Website Detection Using Remote Traffic Analysis

Xun Gong[1], Nikita Borisov[1], Negar Kiyavash[2], and Nabil Schear[3]

[1] Department of Electrical and Computer Engineering, UIUC
[2] Department of Industrial and Enterprise Systems Engineering, UIUC
[3] Department of Computer Science, UIUC
{xungong1,kiyavash,nikita,nschear2g}@illinois.edu

Abstract. Recent work in traffic analysis has shown that traffic patterns leaked through side channels can be used to recover important semantic information. For instance, attackers can find out which website, or which page on a website, a user is accessing simply by monitoring the packet size distribution. We show that traffic analysis is even a greater threat to privacy than previously thought by introducing a new attack that can be carried out remotely. In particular, we show that, to perform traffic analysis, adversaries do not need to directly observe the traffic patterns. Instead, they can gain sufficient information by sending probes from a far-off vantage point that exploits a queuing side channel in routers.

To demonstrate the threat of such remote traffic analysis, we study a remote website detection attack that works against home broadband users. Because the remotely observed traffic patterns are more noisy than those obtained using previous schemes based on direct local traffic monitoring, we take a dynamic time warping (DTW) based approach to detecting fingerprints from the same website. As a new twist on website fingerprinting, we consider a website detection attack, where the attacker aims to find out whether a user browses a particular web site, and its privacy implications. We show experimentally that, although the success of the attack is highly variable, depending on the target site, for some sites very low error rates. We also show how such website detection can be used to deanonymize message board users.

1 Introduction

Traffic analysis is the practice of inferring sensitive information from patterns of communication. Recent research has shown that traffic analysis applied to network communications can be used to compromise users' secrecy and privacy. By using packet sizes, timings, and counts, it is possible to fingerprint websites visited over an encrypted tunnel [2,4,11,17], infer keystrokes sent over a secure interactive connection [27,34] and even detect phrases in VoIP sessions [31–33]. These attacks have been explored in the context of a *local adversary* who can observe the target traffic directly on a shared network link or can monitor a wireless network from a nearby vantage point [25].

We consider an alternate traffic analysis approach that is available to *remote adversaries*. We notice that it is possible to infer the state of a router's queue through the observed queueing delay of a probe packet. By sending frequent probes, the attacker can measure the dynamics of the queue and thus learn an approximation of the sizes, timings, and counts of packets arriving at the router. In the case of home broadband networks, in particular, DSL lines, the attacker can send probe packets from a geographically distant vantage point, located as far away as another country; the large gap between the bandwidth of the DSL line and the rest of the Internet path makes it possible to isolate the queueing delay of the "last-mile" hop from that experienced elsewhere.

To demonstrate the feasibility of using remote traffic analysis for real malicious attack to learn sensitive information, we adapt the website fingerprinting attack [2,11,17], previously targeted at local victims, to a new scenario and introduce a remote website detection attack. Our attack can find out when a victim user under observation visits a particular target site without directly monitoring the user's traffic. This would allow, for example, a company to find out when its employees visit its competitors' sites *from their home computers* or deanonymize users of web boards.

In our adaptation, we encountered two challenges: the information obtained through remote traffic analysis is more noisy than in the local case, and there is no easily available training set from which to create a fingerprint. To address the former problem, we improved on the previous inference methodology, which used the distribution of packet sizes and inter-arrival times, and developed a fingerprint detection technique that makes use of ordered packet size sequences and the dynamic time warping (DTW) distance metric. To create a training set, we designed a testbed that uses an emulated DSL link and a virtual execution environment to replicate the victim's home environment.

To evaluate our work, we sent probes to a home DSL line in the United States from a rented server in a data center near Montreal, Canada; we chose this set up to demonstrate the low cost and barrier to entry to conduct the attack. We then compared the probe results with profiles of website fetches generated in a virtual testbed at our university. We tested our attack on detecting each of a list of 1 000 popular websites. We found that detection performance was highly variable; however, for a significant fraction of sites, it was possible to obtain very low false-positive rates without incurring significant false-negative rates. We also found that there is some accuracy loss due to the discrepancies in the test and training environments (distant from each other) that we were not (yet) able to eliminate. If the training and test data are both collected from the same location, a much larger fraction of sites can be accurately detected with low error rates. We find that despite working with a much noisier information source than previous web fingerprinting work [2, 11, 17], our website detection attack nevertheless shows that remote traffic analysis is a serious threat to Internet privacy.

The rest of the paper is organized as follows. We describe our approach to remote traffic analysis in §2. In §3 we describe our adaptation of previous website fingerprinting attack to remotely confirming user's browsing activities. We evaluate our website detection attack in §4. We then discuss further extensions and the limitations of our technique in §5 and present related work in §6, concluding in §7.

2 Remote Traffic Analysis

Traffic analysis attacks have been known to be effective for quite some time. And yet, for most Internet users, they represent a minor concern at best. Although a dedicated attacker could always intercept traffic by, say, bribing a rogue ISP employee, or tapping a switch box, he would run the risk of being caught and potentially incurring criminal

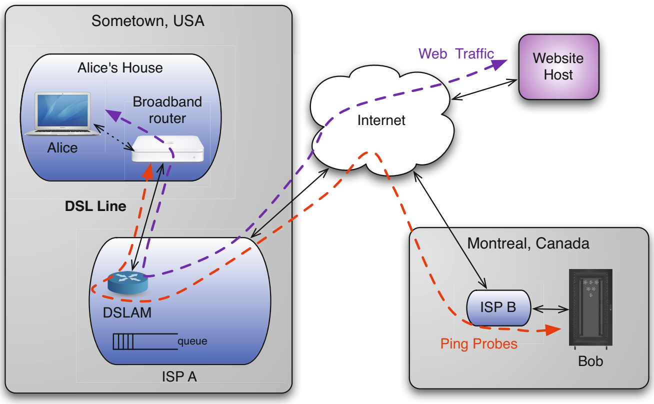


Fig. 1. Queueing side channel. Bob remotely sends probes to Alice's router to infer her activities.

charges. In any case, this level of effort seems justified only for highly sensitive material, rather than casual snooping; therefore, as long as sensitive data are protected by encryption or other techniques, a user may feel relatively safe.

We show, however, that traffic analysis can be carried out at a significantly lower cost, and by attackers who never come into physical proximity with the user. In fact, the attackers can launch their attacks from another state or country, as long as they have access to a well-provisioned Internet connection. This, in turn, is very easy to obtain due to the highly-competitive Internet hosting business sector: a virtual private server in a data center can cost as little as a few dollars a month.[1] We show that the attacker's traffic has very low rate, thus attackers do not need to incur high bandwidth costs. Furthermore, users who are being spied upon are unlikely to notice the small amount of performance overhead. Thus, anyone with a credit card[2] can carry out the attack and leave little trace.

In this section, we describe our approach to remote traffic analysis. We first introduce the queueing side channel, which is the basis of the attack. Then we design an algorithm to recover users' traffic patterns from the information leaked through this side channel.

2.1 Queuing Side Channel

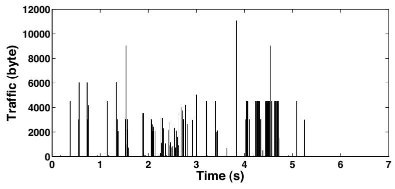
We consider the following scenario as depicted in Figure 1. Alice is a home user at Sometown USA, browsing a website via her DSL Internet connection. Her computer is connected to a broadband router, using a wireless or wired LAN connection.[3] The router is connected via a DSL line to a DSLAM[4] or similar device operated by her ISP, which is then (eventually) connected to the Internet. Unbeknownst to Alice, Bob, who is located in another state, or another country wishes to attack Alice's privacy. If Bob

[1] See, for example, `www.vpslink.com` (retrieved February 2011).

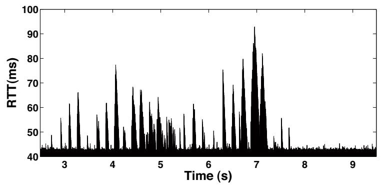
[2] Working stolen credit cards are an easily acquired commodity on the black market [10].

[3] In some cases, Alice's computer might be connected to the DSL line directly.

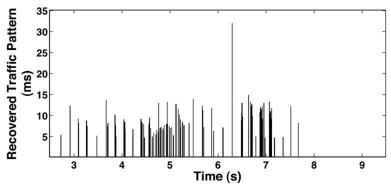
[4] DSL access multiplexer.



(a) Alice's traffic pattern



(b) RTTs measured by Bob



(c) Recovered traffic pattern

Fig. 2. Real traffic on a DSL vs. probe RTTs. Alice resides in Champaign, IL, while Bob is located in Montereal, Canada.

knows Alice's IP address (for example, if Alice visited a site hosted by Bob), he can use his computer to send a series of ICMP echo requests (pings) to the router in Alice's house and monitor the responses to compute the round-trip times (RTTs). One component of the RTTs is the queueing delay that the packets experience at the DSLAM prior to being transmitted over the DSL line; thus the RTTs leak information about the DSLAM queue sizes. This leakage in turn reveals traffic patterns pertaining to Alice's activities.

Since the probe packets traverse many Internet links, and the queuing delays on Alice's DSL link are but one component of the RTT, the question is, how much information is leaked by this side channel? Furthermore, can it be used to infer any information about Alice's activities? To evaluate the potential of this attack, we carried out a test on a home DSL link located in Champaign, IL, USA. In the test, Alice opens a Web page www.yahoo.com on her computer. Simultaneously, Bob in Montreal, QC, Canada sends a ping request every 10 ms to Alice's home router. Figure 2(a) depicts the traffic pattern of Alice's download traffic. The height of each peak in the figure represents the total size of packets that are downloaded during each 10 ms interval. Figure 2(b) plots the RTTs of Bob's ping requests. We can see a visual correlation between the traffic pattern and observed RTTs; whenever there is a large peak in the user's traffic, the attacker observes a correspondingly large RTT.

The correlation between Alice's traffic and Bob's observed probe RTTs can be explained as follows. The RTTs include both the queuing delay incurred on the DSL link and delays on intermediate routers, which sit between Bob's computer and Alice's router. The intermediate routers are typically well provisioned and are unlikely to experience congestions [1, 16]; furthermore, the intermediate links have high bandwidth and thus queueing delays will be small in all cases. We validate this using our own measurements in §4.1.

On the other hand, Alice's DSL link is, by far, the slowest link that both her traffic and Bob's probes are likely to traverse. The queue at Alice's router can grow to be quite

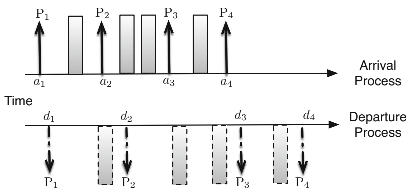


Fig. 3. FIFO queuing in the DSL router

long (in relative terms), due to TCP behaviors, which cause the `www.yahoo.com` server to send a batch of TCP packets at a fast rate. As most routers schedule packets in a First In First Out (FIFO) manner, this congestion will lead to large queuing delays of Bob's ping packets. We saw that the additional delay caused by Alice's incoming traffic could be as high as over 100 ms. Thus, Alice's traffic patterns are clearly visible in the RTTs seen by Bob.

2.2 Traffic Pattern Recovery Algorithm

We now show how the attacker can analyze the information leaked through this queueing side channel. We model the incoming DSL link as a FIFO queue. As most traffic volume in an HTTP session occurs on the download side, we will ignore the queuing behavior on the outgoing DSL link, though it could be modeled in a similar fashion.

Figure 3 depicts the arrival and departure process in this queuing system. The arrows are Bob's ping packets, denoted by P_i's, and the blocks represent HTTP packets downloaded by Alice. The DSLAM serves packets in FIFO manner and at a constant service rate; i.e., the service time is proportional to the packet size. As most HTTP packets are more than an order of magnitude larger than ping packets, we ignore the service time for pings.

Assume that ping packet P_i arrives in the queue at time a_i, waits for the router to serve all the packets currently in the router, and then departs at time d_i. Let us consider the observed RTT of the ping packet P_i; we can represent it as:

$$RTT_i = \sum_{l \in \text{links on path}} q_i^l + p_i^l + t_i^l \tag{1}$$

where q_i^l, p_i^l, and t_i^l are the queueing, propagation, and transmission delays incurred by packet P_i on link l. Note that the propagation and transmission delays are mostly constant, and in fact we can approximate:

$$\sum_{l \in \text{links on path}} p_i^l + t_i^l \approx \min_j RTT_j \tag{2}$$

since Bob is likely to experience near-zero queueing delays for some of the pings. Furthermore, as argued in §2.1, the queueing delay on links other than the DSL line are going to be minimal, thus we can further approximate:

$$RTT_i \approx \min_j RTT_j + (d_i - a_i) \tag{3}$$

Algorithm 1. Traffic pattern recovery algorithm

```
1: let $d_0 = 0$
2: for $i = 1$ to the probe sequence length do
3:     # reconstruct the arrival and departure times using the ping interval
4:     $a_i = t_{ping} \cdot i$
5:     $d_i = RTT_i - RTT_{min} + a_i$
6:     # estimate the total size of packets arriving in $[a_{i-1}, a_i]$
7:     $\widehat{s_i} = d_i - max(d_{i-1}, a_i)$
8:     # discard noise
9:     if $\widehat{s_i} < \eta$ then
10:        $\widehat{s_i} = 0$
11:    end if
12: end for
```

Making use of the queuing delay $d_i - a_i$ from (3), the attacker Bob can further infer the total size of HTTP packets arriving during the interval $[a_{i-1}, a_i]$'s, which produces a similar pattern as Alice's traffic in Figure 2(a). For this purpose, two cases need to be considered.

1. $a_i \geq d_{i-1}$. In this case, when P$_i$ enters the queue, the DSLAM is either idle or serving packets destined for Alice. The delay $d_i - a_i$ reflects the time required to finish serving the HTTP packets currently in the buffer, and is thus approximately proportional to the total size of Alice's arrivals during the interval $[a_{i-1}, a_i]$. P$_2$ in Figure 3 is one example of this case.
2. $a_i < d_{i-1}$. In this case, P$_{i-1}$ is still in the queue when P$_i$ arrives. Only after P$_{i-1}$ departs at d_{i-1}, the router can start to serve packets that arrived in the interval $[a_{i-1}, a_i]$. Thus the delay $d_i - d_{i-1}$ is the service time for those packets and can be used to recover the total size. P$_4$ in Figure 3 is one example of this case.

Algorithm 1 summarizes the traffic pattern recovery procedure based on these observations. To account for minor queueing delays experienced on other links, we define a threshold η such that RTT variations smaller than η are considered noise and do not correspond to any packet arrival at the DSLAM. Figure 2(c) plots the pattern extracted from RTTs in Figure 2(b). After processing, the resulting time series proportionally approximate the packet size sequence of the original traffic in Figure 2(a). As will be shown in the next section, it can be applied to infer more information about Alice's activities, e.g., website fingerprinting.

Note that in case 1, the attacker may underestimate the size of the HTTP packets arriving in the period $[a_{i-1}, a_i]$ because a portion of them will have already been served by time a_i. The error depends both on the frequency of the probes and the bandwidth of the DSL link. Since most HTTP packets are of maximal size (MTU), we can ensure that all such packets are observed by setting the ping period to be less than: $\frac{\text{MTU}}{\text{DSL bandwidth}}$. Thus the adversary must tune the probe rate based on the DSL bandwidth and faster links will require a higher bandwidth overhead (but the pings will form a constant, small fraction of the overall DSL bandwidth.)

3 Website Fingerprinting

Previous work on traffic analysis has shown that it is often possible to identify the website that someone is visiting based on traffic timings and packet sizes [2, 11, 17], namely, *website fingerprinting*. We consider whether it is possible to carry out a similar attack using our remote traffic analysis. We first review the three basic steps in previous work when conducting a website fingerprinting attack.

1. First, the attacker decides some feature of web traffic used to distinguish websites. The feature needs to stay relatively stable for accesses to the same single website, but has significant diversity across different sites. For example, Herrmann et al. use the size distribution of HTTP packets [11].
2. The next step is the *training* procedure. The attacker needs a training data set of fingerprint samples labeled with corresponding destination websites. Usually, these feature profiles are obtained by the attacker browsing websites himself/herself from the same (or similar) network connection as the user.
3. In the final step, the attacker *tests* his/her knowledge from training on the victim user. He/She monitors traffic going to the user and matches extracted features with the profiles in his/her database. The one with most similarity is chosen as the website browsed by the user.

As compared with previous work, using our remote traffic analysis technique for identifying websites introduces two additional challenges. First, previous work used fine-grained information like exact packet size distributions to create features, whereas in our setting this information is not available directly, since the queueing side channel produces only approximate sums of packet sizes. Second, previous work created a training set from *the same vantage point* that was then used for fingerprinting tests. An attacker performing remote traffic analysis must, of course, use a different environment for collecting the training set, potentially affecting the measured features. We describe our approaches to solving these two challenges next.

3.1 Time Series–Based Feature

Since it is hard to infer information about each single packet from our recovered pattern time series, we use the entire time series, which contains the estimated size of all HTTP packets downloaded during each probe period, to create one fingerprint trace. Identification of websites is based on the similarity between the observed fingerprints and samples in the training set.

The challenge is to find a meaningful distance metric between fingerprint traces. Note that pointwise comparisons will produce poor results. This is because parts of the fingerprint may be impacted by the noise from a small queueing delay on a core Internet link. Additionally, the fingerprint could miss some packets contained in the original traffic due to pattern recovery errors. Finally, even fingerprints of the same website are not strictly synchronized in time due to the inter-packet delay variations. To deal with these issues, we turn to the Dynamic Time Warping (DTW) distance [24]. DTW was developed for use in speech processing to account for the fact that when

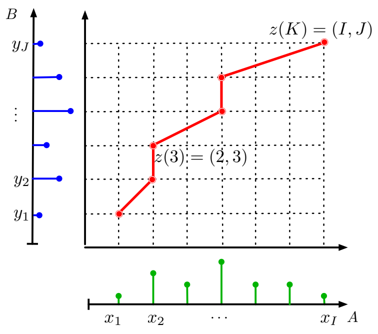


Fig. 4. Warping function in DTW

people speak, they pronounce various features of the phonemes at different speeds, and do not always enunciate all of the features. DTW attempts to find the best alignment of two time series by creating a non-linear time warp between the sequences. Figure 4 visualizes the DTW-based distance between two time series: $X = \{x_1, x_2 \ldots, x_I\}$ and $Y = \{y_1, y_2 \ldots, y_J\}$. Let function $F(z) = \{z(1), \ldots, z(K)\}$ be a mapping from series X to series Y where $z(k) = (x(i), y(j))$. For every pair of matched points based on the mapping, we define the distance as $d(z(k)) = d(i, j) = |x_i - y_j|$. The final distance between the X and Y can then be defined as a weighted and normalized sum over all matched point pairs as $D(X, Y) = \min_F \left\{ \frac{\sum_{k=1}^{K} d(z(k))w(k)}{\sum_{k=1}^{K} w(k)} \right\}$. The weights $w(k)$'s are flexible parameters picked based on the specific application scenario. Applying dynamic programming, one can find the warping function with minimum distance, which captures the similarity between the two time series under best matched alignment.

In our attack, we apply DTW-based distance to account for the estimation errors and time desynchronizations in fingerprints. Based on the distances with the training data set, the attacker will know if a test sample indicates the activity that the user browsed the website of interest.

3.2 Training Environment

To obtain an accurate training fingerprint for a particular user's traffic, the attacker must be able to replicate the network conditions on that user's home network. The approach we use is to set up a virtual machine running a browser that is connected to the Internet via a virtual Dummynet link [23]. The virtual machine is then scripted to fetch a set of web pages of interest; at the same time, an outside probe is sent across the Dummynet link, simulating the attack conditions on a real DSL link.

A number of parameters of the link need to be carefully decided. We found that the most important parameter for the attacker to replicate was the link bandwidth. First, as discussed in §2.2, the probe frequency should be adjusted based on the link bandwidth. Bandwidth also affects the magnitude of observed queuing delays. Additionally, it can significantly alter the traffic pattern itself, as TCP congestion control mechanisms are affected by the available bandwidth. Fortunately, estimating the bandwidth on a link is a well-studied problem [20, 22, 28]. In our tests, we use a packet-train technique by sending a burst of probe packets and measuring the rate at which responses are returned.

Since most DSL lines have asymmetric bandwidth, we used TCP ACK packets with 1000 data bytes to measure the download bandwidth on the link. The target would send a short TCP reset packet for each ACK that it received, with the spacing between resets indicating the downstream bandwidth; we found this method to be fairly accurate.

The round-trip time between the home router and the website hosts also affects the fingerprint. When opening a webpage, the browser can download objects from several host servers. The traffic pattern is the sum of all download connections, hence the shape of observed fingerprint does depend on the RTTs to these servers. However, we did not explicitly model this parameter considering the difficulty to accurately tune up the link delays to multiple destinations. The effects to the attack will be further discussed in §4.

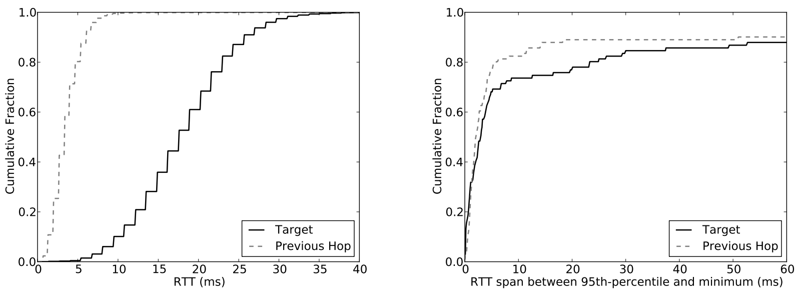
The fingerprint may be affected by the choice of browsers and operating systems as well; for best results, the training environment should model the target as closely as possible. Information about browser and operating system versions can be easily obtained if the target can be convinced to visit a website run by the attacker; additionally, fingerprinting techniques in [18] may be used to recover some of this information.

3.3 Attack Scenarios

We consider several attack scenarios that make use of website fingerprinting. We can first consider the classic website fingerprinting scenario: Bob obtains traces from Alice's computer by sending probes to her DSL router and compares them to fingerprints of websites that he has generated, in order to learn about her browsing habits. Note that this can be seen as a *classification* task: each web request in Alice's trace is classified as belonging to a set of sites. This scenario has been used in most of the previous work on website fingerprinting, but it introduces the requirement that Bob must know the set of potential sites that Alice may visit. Without some prior information about Alice's browsing habits, this potential set includes every site on the Internet, making it infeasible to generate a comprehensive set of fingerprints. One could create fingerprints for popular sites only, but this reduces the accuracy of the classification task [6,11,29]. For example, the top 1 000 US sites, as tracked by Alexa, are responsible for only 56% of all page views, therefore, even a perfect classifier trained on the 1 000 sites would give the wrong result nearly half the time.[5]

We therefore consider a different scenario, where Bob wants to *detect* whether Alice visits a *particular* site. For example, if Bob is Alice's employer, he may wish to check to see if she is considering going to work for Bob's competitor, Carol. To carry out this attack, Bob would create a fingerprint for Carol's jobs site; he would then perform a binary classification task on Alice's traffic, trying to decide whether a trace represents a visit to the target site or some other site on the Internet. As we will see, such binary classification can be performed with relatively high accuracy for some choices of sites. Note that, as Alice's employer, Bob has plenty of opportunities to learn information about Alice's home network, such as her IP address, browser and operating system versions, and download bandwidth, by observing Alice when she connects to a password-protected Intranet site, and can therefore use this information to create accurate training data for building fingerprints.

[5] In fact, the situation is even worse, since Alexa counts *all* page views within a certain top-level domain, whereas fingerprints must be created on each individual URL.



(a) Empirical CDF of ping RTTs from a typical host.

(b) Empirical CDF of 95th percentile minus minimum RTTs.

Fig. 5. Measurement of DSL probe variances

As another example, Bob may be trying to identify an employee who makes posts to a web message board critical of Bob.[6] Bob can similarly build profiles, tailored for each employee's home computer, of the web board and perform remote traffic analysis. He can then correlate any detected matches to the times of the posts by the offending pseudonym; note that this *deanonymization* attack is able to tolerate a significant number of false-positive and false-negative errors by combining observations over many days to improve confidence [7].

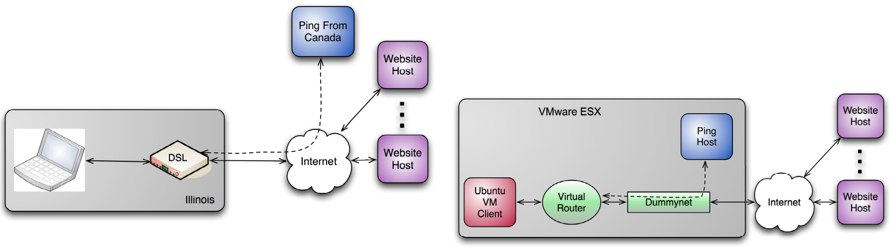
4 Evaluation

We next present our results of website detection attack. First, through measurements of DSL probe variances, we demonstrate the feasibility of the RTT based remote traffic analysis, which is the basis of our attack.

4.1 Measurement of DSL Probe Variance

To further confirm that the fluctuation of user's RTTs are primarily determined by the congestion at the DSL link, we conducted a small Internet measurement study. We harvested IP addresses from access logs of servers run by the authors. We noted that many DSL providers assign a DNS name containing "dsl" to customers. Using reverse DNS lookups, we were able to locate 918 potential DSL hosts. To determine each host's suitability for measurement, we first determine if it responds to ping requests. We then use traceroute to locate the hop before the target DSL host (e.g., the DSLAM). Next, we ensure this previous hop also responds to ping requests. Lastly, we measure the minimum RTT of several hundred ping probes. We exclude any host with a minimum RTT of greater than 100ms to bound the study to hosts in a wide geographical area around our Montreal, Canada probe server. Using this method, we identified 189 DSL

[6] This example is motivated by several actual cases of companies seeking to do this; see `https://www.eff.org/cases/usa-technologies-v-stokklerk` and `https://www.eff.org/cases/first-cash-v-john-doe`.



(a) The DSL-Setup for test (b) The VM-Setup for training

Fig. 6. Experimental setups for website detection

hosts to measure.[7] The measurement consists of sending ping probes every 2 ms for 30 seconds to the target DSL host and then to its previous hop. We collected these traces in a loop over a period of several hours.

We found that, on average, the target host RTT was $\sim$10 ms greater than the previous hop. We frequently observed the pattern in Figure 5(a) where the previous hop RTT was very stable and target RTT variations greater than 10 ms. We then measured the span between the 95th percentile and the minimum observation in each sample. Figure 5(b) shows the CDFs of this data for the each target DSL host and its previous hop from the measurement set. We see that the previous hop RTT span shows more stability than the end host, confirming the one of the primary assumptions of our work.

4.2 Attack Setups and Data Collection

We built a *DSL-Setup* consisting of a target system and a ping server, as shown in Figure 6(a). The target system captured the real environment of a home user. It ran on a laptop, located in Champaign, IL, connected to DSL line with 3 Mbps download and 512 Kbps upload speeds. On the laptop, we used a shell script to automatically load websites using Firefox 4.0[8]. The ping server was a commercial hosting system, located in Montreal, QC, Canada, acting as the remote attacker. It was scripted to send pings at precise time intervals with hping[9] and record ping traces with tcpdump[10]. We set the ping interval to 2 ms.

To emulate the attacker's training procedure, we also built a *VM-Setup*, a VMware ESX host testbed located in our lab, as shown in Figure 6(b). On this machine, we ran several VMware guest operating systems: a Ubuntu VM Client, a virtual router and a host implementing a transparent Dummynet link. The Ubuntu VM Client acted as a virtual target, and was scripted to browse websites using Firefox, similar to the real home user. The virtual router provided NAT service for the client, and was connected

[7] This number is underestimated as many of the IPs on our server log already became out-of-date (the hosts hooked went offline) by the time of ping tests. More accurate number can be obtained if a list of valid DSL IPs is provided in real time.

[8] `http://www.mozilla.com/firefox/`

[9] `http://www.hping.org`

[10] `http://www.tcpdump.org`

to the Internet through the Dummynet link. The Dummynet bridge was configured to replicate the network conditions of the target DSL link (i.e., the bandwidths). As in the DSL-Setup, we sent probes from another host outside the constrained Dummynet link to the virtual NAT router periodically. The attacker then collected training fingerprints while the virtual client was browsing websites through this virtual 'DSL' link. Note the virtual router and ping host were connected to the same dedicated high-speed LAN minimizing the impact of additional noise added by intermediate routers or network congestion caused by other hosts.

We collected fingerprints of the front pages for 1000 websites on the top list on Alexa[11]. For websites which have multiple mirrors in different countries like google.com, we only considered the site with the highest rank. We excluded websites with extremely large loading time (greater than 60 s). For each website, we collected 12 fingerprint samples from both the DSL and VM setups. The delay between collecting two samples is half an hour. Following the same assumptions in previous papers [11,17], the browsers were configured appropriately (no caching, no automatic update checks and no unnecessary plugins). This makes our results comparable with previous work.

4.3 Website Detection

We first analyze the ability of an attacker to detect whether a user visits a particular site. To do so, the attacker checks whether the distance between the user trace and the target web site is smaller than some threshold, and if so, the web site is considered detected. This is a binary classification task and its performance can be characterized by the rates of false positives—a different website incorrectly identified as the target–and false negatives–the target website not being identified. The choice of threshold ν creates a tradeoff between the two rates: a smaller threshold will decrease false positives at the expense of false negatives.

To estimate false-positive rate given a particular threshold ν, we fix a target site w and use the 12 samples $T = \{s_{w,1}, \ldots, s_{w,12}\}$ as the training set. We use the samples from the other sites as a test set; i.e., $U = \{s_{i,j}\}$ for $i \neq w, j \in \{1, \ldots, 12\}$. Given a sample $s_{i,j} \in U$, we calculate the minimum distance from it to the training samples:

$$d_w(s_{i,j}) = \min_k D(s_{i,j}, s_{w,k}) \tag{4}$$

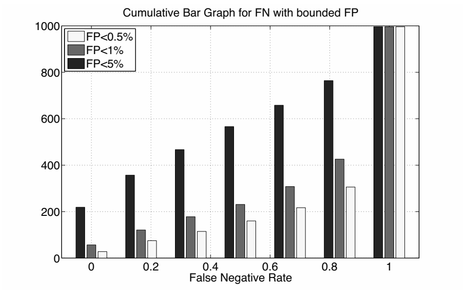
where $D(\cdot, \cdot)$ is the DTW-based distance function defined in §3.1. We then consider every sample $s_{i,j} \in U$ such that $d_w(s_{i,j}) < \nu$ to be a false positive and therefore estimate the false-positive rate:

$$\hat{p}_w = \frac{|\{s_{i,j} \in U | d_w(s_{i,j}) < \nu\}|}{|U|} \tag{5}$$

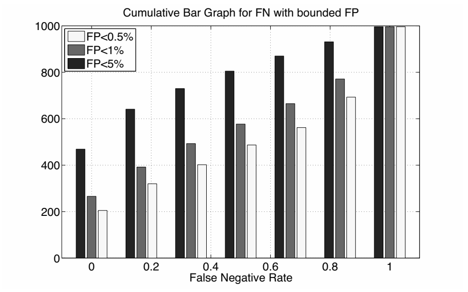
To estimate the false-negative rate, we pick one of the sample $s_{w,i}$ and calculate its minimum distance to the other 11 samples $s_{w,j}, j \neq i$, and count it as a false negative if the distance is at least ν. We then repeat this process for each $i = 1, \ldots, 12$:

$$\hat{q}_w = \frac{|\{s_{w,i} | i \in \{1, \ldots, 12\}, \min_{j \neq i} d(s_{w,i}, s_{w,j}) \geq \nu\}|}{12} \tag{6}$$

[11] `http://www.alexa.com`



(a) VM–DSL case



(b) DSL–DSL case

Fig. 7. Number of sites with a given false negative rate or smaller

Given a target false positive rate of p_w^*, we can calculate the threshold ν_w^* that would ensure $p_w < p_w^*$. Note that because $\hat{p_w}$ is only an estimate of p_w, we calculate a 95% confidence interval for p_w and chose ν such that the upper limit of the CI is below p_w^*[12]. Note that this threshold will be different for each site. We can then estimate the corresponding false negative rate $\hat{q}_w^*$ that corresponds to ν_w^*.

The target false positive rate will largely depend on the prior knowledge the attacker has. Typically, we will want to aim for a small false-positive rate, since even if Bob considers it likely that Alice does in fact visit the target site w *at some point*, most of the web browsing in any trace will still be to other sites; thus a low false-positive rate is needed for the test to have high positive predictive value. On the other hand, Bob can easily tolerate a moderate false-negative rate, since even if he only finds out about employees searching for other jobs 90%, or even 50% of the time, this information is useful nevertheless. Likewise, perfect detection is not needed for the potential attack to have a chilling effect on Alice's behavior.

Figure 7 shows the false negative rates that can be achieved given a target false-positive rate below 0.5%, 1%, and 5%. Each bar represents a cumulative number of websites, i.e., websites for which $\hat{q}_w^*$ is at or below the x-axis value. We show two sets of results; one using the VM setup for training and DSL for testing (Figure 7(a)) and one using the DSL samples for both training and test data sets (Figure 7(b)). Note that there is a significant difference between the two graphs, resulting from the discrepancies between the simulated (VM) and the test environment. We expect that, with some work, an attacker may be able to reduce such discrepancies by more carefully tuning the parameters of the virtual machine and the simulated link, or by using actual hardware and a real DSL line that mimics Alice's setup. The DSL–DSL case therefore shows the limits of what can be achieved by improving the training environment.

An important observation is that, in both cases, the success of the web detection is highly dependent on the target site. For a small number of sites—75 in the VM–DSL case and 320 in the DSL–DSL case—the web detection attack works very well: we are

[12] We use a binomial proportion confidence interval here. This is slightly imprecise, as the 12·999 samples are not independent; we leave computation of confidence intervals that take this into account for future work.

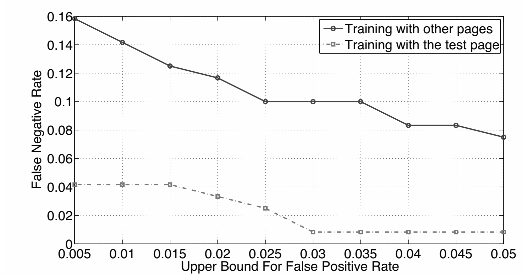


Fig. 8. Detection performance for Warrior Forum when training with the correct post page and when using other pages

able to maintain a very low false-positive rate of 0.5% while experiencing few false negatives (17% or below). On the other hand, some sites are virtually invulnerable to our attack: for 65 of the sites we tested, we were unable to observe *any* true positives with a target false-positive rate of 5% (i.e., $\hat{q}^*_w = 100\%$), even in the best-case DSL–DSL scenario. We found these sites to have either very short traces, making it difficult to distinguish them from other such sites, or highly variable traffic patterns due, for example, to dynamic content, making it difficult to create a useful fingerprint. Note the attacker can predict false-positive rates for different websites ahead of time using experiments carried on VM setups. Making use of this knowledge would make the attack even stronger.

4.4 Deanonymization

We next consider the deanonymization attack described in §3.3. As a case study, we considered the site www.warriorforum.com, a popular Internet marketing forum. It uses the vBulletin software, which was, as of August 2011, the most popular bulletin board software[13], and thus should be representative of a number of other sites. In our attack scenario, Bob wishes to find out if Alice is using a particular pseudonym (say, "diane123") to post on the site. To accomplish this, he first collects traces from Alice's home computer for a period of time. He then waits for posts to the forum from diane123 and performs a detection attack to see if Alice was visiting the site at the time of post. Repeated successful matches can then be used to obtain increasing confidence in tying Alice to diane123.Note that Bob will need to build a profile that targets internal pages of `www.warriorforum.com`, rather than the front page. Alice's post requests will be too small to create an easily-observable feature; however, vBulletin displays the forum thread after a post has been made. Therefore, Bob can collect samples visiting threads where diane123 has posted to create a fingerprint. Note that, in this attack, fingerprint creation happens *after* the trace collection. A problem facing Bob is that different post pages on Warrior Forum will have similar features in their RTT profile. A match, therefore, can show that Alice visited *some* Warrior Forum page with high confidence, but it may not have been the correct thread. Even this information, however, is likely to

[13] `http://www.big-boards.com/statistics/`

be enough for deanonymization. For example, Alexa shows that fewer than 1% of US Internet users actually visit the Warrior Forum, and those that do tend to stay on the site less than 10 minutes on average. If we make the simplifying assumptions that these visits are distributed randomly across a three-hour evening period, the additional false-positive rate due to random visits to *other* Warrior Forum pages is no more than 6%, even if Alice is *known* to be a Warrior Forum user. Combining observations across several posts allows Bob to improve his confidence.

Over a long term, even simpler attacks may suffice: since most people's Internet usage is bursty, simply observing that Alice always actively used the Internet in some way whenever diane123 made posts can be used for deanonymization [7]. Likewise, Bob may be able to rule out Alice as a suspect if she was known to be at home (due to recent DSL activity) but her connection was idle at the times of the target posts.

Finally, Bob may be able to use the similarity between internal forum pages to his advantage. In particular, suppose that Alice publicly participates in the forum under her real identity, in addition to potentially posting under a pseudonym. Bob can use the times of Alice's posts under her real name to label traces collected from Alice's computer and create a training set. In this case, Bob does not need to simulate Alice's computing environment as the training and test environments are exactly the same—the ideal conditions we used in the DSL–DSL case. To study this attack, we collected samples from 100 different posts on the Warrior Forum site. For each sample, we attempt to match it to a fingerprint created from the other 99 posts; our process is similar to (6), except using a different threshold for each sample. From this, we estimate the false negative rate for a given target false-positive rate, calculated using (5), using the traces from 999 other websites. Figure 8 shows the results. The use of different pages to test degrades the matching performance, but it still provides sufficient detection power for deanonymization after a few posts.

5 Discussion

In this section, we discuss more on the feasibility and limitations of our work.

1. *ICMP support.* The attack scenario of our scheme relies on ICMP probe packets, hence we care about whether ICMP is enabled in real routers. In testing over 918 probable DSL hosts on the Internet in Section §4.1, we found over 25% responded to ping requests. Since we harvested these probable DSL hosts from the Internet over a period of several months, it is not clear how many that failed to respond were simply down rather than blocking our probes. Thus, we can assume that the fraction of hosts that respond to ping is even larger. Additionally, in a brief survey of consumer-grade router hardware, we found that many of them do not perform ICMP filtering, at least not in the default configuration. Moreover, even though the ping packets are blocked by firewalls on some home routers, other forms of probes may be exploited as well; for example, if the home router exposes TCP ports for file sharing or other applications, SYN packets can be used as probes with the same effectiveness.
2. *FIFO scheduling policy.* The high correlation between the user's traffic pattern and the attacker's ping RTTs comes from the fact that the router serves packets in FIFO

order. Note that most home routers today do not use QoS extensions and schedule packets on a given link in FIFO order. Thus, information leaked by these routers can be exploited with remote traffic analysis. Certainly, a fair queuing implementation [26] would reduce the impact that cross-traffic would have on the probe sequence and hence reduce the effectiveness of the side channel, but not entirely eliminate it [15].

3. *Limited Last-hop Bandwidth.* The information leaked through our side channel are the states of the queue length in the router's buffer. Hence, to have nontrivial queues built up in the buffer, the broadband link must have limited bandwidth compared to the rest of the links in the path. In our experiments, we have used speeds typical of current home broadband speeds—several Mbps, and our scheme worked well in those environments. The deployment of faster links, such as Fiber-to-the-Home (FTTH), may reduce the effectiveness of the queueing side channel, but notice that if the core network is similarly upgraded in speed, the bandwidth disparity necessary for our attack will remain.
4. *Victim's IP address.* In our attack, the attacker needs to know the user's IP address to send the probes. Although this mapping is typically only explicitly known to ISPs, many protocols, such as file sharing, instant messaging, VoIP, and email, will reveal the IP address of a user. Other forms of IP address reconnaissance may also be possible but are outside the scope of this work.
5. *Multiple users.* In the scenario of our remote traffic analysis, the attacker's probes cannot distinguish between the traffic of multiple users on the same link, so shared broadband connections present an obstacle to our attack. However, even in multi-user installations, it is still common for only one user to be using the Internet at any given point during the day. Some previous work on traffic analysis has used blind source separation to separate traffic from multiple users [35]; similar techniques may be applicable here. For example, in Figure 2, traffic follows a periodic pattern based on the RTT between Alice and the website; such periodicity might help separate the sources.
6. *Dynamic nature of websites.* Our attack relies on web sites having relatively stable fingerprints. Although the overall pattern captured by our RTT probes remains static enough within days, the website content may incur significant changes (e.g., site redesigns) over time; which in turn will result in a change of its fingerprint. Thus, for best results, the training set should be updated continuously. This limitation applies to any website fingerprinting approach even local website fingerprinting techniques which benefit from better vantage points [11, 17].
7. *Content distribution networks.* Websites that use content delivery networks (CDNs) will use different servers to deliver content based on the user's location. They may present localized versions of the site to users in different countries or regions. As shown in our experimental results, this can cause fingerprints to differ significantly. If identifying these sites is a high priority for the attacker, additional work would be needed to obtain fingerprints of the right version by, for example, using proxies and other techniques to fool IP-based localization.
8. *Cache issues.* In our tests, we followed the assumption in previous work [11, 17] and disabled the cache in the browser. This implies that our results demonstrates

the attacker's ability to verify that a user visits a web page for the first time. To investigate cases with cache enabled, one possible solution would be build separate fingerprints based on the time since the site was first downloaded, e.g., after 1 hour, 6 hours, 1 day, 1 week, to minimize the effect that caching would have on the attack. Note that with continuous observation of a computer, the attacker may be able to guess how long ago the last visit was.

6 Related Work

The use of network probes to infer information about traffic at a remote location has been explored in previous work in the context of anonymous communication networks. Murdoch and Danezis used a remote traffic analysis approach to expose the identity of relays participating in a circuit in the Tor [8] and MorphMix [21] anonymous communication networks [19]. Their approach was to send an on–off pattern of high-volume traffic through the anonymous tunnel and a low-volume probe to a router under test. If the waiting times of the probe showed a corresponding increase during the "on" periods, the router was assumed to be routing the flow. However, when Murdoch and Danezis evaluated their attack, the Tor network was lightly loaded and consisted only of 13 relays; to repeat their attack on today's network, with around 2 000 relays and high traffic load[14], an attacker would needs extremely large amounts of bandwidth to measure enough relays during the attack window. Evans et al. [9] strengthened Murdoch and Danezis's attack of by a bandwidth amplification attack which make their attack feasible in modern-day deployment of Tor. Hopper et al. [13, 14] use a combination of Murdoch and Danezis's approach and pairwise round trip times (RTTs) between Internet nodes to correlate Tor nodes to likely clients. Chakravarty et al. [3] propose an attack for exposing Tor relays participating in a circuit of interest by modulating the bandwidth of an anonymous connection and then using available bandwidth estimation to observe this pattern as it propagates through the Tor network. Note that these techniques relied on detecting a specially-crafted coarse-grained communication pattern, whereas our attack makes use of fine-grained information obtained through remote traffic analysis.

We also survey previous work on recovering information about encrypted HTTP traffic.The fact that object sizes could be used to infer sensitive information, even after encryption, was first mentioned by Yee (as related by Wagner and Schneier [30]). A specific concern listed by Yee is that the particular page within a site accessed by the user could be revealed by considering URL and object lengths. Chen et al. [4] applied this observation to AJAX applications to recover detailed information about the internal state of the application and users' data. Cheng et al. [5] present the earliest implementation of website fingerprinting. The classification features used in their scheme are the object sizes and the HTML file sizes. Hintz [12] and Sun et al. [29] both consider website fingerprinting attacks in SSL-encrypted HTTP connections. Their classification features are object sizes and counts. While Hintz did not present implementation details and experimental results, Sun et al. use a Jaccard's coefficient based classifier and show that their attack can achieve a correct identification rate of 75%. Instead of looking at web objects, Bissias et al. [2], Liberatore et al. [17], and Herrmann et al. [11] study

[14] See `http://torstatus.blutmagie.de` (retrieved November 2010)

the statistical characteristics of individual packets in the traffic flows. Bissias et al. use packet sizes and inter-arrival timings as classification features. Their method is fragile to the changes in the network environment, as the inter-arrival timing is highly dependent on the specific routing path and varies from time to time. To address this problem, Liberatore et al. only use packet sizes and counts in classification. They implement both Jaccard coefficient and Naïve Bayes classifier, and show the efficacy of the attack in practice. Using similar scheme, Herrmann et al. further improve the classification accuracy using Multinomial Naïve Bayes classifier.

7 Conclusion

We show that traffic analysis attacks can be carried out remotely, without access to the analyzed traffic, thus greatly increasing the attack surface and lowering the barrier to entry for conducting the attack. We identify a queuing side channel that can be used to infer the queue size of a given link with good accuracy and thus monitor traffic patterns. We show how this channel can be used to carry out a remote attack to detect a remote user's browsing patterns. This highlights the importance of traffic analysis attacks in today's connected Internet.

References

1. Akella, A., Seshan, S., Shaikh, A.: An empirical evaluation of wide-area Internet bottlenecks. In: Crovella, M. (ed.) 3rd ACM SIGCOMM Conference on Internet Measurement, pp. 101–114. ACM, New York (2003), `http://dl.acm.org/citation.cfm?id=948205.948219`
2. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy Vulnerabilities in Encrypted HTTP Streams. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 1–11. Springer, Heidelberg (2006)
3. Chakravarty, S., Stavrou, A., Keromytis, A.D.: Identifying proxy nodes in a Tor anonymization circuit. In: Dipanda, A., Chbeir, R., Yetongnon, K. (eds.) IEEE International Conference on Signal Image Technology and Internet Based Systems, pp. 633–639. IEEE Computer Society, Los Alamitos (2008)
4. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In: Evans, D., Vigna, G. (eds.) IEEE Symposium on Security and Privacy, pp. 191–206. IEEE Computer Society (May 2010), `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5504714`
5. Cheng, H., Avnur, R.: Traffic Analysis of SSL Encrypted Web Browsing (1998), `http://www.cs.berkeley.edu/ daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps`
6. Coull, S.E., Collins, M.P., Wright, C.V., Monrose, F., Reiter, M.K.: On web browsing privacy in anonymized netflows. In: Provos, N. (ed.) 16th USENIX Security Symposium. USENIX Association, Berkeley (2007), `http://www.usenix.org/events/sec07/tech/coull.html`
7. Danezis, G., Serjantov, A.: Statistical Disclosure or Intersection Attacks on Anonymity Systems. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 293–308. Springer, Heidelberg (2004), `http://www.springerlink.com/index/TQLJB3HYBK4RUBLA.pdf`

8. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Blaze, M. (ed.) USENIX Security Symposium, pp. 303–320. USENIX Association, San Diego (2004), http://portal.acm.org/citation.cfm?id=1251396
9. Evans, N.S., Dingledine, R., Grothoff, C.: A practical congestion attack on Tor using long paths. In: Monrose, F. (ed.) 18th USENIX Security Symposium, pp. 33–50. USENIX Association (August 2009), http://www.usenix.org/events/sec09/tech/full_papers/evans.pdf
10. Franklin, J., Paxson, V., Perrig, A., Savage, S.: An inquiry into the nature and causes of the wealth of Internet miscreants. In: De Capitani di Vemarcati, S., Syverson, P. (eds.) 14th ACM Conference on Computer and Communications Security, pp. 375–388. ACM, New York (2007), http://dl.acm.org/citation.cfm?id=1315245.1315292
11. Herrmann, D., Wendolsky, R., Federrath, H.: Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-Bayes classifier. In: ACM Workshop on Cloud Computing Security, pp. 31–42. ACM, Chicago (2009), http://portal.acm.org/citation.cfm?id=1655013
12. Hintz, A.: Fingerprinting Websites Using Traffic Analysis. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 171–178. Springer, Heidelberg (2003), http://www.springerlink.com/index/C4QWE6D608P2CJYV.pdf
13. Hopper, N., Vasserman, E.Y., Chan-Tin, E.: How much anonymity does network latency leak? In: De Capitani di Vimercati, S., Syverson, P. (eds.) 14th ACM Conference on Computer and Communications Security, pp. 82–91. ACM, New York (2007), http://dl.acm.org/citation.cfm?id=1315245.1315257
14. Hopper, N., Vasserman, E., Chan-Tin, E.: How much anonymity does network latency leak? ACM Transactions on Information and System Security 13(2) (2010), http://portal.acm.org/citation.cfm?id=1698753
15. Kadloor, S., Gong, X., Kiyavash, N., Tezcan, T., Borisov, N.: Low-Cost Side Channel Remote Traffic Analysis Attack in Packet Networks. In: Xiao, C., Olivier, J.C. (eds.) 2010 IEEE International Conference on Communications. IEEE (May 2010), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5501972
16. Lakshminarayanan, K., Padmanabhan, V.N.: Some findings on the network performance of broadband hosts. In: Crovella, M. (ed.) Proceedings of the 2003 ACM SIGCOMM Conference on Internet Measurement, IMC 2003, pp. 101–114. ACM Press, New York (2003), http://portal.acm.org/citation.cfm?doid=948205.948212
17. Liberatore, M., Levine, B.N.: Inferring the source of encrypted HTTP connections. In: Wright, R., De Capitani di Vemarcati, S. (eds.) 13th ACM Conference on Computer and Communications Security, pp. 255–263. ACM, New York (2006), http://portal.acm.org/citation.cfm?id=1180437
18. Lyon, G.F.: Nmap Network Scanning. Nmap Project (1999)
19. Murdoch, S., Danezis, G.: Low-Cost Traffic Analysis of Tor. In: Paxson, V., Waidner, M. (eds.) 2005 IEEE Symposium on Security and Privacy, pp. 183–195. IEEE Computer Society, Berkeley (2005), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1425067
20. Prasad, R., Davrolis, C., Murray, M., Claffy, K.: Bandwidth estimation: metrics, measurement techniques, and tools. IEEE Network 17(6), 27–35 (2003), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1248658

21. Rennhard, M., Plattner, B.: Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection. In: Samarati, P. (ed.) ACM Workshop on Privacy in Electronic Society, pp. 91–102. ACM Press, New York (2002), http://portal.acm.org/citation.cfm?id=644537
22. Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J., Cottrell, L.: pathchirp: Efficient available bandwidth estimation for network paths. In: Passive and Active Measurement Workshop, vol. 4. Citeseer (March 2003)
23. Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols. ACM SIGCOMM Computer Communication Review 27(1), 31–41 (1997), http://portal.acm.org/citation.cfm?doid=251007.251012
24. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing 26(1), 43–49 (1978), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1163055
25. Saponas, T.S., Lester, J., Hartung, C., Agarwal, S., Kohno, T.: Devices that tell on you: Privacy trends in consumer ubiquitous computing. In: Provos, N. (ed.) 16th USENIX Security Symposium, pp. 55–70. USENIX Association (2007), http://portal.acm.org/citation.cfm?id=1362908
26. Shreedhar, M., Varghese, G.: Efficient fair queuing using deficit round-robin. IEEE/ACM Transactions on Networking 4(3), 375–385 (1996), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=502236
27. Song, D.X., Wagner, D., Tian, X.: Timing analysis of keystrokes and SSH timing attacks. In: Wallach, D.S. (ed.) 10th USENIX Security Symposium. USENIX Association (August 2001), http://www.usenix.org/events/sec01/song.html
28. Strauss, J., Katabi, D., Kaashoek, F.: A measurement study of available bandwidth estimation tools. In: Crovella, M. (ed.) 3rd ACM SIGCOMM Conference on Internet Measurement, pp. 39–44. ACM, New York (2003), http://portal.acm.org/citation.cfm?id=948211
29. Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, V.N., Qiu, L.: Statistical identification of encrypted Web browsing traffic. In: Abadi, M., Bellovin, S.M. (eds.) IEEE Symposium on Security and Privacy, pp. 19–30. IEEE Computer Society (May 2002), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1004359
30. Wagner, D., Schneier, B.: Analysis of the SSL 3.0 Protocol. In: Tygar, D. (ed.) USENIX Workshop on Electronic Commerce. USENIX Association (November 1996), http://www.usenix.org/publications/library/proceedings/ec96/wagner.html
31. White, A.M., Matthews, A.R., Snow, K.Z., Monrose, F.: Phonotactic reconstruction of encrypted VoIP conversations: Hookt on Foniks. In: Vigna, G., Jha, S. (eds.) IEEE Symposium on Security and Privacy, pp. 3–18. IEEE Computer Society (May 2011), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5958018
32. Wright, C.V., Ballard, L., Coull, S.E., Monrose, F., Masson, G.M.: Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In: IEEE Symposium on Security and Privacy, pp. 35–49. IEEE Computer Society, Washington, DC (2008), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4531143

33. Wright, C.V., Ballard, L., Coull, S.E., Monrose, F., Masson, G.M.: Uncovering Spoken Phrases in Encrypted Voice over IP Conversations. ACM Transactions on Information and System Security 13(4), 1–30 (2010), http://doi.acm.org/10.1145/1880022.1880029
34. Zhang, K., Wang, X.: Peeping Tom in the neighborhood: Keystroke eavesdropping on multi-user systems. In: Monrose, F. (ed.) 18th USENIX Security Symposium USENIX Security. USENIX Association (August 2009), http://www.usenix.org/events/sec09/tech/full_papers/zhang.pdf
35. Zhu, Y., Bettati, R.: Unmixing Mix Traffic. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 110–127. Springer, Heidelberg (2006)