack to work effectively, the attacker needs to be able to extract circuit-level details such as the lifetime, number of incoming and outgoing cells, sequences of packets, and timing information. We note that similar assumptions have been made in previous works [10, 35, 36]. We discuss the conditions under which our assumptions are true for the case of a network admin/ISP and an entry guard.

**Network administrator or ISP.** A network administrator (or ISP) may be interested in finding out who is accessing a specific hidden service, or if a hidden service is being run from the network. Under some conditions, such an attacker can extract circuit-level knowledge from the TCP traces by monitoring all the TCP connections between Alice and her entry guards. For example, if only a single active circuit is used in every TCP connection to the guards, the TCP segments will be easily mapped to the corresponding Tor cells. While it is hard to estimate how often this condition happens in the live network, as users have different usage models, we argue that the probability of observing this condition increases over time.

**Malicious entry guard.** Controlling entry guards allows the adversary to perform the attack more realistically and effectively. Entry guards are in a perfect position to perform our traffic analysis attacks since they have full visibility to Tor circuits. In today's Tor network, each OP chooses 3 entry guards and uses them for

45 days on average [16], after which it switches to other guards. For circuit establishment, those entry guards are chosen with equal probability. Every entry guard thus relays on average 33.3% of a user's traffic, and relays 50% of a user's traffic if one entry guard is down. Note that Tor is currently considering using a single fast entry guard for each user [13]. This will provide the attacker with even better circuit visibility which will exacerbate the effectiveness of our attack. This adversary is shown in Figure 6.

## 5 Circuit Fingerprinting Attack

In this section, we present our circuit fingerprinting attacks. Our attack allows an adversary to accurately and efficiently identify the presence of hidden service activity of a client or a server, and the circuit used to communicate with or by the hidden service (i.e., RP circuit). We first present an attack feasible for a more traditional attacker. Then, we describe a stronger attack for a more powerful adversary who can see more of the circuits from a user.

### 5.1 Classifying Special Circuits

Since the attacker is monitoring thousands of users, who produce hundreds of thousands of circuits, it is important to find an easy and straightforward approach to flag potentially "interesting" circuits for further examination. The attacker can exploit the simple and surprisingly distinctive features exhibited by IP and RP circuits (both client and hidden service side) to identify those circuits. In particular, we use the following features which are based on our observations in Section 3:

- **Incoming and outgoing cells**: This category of features will be useful in identifying IP circuits. For example, if a circuit sends precisely 3 cells, but has slightly more incoming cells (within a 1-hour duration), then this circuit is HS-IP with a high probability. Furthermore, if a circuit sends more than 3 cells, but has the exact same number of incoming and outgoing cells, then it is a client-IP with a high probability. This feature is also useful in distinguishing Client-RP from HS-RP circuits since we expect that HS-RP circuits to have more outgoing than incoming cells, and vice-versa for Client-RP circuits.

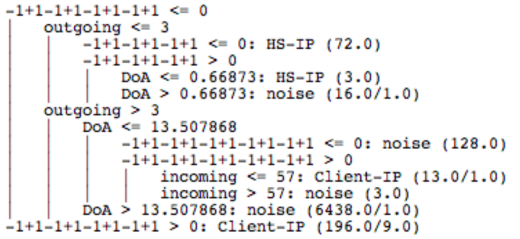
- **Duration of activity**: This feature is useful in distinguishing three groups of circuits: Client-IP circuits, HS-IP circuits, and all other circuits consisting of general, Client-, and HS-RP circuits. Recall that HS-IP circuits are long lived by design in order to be contacted by all interested clients, whereas client-IP circuits are inactive after performing the introduction process between the client and the hidden service, and have a median DoA of 1 second. Active general, Client-RP and HS-RP circuits can be alive and have a median of 600 seconds, which is the default lifetime of a circuit in Tor.

- **Circuit construction sequences**: We represent each of the first 10 cells (enough cells to capture the sequence of circuit establishment) either by the string -1 or +1. Each string encodes the direction of the corresponding cell. For example, the sequence "-1-1+1" corresponds to two outgoing cells followed by one incoming cell. This feature is useful in distinguishing Client-RP circuits from general and HS-RP circuits. The reason is that the circuit construction cell sequences in the case of Client-RP circuits differs from HS-RP and general circuits. This can be observed in Figures 1, 2, and 3. For example, we noticed that the sequence -1+1-1+1-1+1+1-1+1-1 is very common in Client-RP circuits, which corresponds to the sequence between the OP and G1 in Figure 3. However, HS-RP and general circuits have similar sequences so this feature alone cannot differentiate between those two circuit types.

**Strategy.** Different features are more indicative of certain circuit types. To best exploit those features, we perform our classification in two steps. First, the adversary looks for Client-IP and HS-IP circuits since those are the easiest ones to classify. This also allows the adversary to figure out if he is monitoring a HS or client of a HS. In the second step, the adversary examines the non-IP circuits to find RP circuits among them.

We use decision-tree classification algorithms, since identifying IP and RP circuits is dependent on an if-then-else conditional model as we discussed above. Tree-based algorithms build decision trees whose internal nodes correspond to the tests of features, and the branches correspond to the different outcomes of the tests. The leaves of the tree correspond to the classes, and the classification of a test instance corresponds to selecting the path in the tree whose branch values best reflect the new testing instance. Decisional trees have been used previously in the traffic classification literature [27, 4, 26] and are ideal for our problem.

Figures 7 and 8 depict decision trees which we use in the first step of this attack to identify the IP circuits. Note that general and RP circuits are treated as "noise". The tree in Figure 7 uses all features described above, and has a size of 15 nodes and 8 leaves, whereas the tree in Figure 8 omits the sequences, and only relies on incoming/outgoing packets and the DoA, which results in 10 leaves and a total size of 19 nodes. Both trees are very



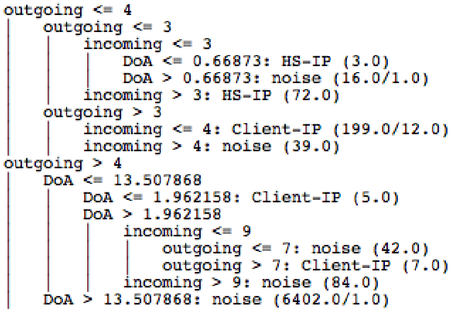
```
-1+1-1+1-1+1-1+1 <= 0
|   outgoing <= 3
|   |   -1+1-1+1-1+1 <= 0: HS-IP (72.0)
|   |   -1+1-1+1-1+1 > 0
|   |   |   DoA <= 0.66873: HS-IP (3.0)
|   |   |   DoA > 0.66873: noise (16.0/1.0)
|   outgoing > 3
|   |   DoA <= 13.507868
|   |   |   -1+1-1+1-1+1-1+1-1+1 <= 0: noise (128.0)
|   |   |   -1+1-1+1-1+1-1+1-1+1 > 0
|   |   |   |   incoming <= 57: Client-IP (13.0/1.0)
|   |   |   |   incoming > 57: noise (3.0)
|   |   DoA > 13.507868: noise (6438.0/1.0)
-1+1-1+1-1+1-1+1 > 0: Client-IP (196.0/9.0)
```

**Figure 7:** Decisional Tree (C4.5 algorithm) used in identifying IP circuits when cell sequences are used.

small, which allows for efficient classification. We discuss their performance in Section 6.1.

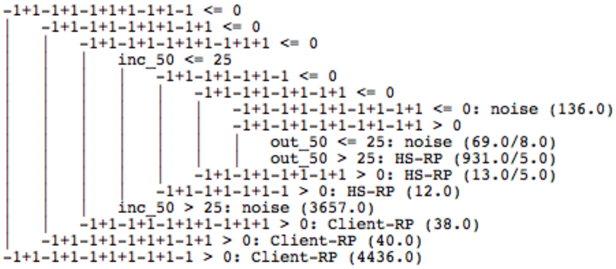
Once the adversary succeeds in identifying IP circuits, he is able to mark suspicious clients, and he can proceed to identifying their RP circuits. This can reduce his classification costs, and false positives. One challenge in distinguishing RP circuits from general circuits is that we cannot rely on DoA or the total number of incoming and outgoing cells as we did for IP circuits: in the case of general and RP circuits, those values are based on the user activity and can be biased by our models. To avoid such biases, we rely again on features that are protocol-dependent rather than user-dependent. Using our observation about sequences of Client-RP described previously, we can classify the circuit. Finally, to distinguish between HS-RP and general circuits, we use the first 50 cells of each circuit, and count the number of its incoming and outgoing cells. HS-RP circuits will generally have more outgoing than incoming, and the opposite should be true for general browsing circuits.

Figure 9 depicts a decision tree for classifying Client-RP, HS-RP and general circuits. It can be seen from the tree that Client-RP circuits are completely distinguished by their packet sequence fingerprint. Recall that those sequences represent the first 10 cells from the circuit, which is important as we want our sequences to be application independent. Also, HS-RP and general circuits are distinguished from each other by the fraction of incoming and outgoing cells of the first 50 cells. The tree contains a total of 17 nodes and only 9 leaves. We present the performance of this tree in Section 6.1.



```
outgoing <= 4
|   outgoing <= 3
|   |   incoming <= 3
|   |   |   DoA <= 0.66873: HS-IP (3.0)
|   |   |   DoA > 0.66873: noise (16.0/1.0)
|   |   incoming > 3: HS-IP (72.0)
|   outgoing > 3
|   |   incoming <= 4: Client-IP (199.0/12.0)
|   |   incoming > 4: noise (39.0)
outgoing > 4
|   DoA <= 13.507868
|   |   DoA <= 1.962158: Client-IP (5.0)
|   |   DoA > 1.962158
|   |   |   incoming <= 9
|   |   |   |   outgoing <= 7: noise (42.0)
|   |   |   |   outgoing > 7: Client-IP (7.0)
|   |   |   incoming > 9: noise (84.0)
|   DoA > 13.507868: noise (6402.0/1.0)
```

**Figure 8:** Decisional Tree (C4.5 algorithm) used in identifying IP circuits when cell sequences are not used



```
-1+1-1+1-1+1+1-1+1-1 <= 0
|   -1+1-1+1-1+1+1-1+1 <= 0
|   |   -1+1-1+1-1+1+1-1+1+1 <= 0
|   |   |   inc_50 <= 25
|   |   |   |   -1+1-1+1-1+1-1 <= 0
|   |   |   |   |   -1+1-1+1-1+1-1+1 <= 0
|   |   |   |   |   |   -1+1-1+1-1+1-1+1-1+1 <= 0: noise (136.0)
|   |   |   |   |   |   -1+1-1+1-1+1-1+1-1+1 > 0
|   |   |   |   |   |   |   out_50 <= 25: noise (69.0/8.0)
|   |   |   |   |   |   |   out_50 > 25: HS-RP (931.0/5.0)
|   |   |   |   |   -1+1-1+1-1+1-1+1 > 0: HS-RP (13.0/5.0)
|   |   |   |   -1+1-1+1-1+1-1 > 0: HS-RP (12.0)
|   |   |   inc_50 > 25: noise (3657.0)
|   |   -1+1-1+1-1+1+1-1+1+1 > 0: Client-RP (38.0)
|   -1+1-1+1-1+1+1-1+1 > 0: Client-RP (40.0)
-1+1-1+1-1+1+1-1+1-1 > 0: Client-RP (4436.0)
```

**Figure 9:** Decisional Tree (C4.5 algorithm) used in identifying RP circuits out of web browsing circuits.

## 5.2 Correlating Two Circuits

As mentioned in Section 4, Tor is considering using only a single entry guard per user. This changes the adversarial model: a malicious entry guard can now see *all* of the circuits used by a connected user. In this scenario, the attacker can see both IP and RP circuits. Even for a traditional entry guard, it has at least 11-25% chance of seeing both circuits. Such an attacker can leverage the fact that *the process of establishing the connection with the hidden service is fingerprintable*.

A client accessing a hidden service will exhibit a different circuit construction and data flow pattern from that of a client accessing a non-hidden service. For a client accessing a hidden service, the OP first builds the RP circuit, and simultaneously starts building a circuit to the IP. In contrast, a client visiting a regular website only establishes one circuit. (Figures 1 and 4 in Section 2 illustrate the exact flow of cells.) Using this fact, the attacker can classify behavior of *pairwise circuits*, and learn the RP of a user. In particular, we show that the first 20 cells of a circuit pair, which include all the cells used to establish connections with IP and RP, are enough to identify IP-RP pairs.

# 6 Evaluation

To evaluate our features with different machine learning algorithms, we used Weka [18], a free and open-source suite which provides out-of-the-box implementation of various machine learning algorithms. We experimented with the following algorithms: CART [8], which builds binary regression trees based on the Gini impurity, C4.5 [34], which uses information gain to rank possible outcomes, and $k$-nearest neighbors ($k$-NN for short), which considers the neighbors that lie close to each other

**Table 3:** Number of instances of different circuit types

| Dataset | HS-IP | HS-RP | Client-IP | Client-RP | general |
|---|---|---|---|---|---|
| IP-Noise | 76 | 954 | 200 | 4514 | 3862 |
| RP-Noise | N/A | 954 | N/A | 4514 | 3862 |

in the feature space. When the $k$-NN classifier is used, we set $k = 2$ with a weight that is inversely proportional to the distance. For each algorithm, we study the true positive rate ($TPR = \frac{TP}{TP+FN}$) and the false positive rate ($FPR = \frac{FP}{TN+FP}$). Specifically, TPR is the rate of correctly identified sensitive class, and FPR is the rate of incorrectly identified non-sensitive class. We collected network traces over the live Tor networks for our clients and server, and we did not touch or log the traffic of other users. We used the latest stable release of the Tor source code (tor-0.2.5.10) in all our experiments.

## 6.1 Accuracy of Circuit Classification

**Datasets.** From our long-term experiments described in Section 3, we extracted 5 types of circuits: Client-IP, Client-RP, HS-IP, HS-RP and general circuits. From every circuit, we created an instance consisting of its sequences (first 10 cell), DoA, total incoming and outgoing number of cells within the first 50 cells in the circuit, and a class label corresponding to the circuit type. Furthermore, since Tor is mainly used for web browsing [29], we designed our datasets so that most of the instances are "general" circuits to reflect realistically what an adversary would face when attempting to classify circuits of real users on the live network.

Recall that our general circuits are generated by browsing a random page from the top 1000 websites published by Alexa [1]. This list contains very small webpages (such as localized versions of google.com), and large websites (such as cnn.com). While it is not clear if this set of websites represents what real Tor users visit, we believe that this would not affect our approach since our features are protocol-dependent rather than website- or user-dependent. This is also true about our RP circuits. Therefore, we believe that the specific models and websites should not have an impact on our classification approach. Table 3 shows the number of instances of every class for both datasets.

Since we perform the classification in two steps, we created the following datasets:

- IP-Noise dataset: This dataset consists of 76 HS-IP circuits, 200 Client-IP circuits, and 6593 "noise" circuits. The 200 Client-IP circuits were selected uniformly at random from a large collection of 4514 Client-IP circuits.[2] The circuits labeled with the class "noise" consist of 954 HS-RP, 4514 Client-RP, and 3862 general browsing circuits.

- RP-Noise dataset: This dataset contains 200 Client-RP, 200 HS-RP circuit, and 3862 "noise" circuits (general browsing). The Client-RP and HS-RP circuits were selected uniformly at random from our collection of 954 and 4514 HS-RP and Client-RP circuits, respectively. Again, our goal is to imitate the conditions that the adversary would most likely face on the live network, where the majority of circuits to be classified are general browsing circuits.

**Results.** We used $n$-fold cross-validation for the three classification algorithms. This is a validation technique where the dataset is divided into $n$ subsets and $n-1$ subsets are used for training and 1 subset is used for testing, and the process is repeated $n$ times, where each subset is used for validation exactly once. Finally, the results from all $n$ folds are averaged. We set $n$ to 10 for our experiments.

We found that both C4.5 and CART perform equally well in classifying both datasets. We also found that $k$-NN performs well when cell sequences are used as features but otherwise performs poorly. For the IP-Noise dataset, when cell sequences are not used as a feature, as shown in Figure 10, the per-class TPR for CART ranges between 91.5% (Client-IP class) and 99% (for noise class), whereas the per-class accuracy for C4.5 ranges between 95.5% (Client-IP), and 99.8% (for noise class). $k$-NN performs worse with a TPR ranging from 55% (for HS-IP class) and 99% (for noise class). $k$-NN also has a high FPR for the noise class that exceeds 20%. Both C4.5 and Cart have 0% FPR for HS-IP, and have 0.2% and 0.1% FPR for Client-IP, respectively. However, we found that Cart has 7% FPR for the noise class because 17 out of 200 Client-IP instances got misclassified as noise. Therefore, based on the TPR and FPR rates, we conclude that C4.5 outperforms $k$-NN and Cart for the IP-Noise dataset when no sequences are used as features.

Figure 11 shows that when sequences are used as classification features, all three classifiers perform very well, but C4.5 still outperforms both Cart and $k$-NN with a nearly perfect per-class TPR. Interestingly, all classifiers provide 0% FPR for HS-IP and very low FPR for Client-IP and noise. We note that C4.5 also provides the best performance since it provides the highest TPR and lowest FPR among other classifiers.

[2] Recall that Client-IP are short-lived and one of these circuits is created every time a client attempts to connect to a HS, whereas HS-IP circuit samples are the most difficult to obtain since we observe each of them for an hour before we repeat experiments.

**Table 4:** Impact of different features on the TPR and FPR for the RP-Noise dataset. The table shows the accuracy results (TPR / FPR) if individual categories of features are used.

| TPR/FPR | Sequences | Cells | Sequences and Cells |
|---|---|---|---|
| HS-RP | 0% / 0% | 95% / 0.1% | 95.5% / 0.1% |
| Client-RP | 98% / 0% | 15% / 0.3% | 99% / 0% |
| Noise | 100% / 46% | 99.5% / 44.8% | 99.9% / 2.5% |

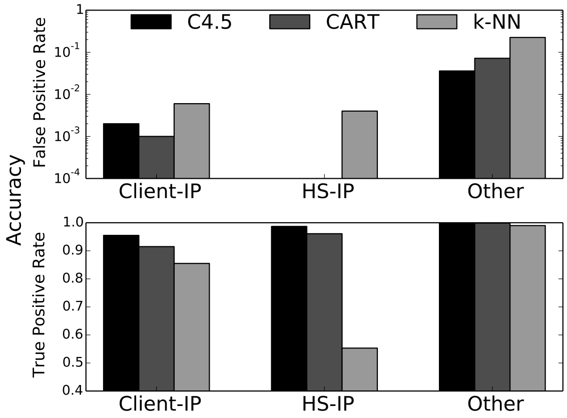


**Figure 10:** TPR and FPR of circuit classification with 3 classes when no cell sequences are used. FPR is shown in log scale.

For the RP-Noise dataset, we observe that both C4.5 and Cart provide identical performances in terms of TPR and FPR as shown in Figure 12. Both provide very high TPR for all classes, and 0% FPR for HS-RP and Client-RP classes. The FPR achieved by C4.5 and CART for the noise class is also low at 3%. *k*-NN provides slightly higher TPR for the HS-RP class than CART and C4.5, but the TPR is very similar to that achieved by CART and C4.5. We thus conclude that all classifiers perform equally well for the RP-noise dataset. Table 4 shows the impact on the overall accuracy based on different features.

## 6.2 Accuracy of Circuit Correlation

**Datasets.** We collected data of both the clients' and the servers' IP and RP circuit pairs for different hidden services. To collect client side circuit pairs, we used both `firefox` and `wget` (with flags set to mimic a browser as much as possible) to connect to hidden and non-hidden services from many different machines, each with one Tor instance. Each client visited 1000 different hidden services and 1000 most popular websites [1] several times. To collect server side circuit pairs, we spawned our own hidden service, and had multiple clients connect and view a page. The number of simultaneously connecting clients ranged from 1 to 10 randomly.

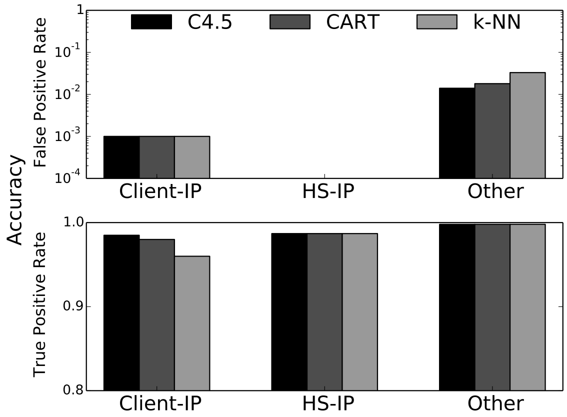


**Figure 11:** TPR and FPR of circuit classification with 3 classes when cell sequences are used. FPR is shown in log scale.

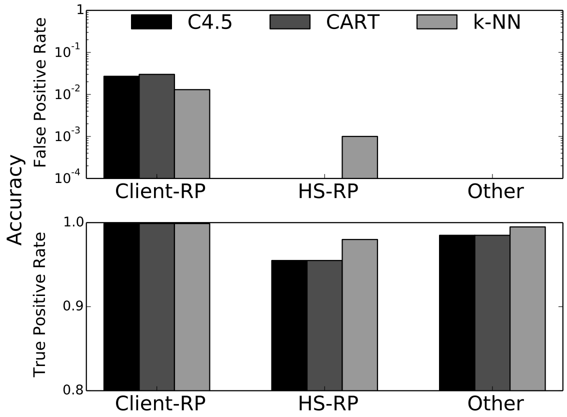


**Figure 12:** TPR and FPR of circuit classification with 3 classes. FPR is shown in log scale.

We then extracted traces of the first 20 cells of 6000 IP-RP circuit pairs (3000 client and 3000 server pairs) and 80000 of non-special circuit pairs. The non-special circuit pairs included any combination that is not IP-RP (i.e., IP-general, RP-general, general-general).

**Result.** The accuracy of IP-RP classification is shown in Table 5. We again used 10-fold cross validation to evaluate our attack. We can see that IP-RP circuit pairs are very identifiable: all three algorithms have 99.9% true positive rate, and less than 0.05% false positive rate. This accuracy is likely due to the uniqueness of the exact sequence of cells for IP-RP circuits. From the 6000 sequences of IP-RP pairs and 80000 non-special pairs, there were 923 and 31000 unique sequences respectively. We found that only 14 sequences were shared between the two classes. Furthermore, of those 14 sequences, only 3 of them had more than 50 instances.

This result implies that an adversary who can see a user's IP and RP (e.g., entry guard) can classify IP and RP circuits with almost 100% certainty by observing a

**Table 5:** IP/RP Pair Classification

| Algorithm | True Positive Rate | False Positive Rate |
|---|---|---|
| CART | 0.999 | $2.07 \cdot 10^{-4}$ |
| C4.5 | 0.999 | $3.45 \cdot 10^{-4}$ |
| *k*-NN | 0.999 | $6.90 \cdot 10^{-5}$ |

few cells. Moreover, the attack can be carried out in near real-time speed since we only need the first 20 cells. The attacker can thus effectively rule out most non-sensitive circuits, making data collection much easier.

## 7 Website Fingerprinting Revisited

In this section, we discuss the impact of our observations and attacks on WF, and show the result of applying modern WF techniques to hidden services. We show that the adversary can classify both the clients' and the operators' hidden service activities with high probability.

### 7.1 Adversaries Targeting Hidden Services

Juarez *et al.* [24] recently criticized various WF attacks because they made assumptions which were too advantageous for the adversary, and exacerbated the effectiveness of their attacks. In this section, we discuss some of the points that were raised by Juarez *et al.* [24] and show how our attacks address the concerns in the case of attacking hidden services.

**Noisy streams.** Previous WF attacks assumed that the adversary is able to eliminate noisy background traffic [10, 35, 36]. For example, if the victim's file download stream (noise) is multiplexed in the same circuit with the browsing stream (target), the attacker is able to eliminate the noisy download stream from the traces. With a lack of experimental evidence, such an assumption might indeed overestimate the power of the attack.

In the world of hidden services, we observed that Tor uses separate circuits for different `.onion` domains (Section 3). Furthermore, Tor does not multiplex general streams accessing general non-hidden services with streams accessing hidden services in the same circuit. From the attacker's perspective, this is a huge advantage since it simplifies traffic analysis; the attacker does not have to worry about noisy streams in the background of target streams. Furthermore, the previous assumption that the attacker can distinguish different pages loads is still valid [35]. User "think times" still likely dominate the browsing session, and create noticeable time gaps between cells.

**Size of the world.** All previous WF attacks have a problem space that is potentially significantly smaller than a realistic setting. Even in Wang *et al.*'s "large" open-world setting, the number of all websites are limited to 10,000 [35]. Moreover, different combinations of websites sharing one circuit could make it impossible to bound the number of untrainable streams. This implies that the false positive rate of WF techniques in practice is significantly higher, since the ratio of trained non-monitored pages to all non-monitored pages go down.

However, in the case of hidden services, the size of the world is significantly smaller than that of the world wide web. Also, while it is true that not all existing hidden services are publicly available, it has been shown that enumerating hidden services is possible [6][3]. In some cases, the attacker could be mainly interested in identifying a censored list of services that make their onion address public. Furthermore, we do not need to consider the blow up of the number of untrainable streams. Since RP always produces clean data, the number of untrained streams is bounded by the number of available hidden services.

**Rapidly changing pages.** The contents of the general web changes very rapidly as shown by Juarez *et al.* [24]. However, hidden pages show minimal changes over time (Section 3), contrary to non-hidden pages. The slowly changing nature of hidden services reduces the attacker's false positives and false negatives, and minimizes the cost of training. Furthermore, hidden services do not serve localized versions of their pages.

**Replicability.** Another assumption pointed out by Juarez *et al.* [24], which we share and retain from previous WF attacks, is the replicability of the results. That is, we are assuming that we are able to train our classifier under the same conditions as the victim. Indeed, we acknowledge that since it is difficult to get network traces of users from the live Tor network, we are faced with the challenge of having to design experiments that realistically model the behavior of users, hidden services, and the conditions of the network. That said, our attacks described above use features that are based on circuit interactions and are independent of the users' browsing habits or locations, which can reduce the false positive rate for the WF attacker.

Based on the above discussion, we claim that our attacker model is significantly more realistic than that of previous WF attacks [10, 35, 36]. While the conclusions made by Juarez *et al.* [24] regarding the assumptions of previous WF attacks are indeed insightful, we argue that many of these conclusions do not apply to the realm of hidden services.

[3] As pointed out by a reviewer, it is worth noting that the specific technique used in [6] has since been adressed by a change in the HS directory protocol.

## 7.2 Methodology

We first note here that hidden services have significantly lower uptime than a normal website on average. We found that only about 1000 hidden services were consistently up of the 2000 hidden services we tried to connect to. This makes collecting significant amounts of traces of hidden services very difficult. Furthermore, we found that hundreds of the available services were just a front page showing that it had been compromised by the FBI. This introduces significant noise to WF printing techniques: we now have hundreds of "different" pages that look exactly the same. We thus tried to group all of these hidden services as one website. This unfortunately limited our open world experiments to just 1000 websites. We also note that there may be other similar cases in our data, where a hidden service is not actually servicing any real content.

### 7.2.1 Data Collection

We gathered data to test OP servicing a normal user and a hidden service for both closed and open world settings. For the normal user case, we spawned an OP behind which a client connects to any website. We then used both `firefox` and `wget` to visit 50 sensitive hidden services that the attacker monitors (similar to experiments in Section 3). Our sensitive hidden service list contained a variety of websites for whistleblowing, adult content, anonymous messaging, and black markets. We collected 50 instances of the 50 pages, and 1 instance of 950 the other hidden services.

For the hidden service case, we first downloaded the contents of 1000 hidden services using a recursive `wget`. We then started our own hidden service which contains all the downloaded hidden service contents in a subdirectory. Finally, we created 5 clients who connect to our service to simulate users connecting to one server, and visiting a cached page. We then reset all the circuits, and visited a different cached page to simulate a different hidden service. We repeated this experiment 50 times for the 50 monitored hidden services, and once for the other 950 hidden services.

We argue that this setup generates realistic data for the following reasons. First, as shown in Section 3, the actual contents of hidden services changes minimally. Thus servicing older content from a different hidden service within our hidden service should not result in a significantly different trace than the real one. Second, the exact number of clients connected to the service is irrelevant once you consider the results in Section 6. An RP circuit correlates to one client, and thus allows us to consider one client trace at a time. Note that this is how a real-life adversary could generate training data to deanonymize the servers: it could run its own servers of the cached hidden services, and collect large samples of the servers' traffic patterns.

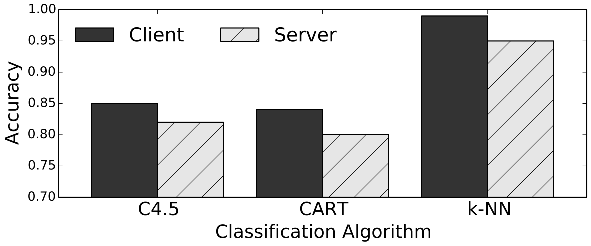


**Figure 13:** Accuracy of website fingerprinting attacks in closed world setting.

### 7.2.2 Website Fingerprinting Hidden Services

We extracted features similar to the ones presented in Wang *et al.* [35] from the data we collected.

- **General Features:** We use the total transmission size and time, and the number of incoming and outgoing packets.
- **Packer Ordering:** We record the location of each outgoing cell.
- **Bursts:** We use the number of consecutive cells of the same type. That is, we record both the incoming bursts and outgoing bursts, and use them as features.

We performed WF in closed and open world settings. In the closed world setting, the user visits/hosts a hidden service selected randomly from the list of 50 pages known to the attacker. In the open world setting, the user visits/hosts any of the 1000 pages, only 50 of which are monitored by the attacker. In either case, the attacker collects network traces of the Tor user, and tries to identify which service is associated with which network trace. We can consider the clients and the servers separately since we can identify HS-IP and Client-IP using the attack from Section 5.1 with high probability.

## 7.3 WF Accuracy on Hidden Services

We ran the same classifiers as the ones used in Section 6: CART, C4.5, and *k*-NN.[4] The accuracy of the classifiers in the closed world setting of both client and server is shown in Figure 13. For the open world setting, we varied the number of non-monitored training pages from 100 to 900 in 100 page increments (i.e., included exactly

[4] For *k*-NN, we tested with both Wang *et al.* [35] and the implementation in Weka, and we got inconsistent results. For consistency in our evaluation, we used the Weka version as with the other two classifiers.

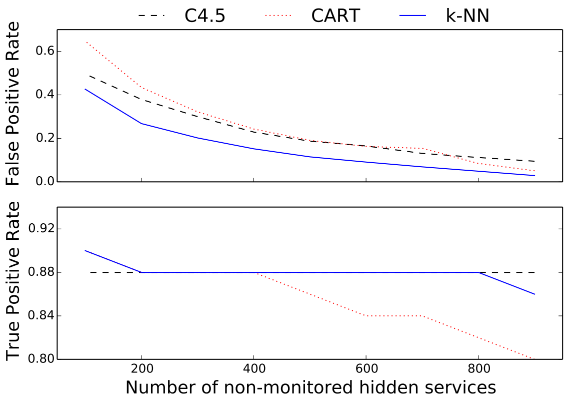


**Figure 14:** TPR and FPR of the client side classification for different classifiers.

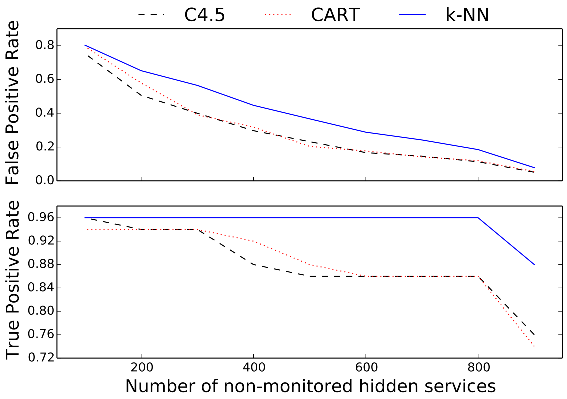


**Figure 15:** TPR and FPR of the server side classification for different classifiers.

one instance of the websites in the training set), and measured the TPR and FPR. The results of the open world experiment on the clients and the servers are shown in Figure 14 and Figure 15 respectively.

In all settings, we found that *k*-NN classifier works the best for classifying hidden services. We believe this is because *k*-NN considers multiple features simultaneously while the tree-based classifiers consider each feature one after another. In the closed world, the accuracy of *k*-NN was 97% for classifying the clients and 94.7% for the servers. In the open world, the *k*-NN classifier again performed the best. The TPR of *k*-NN reduced slightly as we increased the number of trained non-monitored websites. The TPR ranged from 90% to 88% and 96% to 88% for classifying clients and servers respectively. The FPR steadily decreased as we trained on more non-monitored websites: for classifying clients, it varied from 40% to 2.9% depending on the number of trained pages. Similarly, FPR of classifying servers varied from 80.3% to 7.8% for attacking servers.[5] Though the FPR is too a large for accurate classification when trained only on small number of non-monitored websites, we found that it quickly decreased as we increased the number of websites in the training set.

In general, the classifiers performed better in identifying clients' connections better than the hidden services servers; the TPR was comparable, but the FPR was significantly lower for classifying clients. We believe that this is, at least partially, due to the fact that we are using one real hidden service to emulate multiple hidden services; our data does not capture the differences in the differences in hardware, software, locations, and other characteristics real hidden service servers would have.

## 8 Future Possible Defenses

Our attacks rely on the special properties of the circuits used for hidden service activities. For the first attack (Section 5.1), we used three very identifiable features of the circuits: (1) DoA, (2) number of outgoing cells, and (3) number of incoming cells. To defend against this attack, Tor should address the three features. First, *all* circuits should have similar lifetime. Client IP and hidden service IP lasts either a very short or very long time, and this is very identifying. We recommend that circuits with less than 400 seconds of activity should be padded to have a lifetime of 400-800 seconds. Furthermore, we suggest that hidden services re-establish their connection to their IPs every 400-800 seconds to avoid any circuits from lasting too long. Second, hidden service and client IP should have a larger and varying number of outgoing and incoming cells. IPs are only used to establish the connection which limits the possible number of exchanged cells. We believe they should send and receive a random number of `PADDING` cells, such that their median value of incoming and outgoing cells is similar to that of a general circuit. We evaluated the effectiveness of this defense on the same dataset used in Section 6.1, and found that the true positive rate for the IPs and RPs fell below 15%. Once the features look the same, the classifiers cannot do much better than simply guessing.

To prevent the second attack (Section 5.2), we recommend that every circuit be established in a pair with the same sequence for the first few cells. If an `extend` fails for either circuit (which should be a rare occurrence), then we should restart the whole process to ensure no information is leaked. To do this efficiently, Tor could use its preemptive circuits. Tor already has the practice of building circuits preemptively for performance reasons. We can leverage this, and build the preemptive circuit

[5] The results are not directly comparable to previous WF attacks due to the differences in the settings, such as the size of the open world.

with another general circuit with the same sequence as the IP-RP pairs. This would eliminate the second attack.

For WF attacks (Section 7), defenses proposed by previous works [35, 9] will be effective here as well. Furthermore, for the clients, the results of Juarez *et al.* [24] suggest that WF attacks on hidden service would have significantly lower accuracy if an RP circuit is shared across multiple hidden service accesses.

## 9 Related Work

Several attacks challenging the security of Tor have been proposed. Most of the proposed attacks are based on side-channel leaks such as congestion [31, 17], throughput [30], and latency [21]. Other attacks exploit Tor's bandwidth-weighted router selection algorithm [5] or its router reliability and availability [7]. Most of these attacks are active in that they require the adversary to perform periodic measurements, induce congestion, influence routing, or kill circuits.

Our attacks on the other hand, like the various WF attacks, are passive. Other passive attacks against Tor include Autonomous Systems (AS) observers [15], where the attacker is an AS that appears anywhere between the client and his entry guard, and between the exit and the destination.

In addition, several attacks have been proposed to deanonymize hidden services. Øverlier and Syverson [32] presented attacks aiming to deanonymize hidden services as follows: the adversary starts by deploying a router in the network, and uses a client which repeatedly attempts to connect to the target hidden service. The goal is that, over time, the hidden service will choose the malicious router as part of its circuit and even as its entry guard to the client allowing the attacker to deanonymize him using traffic confirmation.

A similar traffic confirmation attack was described by Biryukov *et al.* [6]. The malicious RP sends a message towards the hidden service consisting of 50 padding cells when it receives the `rendezvous1` sent by the hidden service. This signal allows another malicious OR along the circuit from the hidden service to the RP, to identify the hidden service or its entry guard on the circuit. Biryukov *et al.* also show how it is possible for the attacker to enumerate all hidden services and to deny service to a particular target hidden service.

## 10 Conclusion

Tor's hidden services allow users to provide content and run servers, while maintaining their anonymity. In this paper, we present the first passive attacks on hidden services, which allow an entry guard to detect the presence of hidden service activity from the client- or the server-side. The weaker attacker, who does not have perfect circuit visibility, can exploit the distinctive features of the IP and RP circuit communication and lifetime patterns to classify the monitored circuits to five different classes.

For the stronger attacker, who has perfect circuit visibility (in the case where the client uses only one entry guard), the attacker runs a novel pairwise circuit correlation attack to identify distinctive cell sequences that can accurately indicate IP and RP circuits.

We evaluated our attacks using network traces obtained by running our own clients and hidden service on the live Tor network. We showed that our attacks can be carried out easily and yield very high TPR and very low FPR. As an application of our attack, we studied the applicability of WF attacks on hidden services, and we made several observations as to why WF is more realistic and serious in the domain of hidden services. We applied state-of-the-art WF attacks, and showed their effectiveness in compromising the anonymity of users accessing hidden services, and in deanonymizing hidden services. Finally, we propose defenses that would mitigate our traffic analysis attacks.

## 11 Code and Data Availability

Our data and scripts are available at `http://people.csail.mit.edu/kwonal/hswf.tar.gz`.

## 12 Acknowledgements

The authors thank Tao Wang and the reviewers for their useful feedback and comments. This reserach was supported in part by the QCRI-CSAIL partnership.

## References

[1] Alexa The Web Information Company. `https://www.alexa.com`.

[2] Tor Hidden Service Search. `https://ahmia.fi`.

[3] Tor. Tor Metrics Portal. `https://metrics.torproject.org/`.

[4] AlSabah, M., Bauer, K., and Goldberg, I. Enhancing tor's performance using real-time traffic classification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (New York, NY, USA, 2012), CCS '12, ACM, pp. 73–84.

[5] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., and Sicker, D. Low-Resource Routing Attacks Against Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)* (October 2007), pp. 11–20.

[6] Biryukov, A., Pustogarov, I., and Weinmann, R.-P. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2013), SP '13, IEEE Computer Society, pp. 80–94.

[7] Borisov, N., Danezis, G., Mittal, P., and Tabriz, P. Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity. In *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '07* (October 2007), pp. 92–102.

[8] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. CRC Press, New York, 1999.

[9] Cai, X., Nithyanand, R., Wang, T., Johnson, R., and Goldberg, I. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2014), CCS '14, ACM, pp. 227–238.

[10] Cai, X., Zhang, X., Joshi, B., and Johnson, R. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the 19th ACM conference on Computer and Communications Security (CCS 2012)* (October 2012).

[11] Dingledine, R. Using Tor Hidden Services for Good. `https://blog.torproject.org/blog/using-tor-good`, 2012. Accessed February 2015.

[12] Dingledine, R. Tor security advisory: "relay early" traffic confirmation attack. https://blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack, 2014. Accessed February 2015.

[13] Dingledine, R., Hopper, N., Kadianakis, G., and Mathewson, N. One Fast Guard for Life (or 9 months). `https://www.petsymposium.org/2014/papers/Dingledine.pdf`, 2015. Accessed February 2015.

[14] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium* (August 2004), pp. 303–320.

[15] Edman, M., and Syverson, P. As-awareness in Tor Path Selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security* (2009), CCS '09, pp. 380–389.

[16] Elahi, T., Bauer, K., AlSabah, M., Dingledine, R., and Goldberg, I. Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society* (2012), WPES '12, pp. 43–54.

[17] Evans, N., Dingledine, R., and Grothoff, C. A Practical Congestion Attack on Tor Using Long Paths. In *Proceedings of the 18th USENIX Security Symposium* (Berkeley, CA, USA, 2009), USENIX Association, pp. 33–50.

[18] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl. 11*, 1 (November 2009), 10–18.

[19] Hernández-Campos, F., Jeffay, K., and Smith, F. D. Tracking the Evolution of Web Traffic: 1995-2003. In *11th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2003), 12-15 October 2003, Orlando, FL, USA* (2003), pp. 16–25.

[20] Herrmann, D., Wendolsky, R., and Federrath, H. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naive Bayes Classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security* (2009), CCSW '09, pp. 31–42.

[21] Hopper, N., Vasserman, E. Y., and Chan-Tin, E. How Much Anonymity does Network Latency Leak? In *Proceedings of the 14th ACM conference on Computer and Communications Security* (October 2007), CCS '07.

[22] Jansen, R., Tschorsch, F., Johnson, A., and Scheuermann, B. The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2013* (2014).

[23] Johnson, A., Feigenbaum, J., and Syverson, P. Preventing Active Timing Attacks in Low-Latency Anonymous Communication. In *Proceedings of the 10th Privacy Enhancing Technologies Symposium (PETS 2010)* (July 2010).

[24] Juarez, M., Afroz, S., Acar, G., Diaz, C., and Greenstadt, R. A Critical Evaluation of Website Fingerprinting Attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2014), CCS '14, ACM, pp. 263–274.

[25] Levine, B. N., Reiter, M. K., Wang, C., and Wright, M. K. Timing Attacks in Low-Latency Mix-Based Systems. In *Proceedings of Financial Cryptography* (February 2004), pp. 251–265.

[26] Li, W., and Moore, A. W. A Machine Learning Approach for Efficient Traffic Classification. In *15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2007), October 24-26, 2007, Istanbul, Turkey* (2007), pp. 310–317.

[27] Luo, Y., Xiang, K., and Li, S. Acceleration of Decision Tree Searching for IP Traffic Classification. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (New York, NY, USA, 2008), ANCS '08, ACM, pp. 40–49.

[28] Mathewson, N. Some Thoughts on Hidden Services. `https://blog.torproject.org/category/tags/hidden-services`, 2014. Accessed February 2015.

[29] McCoy, D., Bauer, K., Grunwald, D., Kohno, T., and Sicker, D. Shining Light in Dark Places: Understanding the Tor Network. In *Proceedings of the 8th Privacy Enhancing Technologies Symposium* (July 2008), pp. 63–76.

[30] Mittal, P., Khurshid, A., Juen, J., Caesar, M., and Borisov, N. Stealthy Traffic Analysis of Low-Latency Anonymous Communication Using Throughput Fingerprinting. In *Proceedings of the 18th ACM conference on Computer and Communications Security* (2011), CCS '11, pp. 215–226.

[31] Murdoch, S. J., and Danezis, G. Low-cost traffic analysis of tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2005), SP '05, IEEE Computer Society, pp. 183–195.

[32] Øverlier, L., and Syverson, P. Locating Hidden Servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy* (May 2006), pp. 100–114.

[33] Panchenko, A., Niessen, L., Zinnen, A., and Engel, T. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES)* (October 2011), pp. 103–114.

[34] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[35] Wang, T., Cai, X., Nithyanand, R., Johnson, R., and Goldberg, I. Effective Attacks and Provable Defenses for Website Fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, Aug. 2014), USENIX Association, pp. 143–157.

[36] Wang, T., and Goldberg, I. Improved Website Fingerprinting on Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2013)* (November 2013), ACM.