On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records

Sambuddho Chakravarty,* Marco V. Barbera,† Georgios Portokalidis,‡
Michalis Polychronakis,* Angelos D. Keromytis*

*Columbia University, New York, NY, USA
{sc2516,mikepo,angelos}@cs.columbia.edu

†Sapienza Universita Di Roma, Rome, Italy
barbera@di.uniroma1.it

‡Stevens Institute of Technology, Hoboken, NJ, USA
gportoka@stevens.edu

***Abstract*—Low-latency anonymous communication networks, such as Tor, are geared towards web browsing, instant messaging, and other semi-interactive applications. To achieve acceptable quality of service, these systems attempt to preserve packet interarrival characteristics, such as inter-packet delay. Consequently, a powerful adversary can mount traffic analysis attacks by observing similar traffic patterns at various points of the network, linking together otherwise unrelated network connections. Previous research has shown that having access to a few Internet exchange points is enough for monitoring a significant percentage of the network paths from Tor nodes to destination servers. Although the capacity of current networks makes packet-level monitoring at such a scale quite challenging, adversaries could potentially use less accurate but readily available traffic monitoring functionality, such as Cisco's NetFlow, to mount large-scale traffic analysis attacks.**

In this paper, we assess the feasibility and effectiveness of practical traffic analysis attacks against the Tor network using NetFlow data. We present an active traffic analysis method based on deliberately perturbing the characteristics of user traffic at the server side, and observing a similar perturbation at the client side through statistical correlation. We evaluate the accuracy of our method using both in-lab testing, as well as data gathered from a public Tor relay serving hundreds of users. Our method revealed the actual sources of anonymous traffic with 100% accuracy for the in-lab tests, and achieved an overall accuracy of about 81.4% for the real-world experiments, with an average false positive rate of 6.4%.

I. Introduction

Anonymous communication networks hide the actual source (or destination) address of Internet traffic, preventing the server (or client) and other entities along the network from determining the actual identities of the communicating parties. There are two broad types of anonymity-preserving systems: *low latency* and *high latency*. Low-latency systems are designed primarily for semi-interactive applications, such as web browsing and instant messaging. Among others [1], [2], Tor [3] is probably the most widely used proxy-based low-latency anonymous communication network. In Tor, clients establish *circuits* through a chosen set of proxies, beginning with an *entry node* and reaching the final destination through an *exit node*.

To offer acceptable quality of service, a distinctive characteristic of these systems is that they attempt to maintain packet inter-arrival times. Unfortunately, this makes them vulnerable to traffic analysis attacks [4]–[12], whereby an adversary with access to traffic entering and leaving entry and exit nodes can correlate seemingly unrelated traffic flows and reveal the actual communicating endpoints. High-latency systems, on the other hand, are designed for delay-tolerant applications such as e-mail, and introduce artificial delay to the forwarded packets so as to defend against traffic analysis attacks.

As Tor nodes are scattered around the globe, and the nodes of circuits are selected at random, mounting a traffic analysis attack in practice would require a powerful adversary with the ability to monitor traffic at a multitude of autonomous systems (AS). Murdoch and Zieliński, however, showed that monitoring traffic at a few major Internet exchange (IX) points could enable traffic analysis attacks to a significant part of the Tor network [13]. Furthermore, Feamster et al. [14] and later Edman et al. [15] showed that even a single AS may observe a large fraction of entry and exit node traffic—a single AS could monitor over 39% of randomly generated Tor circuits.

Packet-level traffic monitoring at such a scale would require the installation of passive monitoring sensors capable of processing tens or hundreds of Gbit/s traffic. Although not impossible, setting up a passive monitoring infrastructure of this scale is a challenging endeavor in terms of cost, logistics, and effort. An alternative, more attractive option for adversaries would be to use the readily available (albeit less accurate) traffic monitoring functionality built into the routers of major IXs and ASs, such as Cisco's NetFlow. Compared to packet-level traffic monitoring, the flow-level aggregation of the measured traffic properties and the use of aggressive sampling make NetFlow data less than ideal for traffic analysis attacks. Murdoch and Zieliński showed through simulation that traffic analysis using sampled Netflow data is possible, provided there are adequate samples. Still, there have been no prior efforts to explore the various practical aspects of mounting traffic analysis attacks using NetFlow data.

As a step towards filling this gap, in this paper we study the feasibility and effectiveness of traffic analysis attacks using NetFlow data, and present a practical active traffic analysis attack against Tor. Our approach is based on identifying pattern similarities in the traffic flows entering and leaving the Tor network using statistical correlation. To alleviate the uncertainty due to the coarse-grained nature of NetFlow data, our attack relies on a server under the control of the adversary that introduces deterministic perturbations to the traffic of

anonymous visitors. Such a colluding adversary is similar to the one described in [16]. Among all entry-node-to-client flows, the actual victim flow can be distinguished due to its high correlation with the respective server-to-exit-node, as both carry the induced traffic perturbation pattern.

We assume a powerful adversary, capable enough of observing traffic entering and leaving the Tor network nodes at various points. Such an adversary might be a powerful nations state or a group of colluding nation states that can collaborate to monitor network entering and leaving various ASes simultaneously. For the sake of our experiments, and for gathering flow information for traffic going to the entry node, we hosted a public Tor entry node within our institution.

Alternately, it is not even essential to be a global adversary to launch such traffic analysis attacks. A powerful, yet non-global adversary could use traffic analysis methods such as [6], [8], [17], [18] to determine the various relays participating in a Tor circuit and directly monitor the traffic entering the entry node of the victim connection.

We evaluated the effectiveness of our traffic analysis attack first in an in-lab environment, and later using a set-up involving real Tor network relays. For our research, we used a combination of flow data gathered from open source tools such as `ipt_netflow` [19] and `flowtools` [20], as well as the flow records from our institutional Cisco router. In our controlled in-lab experiments, we relied solely on data from open source tools, while in the experiments involving our public Tor relay, used data both from open source tools as well as from our institutional edge router. In the controlled in-lab environment we had 100% success rate in determining the source of anonymous flows. When evaluating our attack with traffic going through the public Tor relay, we were able to detect the source in 81.4% cases. We observed about 12.2% false negatives and 6.4% false positives in our measurements. Correlation is a sensitive metric that varies considerably due to small variation in input values. We thus couple correlation with heuristics to filter out flows which are very unlikely to be the victim. The false negatives were a function of the filtering threshold and low correlation co-efficient (< 0.2), while the false positives were primarily due to our approximation based to method to make up for lack of input samples which led to loss of information and subsequent inaccurate correlation co-efficient.

Briefly, the main contributions of this research are as follows:

- An active practical traffic analysis attack that relies on performing statical correlation on Netflow data to reveal source of anonymous traffic.
- An empirical evaluation of the feasibility and accuracy of such traffic analysis attack first in an in-lab set-up and later in a set-up involving a public Tor relay.
- A method to perform our correlation based traffic analysis even in the presence of sparse flow data, which can help accurately reveal the source of anonymous traffic in 80% of the cases (with 14% false negatives and 6% false positives).

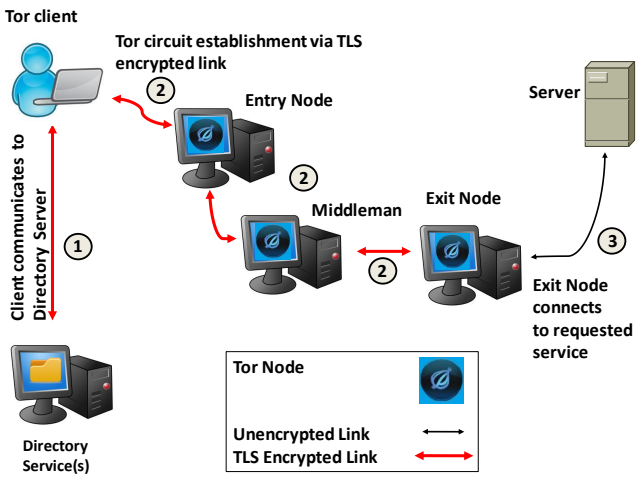


Fig. 1. Basic steps for communicating through Tor. The client obtains a list of the available Tor relays from a directory service ①, establishes a circuit using multiple Tor nodes ②, and then starts forwarding its traffic through the newly created circuit ③.

II. Background and Related Work

A. The Tor Anonymity Network

Tor [3] is the most widely used low-latency anonymous communication network, with over half a million users as of July 2013 [21]. Tor safeguards the anonymity of Internet users by relaying user-generated TCP streams through a network of overlay nodes run by volunteers. It can be used for both *initiator* and *responder* anonymity. That is, it can hide the identity (IP address) of the initiator of a connection from the destination and vice-versa through the use of *hidden services*.

The Tor overlay network consists of volunteer-run proxies distributed across the globe, known as *Onion Routers (ORs)*. User traffic is relayed through *circuits*, which are formed using semi-persistent SSL connections between different Tor nodes. Commonly, a Tor circuit consists of three nodes: the first where the client connects is the *entry node*, the next is the *middleman*, and the last that connects to the destination is the *exit node*.

Figure 1 briefly presents the basic steps for the creation of a new Tor circuit consisting of three onion routers. To establish a circuit, a client first consults Tor's service directory, which is know beforehand, to fetch a list of Tor relays. It then selects all the nodes that are going to compose the circuit using the Tor relay selection algorithm [22]. Next, the client negotiates shared secret keys with the circuit's relays. Messages exchanged over the circuit are encapsulated into fixed-size (512-byte) packets, called *cells*, and encrypted in multiple layers, using the negotiated keys. Encryption starts with key established with the exit node and ends with the key of the entry node. The process is reversed as the cell traverses the circuit, with every node "peeling off" one layer of encryption and forwarding the cell to the next node, until it reaches the exit node that establishes a TCP connection with the destination and transmits the client's original data.

B. Network Flow Monitoring

NetFlow is a network protocol designed for collecting and monitoring network traffic. NetFlow groups exchanged data in *network flows*, which can correspond to TCP connections or other IP packets sharing common characteristics, such UDP packets sharing source and destination IP addresses, port numbers, and other information (see protocol specification [23] for further details).

Sample flow records are listed in Table I. The columns labeled `Start` and `End` denote flow start and end times. `Sif` and `Dif` represent the SNMP source and destination interface `ifIndex` respectively. `SrcIPaddress` and `DstIPaddress` denote source and destination IP addresses, and, similarly, `SrcP` and `DstP` denote port numbers. `P` is the protocol (6 represents TCP and 17 represents UDP). `Pkts` and `Octets` represent the number of packets, and bytes respectively, observed for this flow in the particular time frame. `Fl` denotes the TCP flags present in the TCP header.

Most Cisco routers these days include networking monitor capabilities using NetFlow. Other major router and networking device manufacturers have their own, similar protocols, like Juniper *jflow* [24], Huawei *Netstream* [25], and Alcatel Lucent *sflow* [26]. There are also various open source implementations for collecting network statistics using NetFlow, such as `ipt_netflow` [19] and `fprobe` [27].

The way these systems work is by creating a NetFlow record in memory and updating it as more packets are forwarded through the router. Each flow is associated with two timers, namely the *active* and *inactive* timers. These are specified in seconds, within the router configuration. If data has been recorded for a flow within the recent active timeout period, then it classified as an active flow. Records of active flow is aged out to persistent storage after every expiration of the active timeout and fresh record entries are instantiated in the flow cache. On the other hand, a flow that hasn't recorded activity within the recent inactive timeout period, is labeled as an inactive flow. Following the expiration of the inactive timeout, the flow record is moved to persistent storage and new records, corresponding to the flow, are instantiated. The records are flushed-out to persistent storage, so they can be processed to produce higher-level statistics, generate traffic reports, and so on, based on their age, or when a flow is terminated. For instance, when a TCP FIN packet has been observed for a TCP connection. Administrators can modify various parameters [28] to customize the process, like setting timeouts for controlling when to push active and inactive flow records to persistent storage, and enabling packet sampling. As Internet speeds have grown, vendors introduced packet sampling to minimize the overhead of network monitoring in packet forwarding, in exchange for accuracy in the reported flow records.

C. Traffic Analysis

Traffic analysis attacks have been extensively studied over the past decade [5], [43]. Murdoch and Danezis [6] developed the first practical traffic analysis attack against Tor. They proposed a technique to determine the Tor relays involved in a circuit. The method involved a corrupt server, accessed by the victim client, and corrupt Tor node that can form one-hop circuits with arbitrary legitimate nodes. The server modulates the data being sent back to the client, while the corrupt Tor node is used to measure delay between itself and Tor nodes. The correlation between the perturbations in the traffic exchanged with a Tor node, and the server stream helped identify the relays involved in a particular circuit.

Hopper et al. [7] used this method, along with one-way circuit latency, and the Vivaldi network coordinate system to determine the possible source of anonymous traffic. However, in 2009, it was demonstrated by Evans et. al [8] that the traffic analysis attack proposed by Murdoch and Danezis was more applicable due to the large number of Tor relays, the large volume of Tor traffic, low end-to-end quality of service and possible network bottleneck locations between the adversaries vantage point and the victim relays. They proposed a method to amplify the network traffic by using circuits that repeatedly used the same relays and aided in easier identification of the relays.

Later, we proposed methods for performing traffic analysis using remote network bandwidth estimation tools, to identify the Tor relays and routers involved in Tor circuits [16], [18]. Our method assumed that the adversary is in a position to perturb the victim traffic by colluding with the server and is in control of various network vantage points, from where he can observe variations in network bandwidth. Due to the network traffic perturbations induced by the server, bandwidth monitoring could reveal the relays part of the circuit being attack, eventually leading to the source. Our efforts achieved modest success in confirming the identity of the source of anonymous traffic. On average, about 42% of the connections between victim Tor clients and entry nodes preserved the traffic fluctuations induced by the server.

More recently, Mittal *et. al.* [17], demonstrated a somewhat modified version of the Murdoch and Danezis attack, which rather than using variation in one-way delay, relied on bandwidth variations to determine if two clients were using the same set of Tor relays for their circuits.

In general, traffic analysis attacks belong to a general class of side-channel attacks that try to correlate changes in anonymous communication patterns to changes in various network and system parameters such as network traffic throughput, one-way or round trip latency, temperature variation of network equipment and various other parameters which can indirectly reveal variation in network and system operation parameters. These correlated patterns are used to link network peers which are part of an anonymous communication session. For example, previous work [9], [30] demonstrated how variation in anonymous traffic leads to variation in CPU temperature and, hence, the system clock. TCP clock skew, which can be remotely determined through TCP timestamp options, can easily reveal this system clock drift.

Traffic analysis (and related) side-channel attacks have not only been studied in the context of anonymity networks but in the wider context of various other kinds of secure communication systems such as revealing contents of encrypted traffic [31] and finding recently visited websites [32]. Further Wang *et. al.* [33], [34] show how traffic analysis can been used to reveal anonymously communicating VoIP peers.

Traffic Analysis by Internet Exchange Level Adversaries

Start	End	Sif	SrcIPaddress	SrcP	DIf	DstIPaddress	DstP	P	Fl	Pkts	Octets
0606.23:59:06.616	0606.23:59:36.660	65535	192.168.0.20	50000	2	213.163.65.50	54089	6	0	3	156
0606.23:59:06.616	0606.23:59:36.660	2	213.163.65.50	54089	65535	192.168.0.20	50000	6	0	3	1914
0606.23:59:29.420	0606.23:59:30.572	2	71.58.107.145	42259	65535	192.168.0.20	50000	6	2	10	3961
0606.23:59:29.420	0606.23:59:30.612	65535	192.168.0.20	50000	2	71.58.107.145	42259	6	2	9	3578
0606.23:59:33.396	0606.23:59:33.396	65535	127.0.0.1	55171	1	127.0.0.1	10002	17	0	1	1492

TABLE I. SAMPLE NETFLOW RECORDS SHOWING VARIOUS FIELDS

In 2007, Murdoch et al. [13] proposed using NetFlow data from routers in Internet Exchanges to perform traffic analysis attacks against traffic entering and leaving the Tor network. They presented two novel contributions. First, the showed that there is a small number of Internet Exchanges which can potentially monitor large fraction of Tor traffic. Second, these Internet Exchanges could use network sub-systems built into existing network infrastructure, such as NetFlow found in Cisco routers, to launch traffic analysis attacks. They proposed a model of the traffic and the attack to which they input the NetFlow traffic gathered from monitoring a Tor relay for a short duration and described, through simulations, how varying the number of flows, the data transmission rate, and end-to-end delays may affect the accuracy in determining the source of the anonymous traffic.

However, their efforts did not explore the feasibility and effectiveness of using a sub-system like NetFlow to determine the source of anonymous traffic. Is using NetFlow practical? Can an adversary accurately de-anonymize a Tor client using solely NetFlow data? Our work attempts to answer these questions. We present a methodology for performing traffic analysis using NetFlow data alone, and experimentally evaluate its accuracy in identifying the source of anonymous traffic. We developed an active attack that involves modulating victim traffic at a colluding server and observing its effects in the network using NetFlow. We rely on statistical correlation (Pearson's correlation coefficient) to identify the closest matching flow. We experimentally evaluated the effectiveness of our method, discuss how certain limitations of the NetFlow system can affect detection accuracy, and try to compensate for the same.

III. APPROACH

A. Threat Model

In this work, we mainly focus on the problem of evaluating the effectiveness of using NetFlow data to perform practical traffic analysis attacks for identifying the source of anonymous communication. In our attack model, we assume that the victim is lured to access a particular server through Tor, while the adversary collects NetFlow data corresponding to the traffic between the exit node and the server, as well as between Tor clients and the victim's entry node. The adversary has control of the particular server (and potentially many others, which victims may visit), and thus knows which exit node the victim traffic originates from.

We assume a powerful-enough adversary that can monitor traffic at various network locations, allowing the inspection of Tor traffic towards a significant number of entry nodes [13]–[15]. The adversary could identify important ASes and their topological relationships using methods similar to those presented by Schuchard et al. [35]. Alternatively, the adversary could use throughput fingerprinting to identify the entry node of a particular victim circuit [8], [17], [18]. The challenge for the adversary is to determine the real identity of the anonymous client that corresponds to a connection seen at the server, using solely NetFlow data from the vantage points of i) the exit node towards the server, and ii) the various clients towards entry nodes. *Having determined the identity of the entry node involved in the victim anonymous connection, the adversary needs to only find the entry to client flow that uses this entry node and which correlates closely to the server to exit flow.*

B. Attack Approach

The main objective of our attack is to determine the source of anonymous connection arriving to a server using NetFlow data, available easily available from network routers. We present an active attack strategy. In our strategy the adversary uses statistical correlation over the server to exit and entry to client traffic to find similar traffic patterns. However, in traffic analysis attacks against Tor, that involves correlation of the server to exit traffic with hundreds (or even thousands) of other entry to client traffic flows, it could potentially be difficult to correctly find the flow carrying our victim traffic. Large fraction of Tor traffic is mostly due to web browsing [36]. The human effort of web browsing results in a semi-continuous network process, due to momentary pauses between visiting one web-page and then moving on to the next one. This often generates to inadequate sustained network traffic for long durations. We thus assume that victim downloads a large file from the server, for the duration of the attack, so as to generate sustained traffic for a considerably long duration of time (atleast 5 – 7 minutes). In practice there are several ways to enforce the victim to download a file from the server. An adversary could force a web-server to send out contents such a web page containing iframes containing pages with multimedia content or flash video which would downloaded when the victim access the page. Other possible ways to achieve this is by colluding with popular web services to host decoy pirated multimedia files such as pirated copies of latest movies. A victim might be tempted to download such files. Moreover, our intuition here is that, injecting a specific fingerprinting pattern, as done in [16]–[18], makes identification of the victim traffic easier by causing it to be prominent compared to contending traffic flows. In our attack model, the victim client downloads a file from the server, that colludes with the adversary and is in a position to inject a traffic pattern by perturbing the TCP connection it sees arising from an exit node. Thereafter, the adversary uses NetFlow data to obtain traffic statistics from the server to exit node and correlates it individually to each of the entry node to client traffic statistics so as to find the flow which carries the injected fingerprint (the one that is likely to show high correlation to the server to exit traffic statistics).

In our attack model, the adversary access NetFlow data, corresponding to the server to exit and entry node to client

traffic, and correlates network statistics to find similar patterns, thereby linking otherwise unrelated network connections. The client connects to the server, that colludes with the adversary and downloads a relatively large file, which takes a while to complete. This generates sustained traffic for a relatively large duration and thus records adequate flow samples to compute the correlation coefficient[1]. In our experiments the server hosted a file of 100 Mbytes so that the victim client (assuming the client achieves a maximum throughput of 1 Mbit/s, which is relatively on the higher side for Tor standards), takes over an hour to download it completely. This was long enough for the server to perturb the traffic several times and record enough samples to perform statistical correlation. However, as described ahead in Section IV, even the longest experiment were much shorter than an hour.

While the client downloads the file from the server, the server (in collusion with the adversary) injects a repeating traffic pattern in the TCP connection, that it observes as originating from the exit node. In our experiments, this was achieved by capping the connection throughput to a particular bandwidth value for several seconds and then switching to another one. We achieved this using Linux Traffic Controller [37], a Linux kernel based traffic shaping and conditioning framework. In our experiments, the adversary injected two different kinds of traffic patterns. The first one involves the server injecting a simple "square wave" like pattern, achieved by repeatedly switching the victim's traffic pattern between two bandwidth values. The other was a more complex "step" like pattern, that was achieved by the server switching repeatedly between several pre-calculated bandwidth values. For the sake of our experiments we often chose these values after some initial measurements to see what bandwidth values the victim Tor clients achieved.

After the download proceeds for a while, the victim connection is terminated and the traffic pattern injection is halted. Thereafter, the adversary obtains flow records from the server, corresponding to the recently computed server to exit traffic and from the entry node (or the router that had access to traffic going and from the entry node), corresponding to the clients that were using the entry node during the period when the client was communicating with the server. As described ahead in the Section IV, our experiments consisted of two stages. The first stage consisted of testing our traffic analysis attack in an in-lab environment to evaluate its accuracy in the absence network congestion and system disturbance. In these experiments the flow records were generated and captured using the open source packages of `ipt_netflow` [19] and `flow-tools` [20] respectively.

From the flows, the adversary obtains the traffic statistics corresponding to the sever to exit traffic and all the clients that were using the entry node. In case of NetFlow data, the only statistics one can obtain from the records is the bytes transferred for each time interval, based on flow start and end times. The adversary computes the correlation for the bytes transferred between the server to exit node and each of the individual clients that used the entry node. The correlation coefficient corresponding to the victim client is expected to be the highest, as it is actually the server to exit traffic carrying the injected pattern. The adversary thus selects the client whose traffic statistics closely correlate to the server to exit traffic.

The second stage consisted of testing our attack with data from a public Tor relay, serving hundreds of other Tor clients. In these experiments the, the flow records for server to exit traffic were generated and captured using the aforementioned open source packages. The flows corresponding to the entry node to client traffic were however generated first using the open source packages, and later from our institutional edge router.

This approached works wells, as it is, when the data was generated and captured from only using open source tools, as the records are of equal durations and evenly spaced. The flow interval lengths are determined by the active and inactive timers (as mentioned in Section II. In our set-up, we configured both of these to be 5 seconds. This provided us with a more or less continuous view of the network traffic and that could directly be used to compute the correlation coefficient between the server to exit and entry to client traffic statistics.

This naïve approach however didn't work when when using data from the institutional edge router. In our institutional edge router the active and inactive timers were configured to be 60 and 15 seconds respectively. We thus configured the server's NetFlow generation packages with the same parameters. However, from some initial experiments it became evident that the NetFlow data from the institutional router was often much sparse, compared to that obtained from open source tools. While the flow records derived from the server (that generated the data using the open source tools) were uniformly long and evenly spaced, such was not the case with data obtained from the network router.

A direct one-to-one comparison between the server to exit flow records, obtained from the open source tools and the entry to client records, did not yield accurate correlation coefficient, as they were not correctly aligned in time. This misalignment was not due to unsynchronized clocks. This was due primarily due to information aggregation by the router. Often data corresponding to multiple intervals were combined together into a longer interval. This undocumented feature of the routers, generally seems to arise as combination of the timeout values and existing network load. Such combination of intervals do not appear to be deterministic either. It generally seems difficult to divide such large interval into smaller intervals, especially when we have no way to determine the ordinate values for each of the new intervals.

We thus devised a method to approximate the flow records for different time intervals for the server to exit node and entry to client traffic.

a) Time aligning flow data:: Before describing how we aligned the flow data, let us describe the problem in more detail. Let us suppose that the entry node to client data is quantized into intervals as follows:

$t_{x_1} - t_{x_2}$ $\texttt{Bytes}_{x_1}$
$t_{x_3} - t_{x_4}$ $\texttt{Bytes}_{x_2}$
$t_{x_5} - t_{x_6}$ $\texttt{Bytes}_{x_3}$

[1]It is generally believed in statistics literature that there should be atleast 10 sample points for reliably computing the correlation coefficient

Here, the start and end times are shown as t_{x_i} – t_{x_j}, where t_{x_i} and t_{x_j} represent the flow start and end time points. $\texttt{Bytes}_{x_i}$ represents the data transferred during that interval.

Similarly, assume the server to exit node traffic is also quantized into the following intervals :

t_{y_1} – t_{y_2} $\texttt{Bytes}_{y_1}$
t_{y_3} – t_{y_4} $\texttt{Bytes}_{y_2}$
t_{y_5} – t_{y_6} $\texttt{Bytes}_{y_3}$
t_{y_7} – t_{y_8} $\texttt{Bytes}_{y_4}$
t_{y_9} – t_{y_9} $\texttt{Bytes}_{y_5}$

We assume that the start and end times of both the sets of intervals are not aligned. Thus, $t_{x_1} \neq t_{y_1}$ and $t_{x_2} \neq t_{y_2}$. Let us assume this to be the cases for all the intervals (the worst scenario). However, we also assume that both interval sets (namely the flows corresponding to the traffic between the server and exit node and between the entry node and possible clients) fall within some overall experiment start and end times.

We first convert the data in the intervals to a different scale. We convert the data in bytes transferred for the interval to the average traffic throughput, in terms of bytes transferred per second. For example, for the interval t_{x_1} – t_{x_2} we compute average throughput as $(\frac{\texttt{Bytes}_{x_1}}{t_{x_2}-t_{x_1}})$. Let us denote this average throughput to be R_{x_1}. Thus, let us denote the throughput for each of the intervals for the entry node to clients as R_{x_i}, $\forall\ i \in \{1,2,3\}$ (corresponding to all the intervals). Also, let us also compute the throughput values for each of the intervals corresponding to the server to exit traffic and denote them by R_{y_j}, $\forall\ j \in \{1,2,3,4,5\}$. Each of the throughput values represent the average throughput value for intervals. *This effectively translates to number of bytes transferred at each second corresponding of the interval.* Let us explain this through an example. Suppose for the interval t_{y_1} – t_{y_2}, the average throughput is R_{y_1}, then it means that at t_{y_1} seconds, R_{y_1} bytes were transferred. Similarly, at each of $t_{y_1}+1$ seconds, $t_{y_1}+2$ seconds et. R_{y_1} bytes were transferred.

Thereafter, we divide the intervals smaller as follows:

- Determine which of the two interval sets (namely that corresponding to the traffic between the server and exit, and between the entry node and the client) has more intervals values. Let us denote this set by Y. Let us denote the smaller set by X ($|Y| \geq |X|$).
- Divide the larger set (Y) into steps of one second. For example, if we have an interval t_{y_1} – t_{y_2} with throughput of R_{y_1}, we divide the interval into points that are one second apart. Thus, the points are t_{y_1}, $t_{y_1}+1$, $t_{y_1}+2$, ..., $t_{y_1}+(n-1)$,t_{y_2}. The throughput at each of these points is equal to R_{y_1}. Thus we have the following throughput values (for the interval t_{y_1} – t_{y_2}):
  $t_{y_1} : R_{y_1}$
  $t_{y_1+1} : R_{y_1}$
  $t_{y_1+2} : R_{y_1}$
  ...
  $t_{y_1}+(n-1) : R_{y_1}$
  $t_{y_2} : R_{y_1}$
- Divide the smaller set (X) into steps of only two interval points, namely the start and the end times. For example, the interval t_{x_1} – t_{x_2}, is to be divided only into two points, namely . t_{x_1} and t_{x_2} and the throughput at each would be R_{x_1} calculated using the formula mentioned above.

Having divided the intervals, we select the set X, that denotes the set of intervals with lesser number of points. For each point in X, say $t_{x_i}(\forall i \in |X|)$, we try to find the point in Y corresponding to the same time, say $t_{y_j}(\exists j \in |Y|, t_{y_j} = t_{x_i})$. These two points, namely t_{x_i} and t_{y_j}, along with their respective bandwidth values are output to two separate files. This process is repeated for all points in X. At the end of this process we have two different files which are organized as times and corresponding throughput values. The timestamps in these two files are same. The throughput values correspond to an estimated average throughput for each of those points. We compute the correlation of the throughput values in these two files. This gives us an approximated correlation for the traffic from server to exit node and from entry node to the client, even when the original flow samples are spare or not aligned with each other.

Thus, to summarize, the attack proceeds through the following steps:

1) Client downloads a file from the server that colludes with the adversary.
2) While the download progresses, the server injects a repeating traffic pattern in the TCP connection it sees originating from the exit node.
3) After sometime the download process is stopped and the server halts the traffic shaping procedure.
4) The adversary obtains network statistics from flow records corresponding to the server to exit traffic and for the various clients that used the victim entry node for the duration of the attack experiment.
5) If the flows are not correctly aligned, equally long and evenly separated, then the adversary applies the approximation strategy to correctly align them.
6) The adversary thereafter computes correlation coefficient for the server to exit traffic network statistics and for all the individual clients that used the victim entry node during the attack experiment.
7) The victim client's statistics is expected to be most correlated to the server to exit node traffic. Thus the client, whose traffic statistics are most correlated to the exit node traffic are chosen as the victim and verified.

Figure 2 pictorially presents this overall attack procedure.

IV. Experimental Evaluation

We evaluated our traffic analysis method first using and in-lab test-bed and later through experiments involving data from a public Tor relay. The former experiments we conducted to evaluate the accuracy of our correlation based traffic analysis method in an in-lab set-up in the absence of Internet traffic congestion and related network and system artefacts. Our traffic analysis attack detected the victim client in all cases. Thereafter, we evaluated the effectiveness of our attack through

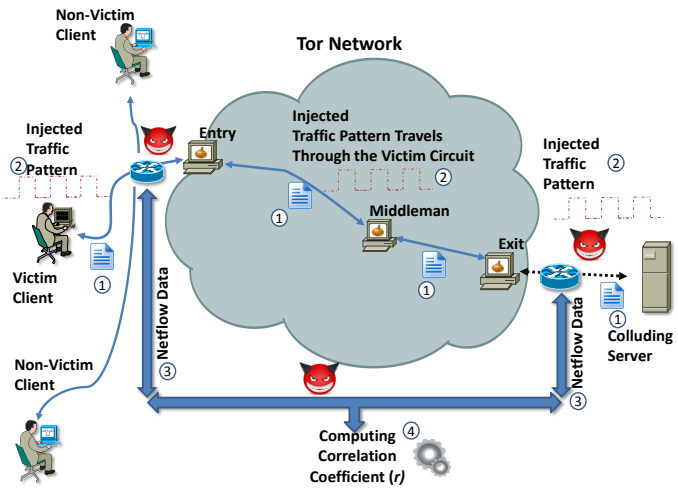


Fig. 2. Overall Process for NetFlow Based Traffic Analysis Against Tor The client downloads a file from the server ①, while the server injects a traffic pattern into the TCP connection it sees arising from the exit node ②. After a while, the connection is terminated and the adversary obtains flow data corresponding to the server to exit and entry node to client traffic ③, and computes the correlation coefficient between the server to exit traffic and entry to client statistics ④.

experiments that involved data obtained from a public Tor relay. These experiments involved both dense and sparse data and various effects of network congestion and path characteristics. Even under such conditions, we were able to identify the victim client in about 83% of the experiments (with about 5% false positives).

A. Experimental Evaluation in Controlled Environments (Using In-Lab Test-bed)

The first in-lab experiment consisted of evaluating the effectiveness of our traffic analysis attack in an in-lab set-up with a private Tor network. The set-up used is shown in Figure 3.

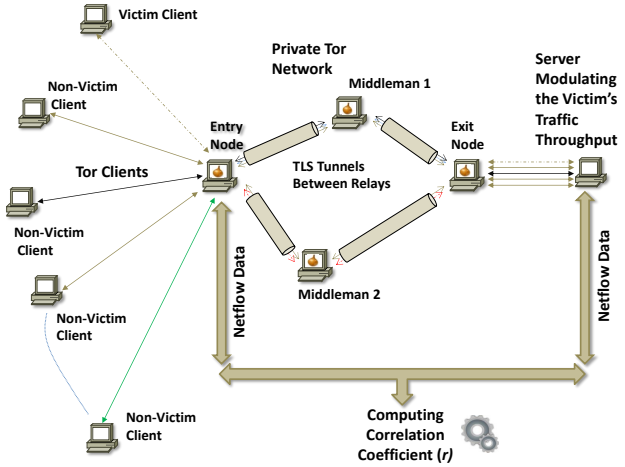


Fig. 3. In-lab test-bed used to evaluate effectiveness of NetFlow traffic analysis method.

In this set-up thirty clients simultaneously communicated to the server through circuits using the relays of the private Tor network. In reality, these clients were actually hosted on two PCs. Each PC hosted fifteen clients. The clients were connected on the same LAN using a 10/100 Mbit/s Ethernet switch (all the client IP addresses were in the same subnet). The individual network connections of the middlemen were on different subnets (corresponding to the incoming and outgoing network connections). The outgoing network connections of the middlemen were connected to two different interfaces of the exit node which connected to the server through a separate network interface, on yet another different LAN (on yet another separate subnet). Fifteen of these clients used the circuits involving the entry node, the middleman 1 and the exit node and communicated to the server. The remaining fifteen used circuits via middleman 2 (instead of middleman 1) to communicate to the server.

We restricted the number of clients to 30. From our initial experience with this set-up we realized, that when using Tor circuits, due to the packetization and Tor's traffic scheduling, each client obtained roughly 2.5 Mbit/s throughput. This number seemed adequate to test the effectiveness of attacks in an in-lab set-up in the absence of any kind of external network and system disturbances while at the same time cause the server to inject complex traffic pattern.

These clients downloaded large files from the server, which selected one of the clients and perturbed its traffic thereby injecting a traffic pattern. Our experiments were conducted with the server injecting two kinds of patterns. The first was a "square-wave" like pattern with amplitude of 1 Mbit/s. The server achieved this by repeatedly switching the victim's traffic between 2 Mbit/s and 1 Mbit/s, every 10 seconds. The second involved the server inject a complex pattern by switching the exit node to server TCP connection throughput through 2 Mbit/s, 1 Mbit/s, 256 Kbit/s, 512 Kbit/s and then again going back to 2 Mbit/s, every 10 seconds. A sample plot, corresponding to the server to exit, entry to victim and non-victim clients' throughput pattern, derived from NetFlow records is shown in Figure 4. The complex pattern is evident from the figure. The server switches the traffic throughput through the said values. This pattern is closely seen by the. Since all the non-victims were using the same entry and exit nodes, they were affected by the same traffic schedules. Since all the circuits are involved in bulk download, they are scheduled with similar priority [38]. Their traffic throughput patterns, being closely synchronized, highlights this fact.

These experiments were also repeated 60 times, 30 times corresponding to the scenario where the server injected the "square-wave" pattern and the 30 times corresponding to to "step" like pattern. The experiments ran for 400 seconds. In case of "square-wave" like pattern, we observed very high correlation between the server to exit and entry node to victim client traffic (μ:0.80 , σ:0.08) and much lower correlation corresponding to non-victim clients (μ:0.06,σ:0.16). Similarly for the "step" like pattern, we observed a very high correlation corresponding to the server to exit node and entry node to client traffic (μ:0.92 , σ:0.06) and much lower correlation corresponding to non-victim clients (μ:0.07,σ:0.14).

We continued to observe high correlation between the server to exit traffic and entry node to victim client traffic

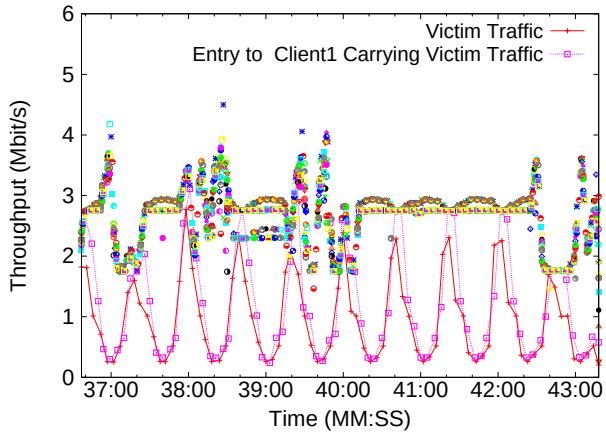


Fig. 4. Server induced "step" like pattern exactly matches the flow going towards the victim client. The points corresponding to the flows are highlighted by connecting the sample points. The remaining points correspond to 29 other non-victim flows.

statistics (μ:0.90,σ:0.06) even when the experiment lasted for a shorter duration of 200 seconds.

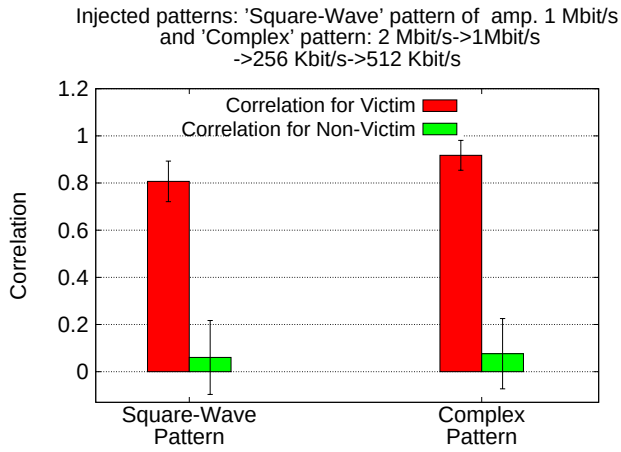


Fig. 5. Average correlation for the server induced traffic pattern and the entry to victim client traffic and the max. correlation between the server induced traffic and non-victim clients, when the server injected "square-wave" and "step" like pattern (number of clients:30).

B. Experimental Evaluation Involving Public Tor Relays

Having evaluated the accuracy of our traffic analysis attack in an in-lab environment we moved to evaluating it using real Tor traffic, obtained from our public Tor relay that served hundreds of Tor circuits simultaneously. The set-up used is as same as the one shown in Figure 2. The victim clients were hosted on three different planetlab locations, namely, Texas (US), Leuven (Belgium) and Corfu (Greece). They communicated, via Tor circuits through our relay, to a server under our control, in Spain. The victim clients downloaded a large file from the server that perturbed the arriving TCP connection's traffic, thereby deliberately injecting a traffic pattern in the flow between the server and the exit node. The process was terminated after a short while and we computed the correlation between the bytes transferred between the server and the recently terminated connection from the exit node and the entry node and the several clients that used it, during this interval.

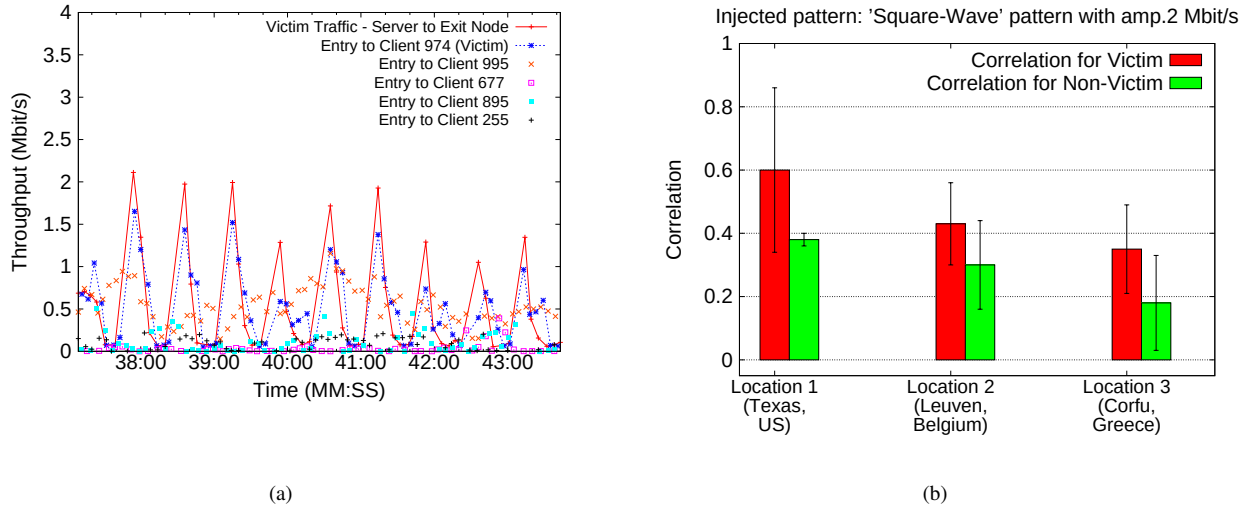
These experiments were divided into two parts. The first consisted of evaluating the effectiveness when gathering data from open-source NetFlow packages. The second part involved sparse data obtained from our institutional Cisco router.

1) Using Open Source NetFlow Tools: The first experiment involved the server injecting a "square-wave" like traffic pattern by freely switching the server to exit traffic bandwidth between approximately 2 Mbit/s and 30 Kbit/s, thereby injecting a traffic pattern with amplitude of roughly 2 Mbit/s. We used the set-up shown in Figure 2. The data was gathered from the server and from the entry node, using open source NetFlow packages. Each such experiment, involving the server switching between these bandwidth values, lasted for about 6 minutes and 40 seconds. Figure 6(a) presents sample traffic throughput variations for five flows, corresponding to one such experiment. These five flows are most correlated to the server to exit victim flow carrying the injected traffic pattern. The victim flow had the highest correlation coefficient of of 0.83, while the one with second-highest correlation, corresponding to a non-victim client, was 0.17. A total of 1104 client were using the entry node at the time of the experiment.

The figure also shows the fluctuating "square-wave" traffic pattern with an amplitude of approximately 2 Mbit/s. The trough of the "square-wave" is chosen to be at 30 Kbit/s so as to sustain the Tor circuit, between the entry node and victim-client, for the duration of the experiment. Tor clients generally terminate circuits inactive circuits (which haven't seen any activity) within about 10 minutes. We computed the correlation coefficient for the bytes transferred between the server and the exit node and each of the entry node to client flows, corresponding to the duration of the experiment. Thereafter, the client that was most correlated to the server to exit traffic was selected as the victim.

These experiments were repeated 45 times (15 times corresponding to each victim client location). Average correlation between the server to exit traffic and entry node to client traffic was 0.60 (σ:0.26), 0.43 (σ:0.13) and 0.35 (σ:0.14) for each of the clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively (see Figure 6(b)). These corresponded to the client flows that were most correlated to the server to exit flow carrying the traffic pattern. The average of second-highest correlation coefficients, corresponding to non-victims, were 0.38 (σ:0.02), 0.30 (σ:0.14) and 0.18 (σ:0.15), respectively for each of the clients for the three sets of experiments (corresponding to each of the victim client location).

Similar experiments were also repeated with the server injecting a "step" like pattern. To achieve this, the server switched the server to exit traffic between roughly 1 Mbit/s, 50 Kbit/s, 300 Kbit/s and 100 Kbit/s, every 20 seconds. This pattern was again repeated several times. Figure 7(a) shows



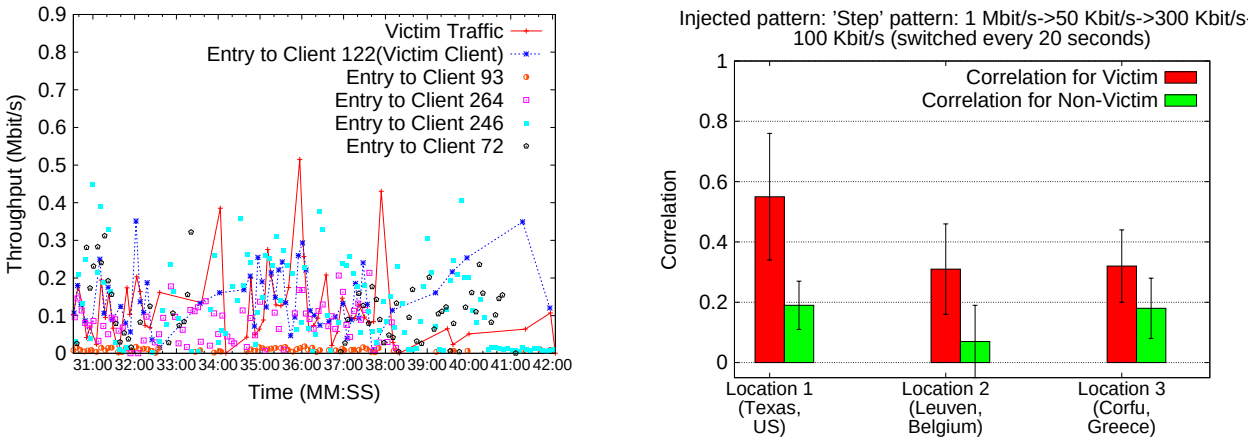
(a) (b)

Fig. 6. (a) Server induced "square-wave" like pattern exactly matches the flow going towards the victim client974 (location : Texas, US), having the highest correlation. The remaining points correspond to the other non-victim flows that whose correlation coefficients are amongst the five highest. (b) Average Pearson's Correlation between server injected "square-wave" like pattern and victim and non-victim flows corresponding to the different planetlab client locations.

one such sample where the server injected the "step" like pattern. In this experiment, the victim flow had the highest correlation coefficient of of 0.84, while the one with second-highest correlation, corresponding to a non-victim client, was 0.25. A total of 874 clients were using the entry node at the time of the experiment. These experiments were repeated 45 times (15 times corresponding to each client location). Average correlation between the server to exit traffic and entry node to client traffic was 0.55 (σ:0.21), 0.31 (σ:0.15) and 0.32 (σ:0.12) for each of the clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively (see Figure 7(b)). These also corresponded to the flows having the highest correlation coefficients. The average of second-highest correlation coefficients, corresponding to non-victims, were 0.19 (σ:0.08), 0.07 (σ:0.12) and 0.18 (σ:0.10), respectively for each of the clients for the three sets of experiments (corresponding to each of the victim client location).

We gathered a total of thirty measurements corresponding to each victim client. This included 15 measurements corresponding to both "square-wave" and "complex" patterns. This resulted in a total of 90 measurements. In most cases, we observed a clear separation between the correlation of the server to exit node traffic (carrying the induced traffic pattern) and the entry node to victim client traffic and that measured between the former and non-victim traffic. However, the average correlation of the injected pattern for the victim traffic was lower than what we observed in case of in-lab controlled testbed. This is because traffic fingerprinting pattern is distored when it leaves the Tor entry node and proceeds towards the victim client, thereby reducing the correlation of the injected traffic pattern with the entry node to victim client traffic. Further, we also found four instances where the correlation of the injected traffic with the victim traffic (from the entry node to the victim client) was lower than the correlation with some other non-victim clients' traffic. Such false negatives and false positives are primarily a combined effect of the background network congestion and routing in Tor relays, wherein the available bandwidth is distributed equally amongst all circuits, while at the same prioritizing interactive circuits over non-interactive ones [38]. Moreover, Pearson's correlation coefficient is known to be very sensitive to minor changes in input values. Small changes to input can drastically change the value of correlation coefficient.

For each of the clients, we also computed the average bandwidth variation of the traffic for the duration of the experiment. Each of these average bandwidth values were subtracted from the average bandwidth variation of the server to exit traffic. For the victim traffic, this difference is often amongst the smallest. Thus, while correlation is sensitive to small variations in input, such an approach of calculating the difference between the victim traffic and server to exit traffic, carrying the induced pattern, can be used to filter out flows that could lead to inaccurate correlation values arising out of lack of adequate values. In fact, in the next subsection we show how this heuristic could be used to filter out flows which can possibly result in inaccurate correlation between server to exit victim traffic and entry node to client traffic flows, when obtained from our institutional Cisco router. The records gathered from the Cisco router were sparse and resulted in inaccurate correlation coefficient computation. We used the above observation to filter out flows which are very unlikely to be the victim flow. The victim to entry traffic is expected to ideally have an average bandwidth variation (for the duration of the experiment), as close as possible to that of the server to exit node traffic. After filtering out the flows, we applied our approximation technique (described previously) to align corrected flow information and compute the correlation coefficient. We pick out the one which shows highest (and statistically significant, i.e. $\geq$ 0.2) correlation amongst this filtered set.



(a) (b)

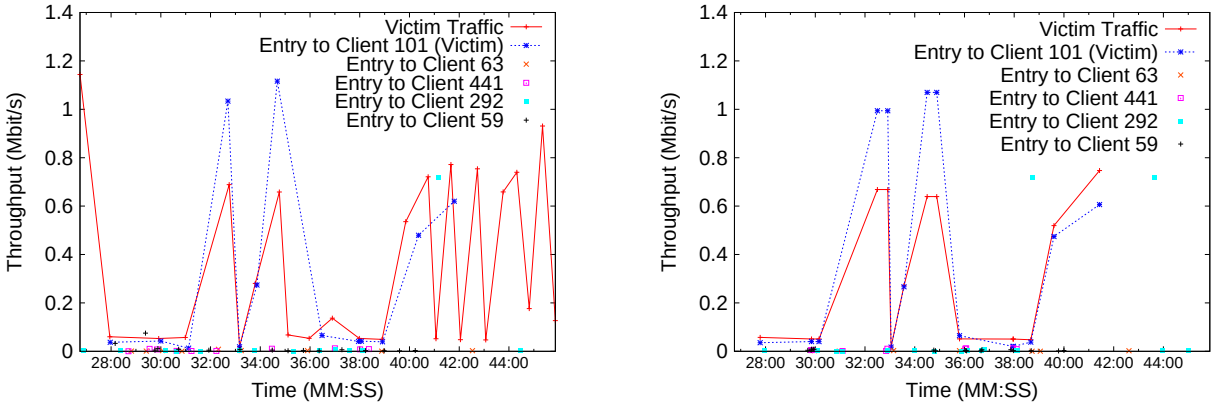
Fig. 7. (a) Server induced "step" like pattern exactly matches the flow going towards the victim client122 (location : Corfu, Greece), having the highest correlation. The remaining points correspond to the other non-victim flows that whose correlation coefficients are amongst the five highest. (b) Average Pearson's Correlation between server injected "step" like pattern and victim and non-victim flows corresponding to the different planetlab client locations.

2) Using data from Cisco router: After evaluating our traffic analysis attack with data from open source NetFlow packages, we moved to evaluating it with data from our institutional edge router. We used the same experimental set-up as that used above to test our attack using data obtained from open source packages. The differences here were that the entry node to client data was gathered from the router instead of directly collecting it from the entry node. The router was configured with an active timeout of 60 seconds and inactive timeout 15 seconds. We had no authority to modify these values as the router served large fraction of our institutional network traffic (several tens of thousands of competing flows at any given point of time). We thus configured the NetFlow packages on the server with these values. As already described in the previous section, the data obtained from the router seemed much sparse and non-uniformly aligned compared to the flow records from server to exit node. We thus applied our approximation strategy (described in the previous section) to align the flows. The strategy primarily operates by interpolating approximate bandwidth values. The rectified flow values were then directly used as input to the correlation coefficient computation formula and the correlation coefficients between the server to exit traffic and entry node to client traffic were determined.

These experiments were essentially the same as those described in the previous subsection. The first experiment involved the server injecting a "square-wave" like traffic pattern with an amplitude of approximately 1 Mbit/s. However, unlike the experiments in the previous subsection, the server switched the throughput every 30 seconds, instead of 20 seconds. This enabled us to capture adequate ($\geq$ 10) samples for computing the correlation coefficient. This was done solely to compensate for the lack of samples obtained when the experiments ran for a shorter duration of 20 seconds (as previously). For example, when injecting the "square-wave" like pattern, if the server switched the server to exit bandwidth every 20 seconds, and if this was repeated 10 times, then the total experiment ran for about 7 minutes and we observed on an average only three sample intervals corresponding to the entry node to client traffic. Figure 8(a) presents a sample bandwidth variation pattern for the server to exit traffic and the entry node to client traffic when the server injects the "square-wave" pattern with an approximate amplitude of about 1 Mbit/s. It shows server to exit traffic with more data points and fewer entry to client points. Figure 8(b) presents the same data pattern after it has been rectified using our approximation strategy. As evident, here the server to exit traffic and entry to client traffic to have equal number of points.

As mentioned in the previous subsection, eliminated flows whose average throughput variation was not comparable to that of the server to exit traffic throughput variation. To do this, we computed the difference of the average traffic throughput variation of the server to exit flow and the average traffic throughput variation of the entry node to client traffic (for all the clients). From our experience, we saw that for the victim traffic, the difference of the averages is within 120 Kbit/s. We used this as a threshold to eliminate flows whose average throughput, for the duration of the experiment, differed from the average throughput of the server to exit traffic by more than 120 Kbit/s.

Just as described in the previous subsection, these experiments were repeated 45 times (15 times corresponding to each client location). Average correlation between the server to exit traffic and entry node to victim client traffic was 0.57 (σ:0.22), 0.35 (σ:0.45) and 0.55 (σ:0.17) for each of the clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively (see Figure 9(a)). These averages were calculated from the flows having the highest correlation coefficients. The average of second-highest correlation coefficients, corresponding to non-victims, were 0.005 (σ:0.30), 0.26 (σ:0.36) and 0.24 (σ:0.17), respectively for each of the clients for the three sets of



(a) (b)

Fig. 8. (a) Server induced "square-wave" pattern of amplitude 1 Mbit/s along with other non-victim flows from the Entry node to victim and non-victim hosts having the five highest correlation co-efficient. Victim location : Texas, US. (b) Flows in Figure 8(a) adjusted and corrected using our approximation strategy.

experiments.

These experiments were repeated with the server injecting a "step" like pattern, by switching the traffic through 1 Mbit/s, 50 Kbit/s, 300 Kbit/s and 100 Kbit/s, every 30 seconds. Average correlation between the server to exit traffic and entry node to client traffic was 0.31 (σ:0.30), 0.31 (σ:0.23) and 0.42 (σ:0.15) for each of the clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively (see Figure 9(b)). The second-highest correlation coefficients, corresponding to non-victims, were -0.04 (σ:0.24), 0.14 (σ:0.29) and 0.10 (σ:0.30), respectively for each of the clients for the three sets of experiments (corresponding to each of the victim client location).

Overall we gathered a total of 90 measurement (thirty measurements for each of the planetlab client locations) and in 71 of those we were able to correctly identify the victim flow (success rate of 80%). In 13 of the remaining cases we were not able to correctly select the victim because either the correlation coefficient was statistically not significant enough (< 0.2) or the victim flow was filtered out because the average throughput differed from the average server to exit throughput by more than 120 Kbit/s.

There were six false positives in our measurements, where non-victim clients showed highest correlation amongst the filtered set, whose average throughput was close to the average throughput of the server to exit traffic. Upon closer examination we found one instance where the there were only 10 sample intervals corresponding to the server to exit traffic while there were about 36 sample intervals corresponding to the server to exit flow. Amongst the remaining five cases there were three situations where the number of sample intervals for the entry node to client was less than half the number of sample intervals corresponding to the server to exit traffic. These fewer sample intervals lead to loss of information and resulted in correlation representing an inaccurate relationship.

a) Monitoring multiple Tor relays: After testing the attack effectiveness in scenarios involving monitoring a single Tor relay, we emulated a scenario involving two relays. Instead of running a single Tor relay, we started a second one, within our institution. The purpose of this second Tor relay was to see how effective our attack was when the adversary had to monitor more clients. In the previous scenario the adversary was monitoring a single Tor relay, which on an average served about 900 clients. Upon adding the extra relay, we were able to attract about an additional 600 clients.

We repeated some of the experiments we performed with the single Tor relay scenario. The only difference here was that we performed correlation between the traffic statistics for server to exit flow and for flow corresponding to all the clients which were using the two relays. In our experiments the server injected the complex "step" like pattern by switching the server to exit traffic between roughly 1 Mbit/s, 50 Kbit/s, 300 Kbit/s and 100 Kbit/s, every 30 seconds. These experiments were repeated 24 times, 8 times corresponding to each of the victim client locations. Average correlation between the server to exit traffic and entry node to client traffic was 0.43 (σ:0.06), 0.54 (σ:0.18) and 0.35 (σ:0.05) for each of the clients at Texas (US), Leuven(Belgium) and Corfu(Greece) respectively (see Figure IV-B2a). The second-highest correlation coefficients, corresponding to non-victims, were 0.07 (σ:0.23), 0.13 (σ:0.25) and 0.23 (σ:0.02), respectively for each of the clients for the three sets of experiments.

In our measurements, we were able to correctly identify the victim client in 14 cases. There were three false positives, where the correlation of the server to exit traffic was higher for a non-victim than for the victim. The remaining seven were false negatives, where the correlation for the victim traffic was not significant (≤ 0.2). These false positives were mostly due to the reasons discussed previously. Small distortion in input traffic resulted in significant changes to Pearson's correlation. The correlation coefficient thus didn't correctly reflect the

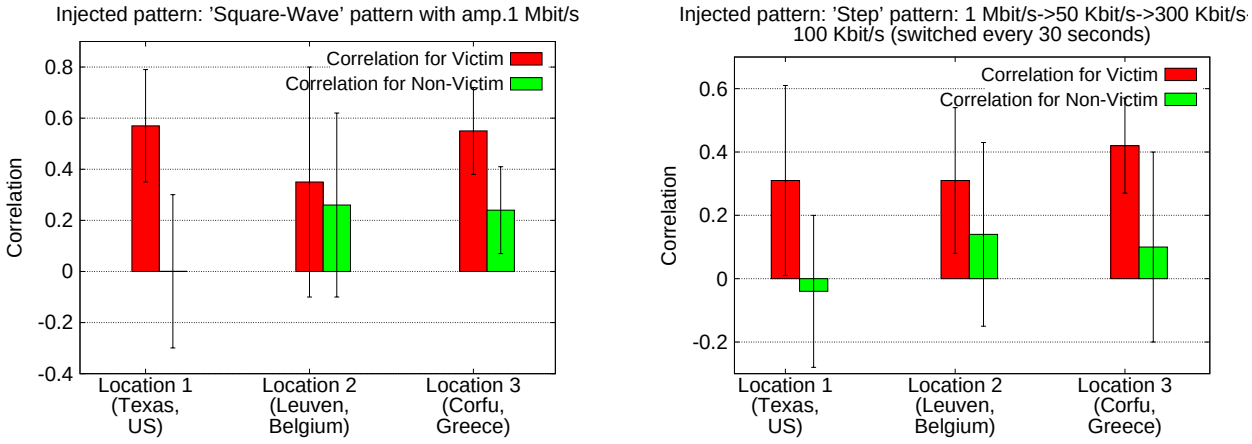


Fig. 9. (a)Average Pearson's Correlation between server injected "square-wave" like pattern of amplitude 1 Mbit/s and victim and non-victim flows, corresponding to the different planetlab client locations. The entry node to client traffic data is captured using NetFlow data derived from institutional Cisco router with NetFlow capability. (b) Average Pearson's Correlation between server injected "step" like pattern and victim and non-victim flows, corresponding to the different planetlab client locations. The entry node to client traffic data is captured using NetFlow data derived from institutional Cisco router with NetFlow capability.

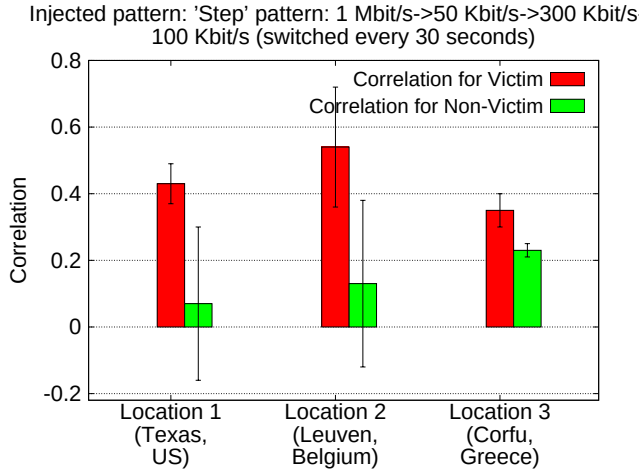


Fig. 10. Average Pearson's Correlation between server injected "step" like pattern and victim and non-victim flows, corresponding to the different planetlab client locations. The entry node to client traffic data is captured using NetFlow data derived from institutional Cisco router with NetFlow capability.

relationship between the server to exit traffic and then entry node to victim client traffic.

This scenario, involving multiple relays, provides us with an intuition of what one can expect when an adversary monitors multiple relays. However, as mentioned in Section I, an adversary need not compare the server to exit flows to flows from all the entry nodes that it monitors. He (or she) could use traffic analysis methods, such as [16]–[18] to determine the actual entry node that is being used in the victim connection, thereby only using flow statistics corresponding to the victim entry node.

V. Discussions and Future Work

Our traffic analysis attack works well in a controlled in-lab set-up with symmetric network paths and path capacities (and other properties) and least congestion and uncontrolled disturbances. In such an enviromnent we were always able to determine the identity of the anonymous client amidst, several others clients that were communicating with the server. We observed sharp contrast between the correlation of the server to exit traffic with that of the entry node to victim client and the correlation of the server to exit node traffic with entry node to non-victim clients' traffic. This enabled easy detection of the victim traffic. This is evident from the results presented in the previous section. However, on moving to tests with real Tor relays, we didn't see such high correlation for server to exit traffic and entry node to client traffic. This is because of existing path congestion and the traffic scheduling by Tor relays which distort the injected traffic pattern. The decrease in correlation is attributed to this distortion in injected pattern, thereby leading information loss. In our experiments involving gathering of data from the institutional Cisco router, such effects were quite pronounced. Moreover, the number of sample intervals were lesser, compared to data obtained from Linux Netflow packages. This was primarily due to flow aggregation arising from existing load on the routers. This undocumented feature leads to to flow records that often have unequal lengths and are not evenly spaced. To compensate for such situations, we devised our approximation strategy that align server to exit and entry to client flow records and use their statistics as input to compute correlation coefficient. Such approximations can cause decrease in the overall correlation of the server to exit traffic with the entry node to victim traffic, since the process crops out data points from the flows that cannot be aligned to any corresponding data point in

flows that the former are being compared to. This reduces the number of input points for correlation coefficient computation, resulting in a coefficient that doesn't accurately represent the variation of the server to exit traffic and entry node to client traffic. Even though the number of input samples would be more than 10, often considered a minimum required number of samples to use correlation to demonstrate similarity in variation of the input samples, the approximation strategy could still reduce the information, which would otherwise be essential to correctly demonstrate the similarity (or the lack of it) in flow statistics. For example, on closer examination of the false positives, we found out that the false positive were indeed due of extreme lack of sample intervals, for the data captured between the entry node and the victim client (often much lower than the half the number of sample intervals for the data between the server and the exit node), even when they might be considered adequate enough to be used in correlation coefficient computation.

Our future work includes implementing and testing various mechanisms to defend against such traffic analysis attacks. The various possible avenues of defense which we plan study are as follows:

- **Dummy traffic (padding)**: Using dummy traffic (also called padding) to hide patterns in correlated traffic flows has been proposed in the past [39]–[41], but has never been implemented due to the possible performance degradation. We plan to implement a conservative dummy traffic sending mechanisms such as [42], [43] or schemes whereby there is least effect of performance, while at the same time the injected patterns are not revealed through correlation.
- **Using built-in traffic shaping parameters**: Tor has built-in traffic shaping parameters, directed towards performance. These parameters, viz. `'BandwidthRate'`, `'BandwidthBurst'`, `'PerConnBWRate'` and `'PerConnBWBurst'` were designed so as to apply various traffic shaping and throttling schemes to the traffic. However, such traffic shaping schemes could have the potential to distort injected traffic patterns. We plan to study the effect of variation of these parameters on the observed correlation between flows.

VI. Conclusions

We have presented a practical traffic analysis attack against Tor, the most popular low latency anonymity preserving system, that relies on using data from existing monitoring framework, already installed in network devices. In particular we have demonstrated the practical feasibility of carrying out traffic analysis attack using statistical correlation on network traffic statistics obtained from Netflow, a popular network monitoring framework installed in various router platforms. We have demonstrated the feasibility of launching such attack to determine source of anonymous traffic. The idea of using Netflow data to de-anonymize traffic was proposed earlier [13]. However, there the authors presented a theoretical model to study the attack that relied on simulation results. The focus had been whether there were a small number of Internet Exchanged (IXes) from which such attack could be launched. We do not focus on finding approprite vantage points and monitoring hosts, but rather on the logical "next-step" once such routers have been determined. We focus on studying how successful such an attack is in practice to identify the source of anonymous traffic. We rely on correlation of traffic statistics to identify the source of anonymous traffic amidst various flows corresponding to clients using our entry node. Our research, demonstrates such an attack first on an in-lab set-up involving a private Tor network and client which we controlled. In such an environment, free from external network congestion and various artefacts due to link characteristics and path asymmetricities, we were able to determine the actual source of anonymous traffic with 100% accuracy. In experiments that involved data from public Tor relays, using both open source Netflow emulation packages and our institutional Cisco router that monitored traffic using Netflow framework, we were able to correctly identify the source of anonymous traffic in about 81.4% of our experiments, with about 6.4% false positives.

References

[1] "JAP." [Online]. Available: http://anon.inf.tu-dresden.de/

[2] "I2P Anonymous Network," http://www.i2p2.de/.

[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium)*, August 2004, pp. 303–319.

[4] J.-F. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, July 2000, pp. 10–29.

[5] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "An analysis of the degradation of anonymous protocols," in *Proceedings of the Network and Distributed Security Symposium (NDSS)*, 2002.

[6] S. J. Murdoch and G. Danezis, "Low-Cost Traffic Analysis of Tor," in *Proceedings of IEEE Symposium on Security and Privacy*, May 2005, pp. 183–195.

[7] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How Much Anonymity does Network Latency Leak?" in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, October 2007, pp. 82–91.

[8] N. Evans, R. Dingledine, and C. Grothoff, "A Practical Congestion Attack on Tor Using Long Paths," in *Proceedings of the* 18^{th} *USENIX Security Symposium (USENIX Security)*, August 2009, pp. 33–50.

[9] S. Zander and S. Murdoch, "An Improved Clock-skew Measurement Technique for Revealing Hidden Services," in *Proceedings of* 17^{th} *USENIX Security Symposium (USENIX Security)*, San Jose, USA, July 2008, pp. 211–225. [Online]. Available: http://www.usenix.org/events/sec08/tech/zander.html

[10] V.Pappas, E.Athanasopoulos, S.Ioannidis, and E.P.Markatos, "Compromising Anonymity Using Packet Spinning," in *Proceedings of the* 11^{th} *Information Security Conference (ISC)*, September 2008, pp. 161–174.

[11] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against tor," in *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES)*, 2007, pp. 11–20.

[12] X.Fu and Z.Ling, "One cell is enough to break tor's anonymity," in *Proceedings of Black Hat Technical Security Conference*, February 2009, pp. 578–589.

[13] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by internet-exchange-level adversaries," in *In Privacy Enhancing Technologies (PET), LNCS*. Springer, 2007.

[14] N. Feamster and R. Dingledine, "Location Diversity in Anonymity Networks," in *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES)*, October 2004, pp. 66–76.

[15] M. Edman and P. F. Syverson, "AS-awareness in Tor path selection," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, November 2009, pp. 380–389.

[16] S. Chakravarty, A. Stavrou, and A. D. Keromytis, "Traffic analysis against low-latency anonymity networks using available bandwidth estimation," in *Proceedings of the 15th European conference on Research in computer security*, ser. ESORICS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 249–267. [Online]. Available: http://dl.acm.org/citation.cfm?id=1888881.1888901

[17] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 215–226. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046732

[18] S. Chakravarty, A. Stavrou, and A. D. Keromytis, "Identifying Proxy Nodes in a Tor Anonymization Circuit," in *Proceedings of the* 2^{nd} *Workshop on Security and Privacy in Telecommunications and Information Systems (SePTIS)*, December 2008, pp. 633–639.

[19] "Netflow iptables module." [Online]. Available: http://sourceforge.net/projects/ipt-netflow/

[20] "Netflow iptables module." [Online]. Available: http://freecode.com/projects/flow-tools

[21] "Tor Metrics Portal." [Online]. Available: http://metrics.torproject.org/

[22] "Tor Path Specification." [Online]. Available: https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt

[23] E. B. Claise, "Cisco systems netflow services export version 9," http://www.ietf.org/rfc/rfc3954.txt.

[24] "J-Flow Statistics." [Online]. Available: http://www.juniper.net/techpubs/software/erx/junose82/swconfig-ip-services/html/ip-jflow-stats-config.html

[25] "Huawei NetStream Analysis, Monitoring and Reporting." [Online]. Available: http://www.solarwinds.com/solutions/netstream-analyzer.aspx

[26] "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks." [Online]. Available: http://www.ietf.org/rfc/rfc3176.txt

[27] "NetFlow probes: fprobe and fprobe-ulog." [Online]. Available: http://fprobe.sourceforge.net/

[28] "Flexible NetFlow Command Reference." [Online]. Available: http://www.cisco.com/en/US/docs/ios/fnetflow/command/reference/fnf_cr_book.pdf

[29] V. Shmatikov and H. M. Wang, "Timing analysis in low-latency mix networks: attacks and defenses," in *Proceedings of the 11th European conference on Research in Computer Security*, ser. ESORICS'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 18–33. [Online]. Available: http://dx.doi.org/10.1007/11863908_2

[30] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, October 2006, pp. 27–36.

[31] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh," in *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10*, ser. SSYM'01. Berkeley, CA, USA: USENIX Association, 2001, pp. 25–25. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251327.1251352

[32] E. W. Felten and M. A. Schneider, "Timing attacks on web privacy," in *Proceedings of the 7th ACM conference on Computer and communications security*, ser. CCS '00. New York, NY, USA: ACM, 2000, pp. 25–32. [Online]. Available: http://doi.acm.org/10.1145/352600.352606

[33] S. Chen, X. Wang, and S. Jajodia, "On the anonymity and traceability of peer-to-peer voip calls," *Netwrk. Mag. of Global Internetwkg.*, vol. 20, no. 5, pp. 32–37, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/MNET.2006.1705881

[34] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the internet," in *Proceedings of the 12th ACM conference on Computer and communications security*, ser. CCS '05. New York, NY, USA: ACM, 2005, pp. 81–91. [Online]. Available: http://doi.acm.org/10.1145/1102120.1102133

[35] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper, "Routing around decoys," in *Proceedings of the 2012 ACM conference on Computer and communications security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 85–96. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382209

[36] D. Mccoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the tor network," in *Proceedings of the 8th international symposium on Privacy Enhancing Technologies (PETS)*, 2008, pp. 63–76.

[37] B. Hubert, T. Graf, G. Maxwell, R. Mook, M.Oosterhout, P.Schroeder, J. Spaans, and P. Larroy, "Linux Advanced Routing and Traffic Control HOWTO." [Online]. Available: http://lartc.org/howto

[38] C. Tang and I. Goldberg, "An improved algorithm for Tor circuit scheduling," in *Proceedings of the 2010 ACM Conference on Computer and Communications Security (CCS 2010)*, A. D. Keromytis and V. Shmatikov, Eds. ACM, October 2010.

[39] X. Fu, B. Graham, R. Bettati, and W. Zhao, "Analytical and empirical analysis of countermeasures to traffic analysis attacks," in *Proceedings of the 2003 International Conference on Parallel Processing*, 2003, pp. 483–492.

[40] N. Mallesh and M. Wright, "Countering statistical disclosure with receiver-bound cover traffic," in *Proceedings of 12th European Symposium On Research In Computer Security (ESORICS 2007)*, ser. Lecture Notes in Computer Science, J. Biskup and J. Lopez, Eds., vol. 4734. Springer, September 2007, pp. 547–562.

[41] O. Berthold and H. Langos, "Dummy traffic against long term intersection attacks," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*, R. Dingledine and P. Syverson, Eds. Springer-Verlag, LNCS 2482, April 2002.

[42] W. Wang, M. Motani, and V. Srinivasan, "Dependent link padding algorithms for low latency anonymity systems," in *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, P. Syverson, S. Jha, and X. Zhang, Eds. ACM Press, October 2008, pp. 323–332.

[43] V. Shmatikov and M. H. Wang, "Timing analysis in low-latency mix networks: Attacks and defenses," in *Proceedings of ESORICS 2006*, September 2006.