Identifying Proxy Nodes in a Tor Anonymization Circuit

Sambuddho Chakravarty
Columbia University, NY
sc2516@cs.columbia.edu

Angelos Stavrou
George Mason University, VA
astavrou@gmu.edu

Angelos D. Keromytis
Columbia University, NY
angelos@cs.columbia.edu

Abstract

We present a novel, practical, and effective mechanism that exposes the identity of Tor relays participating in a given circuit. Such an attack can be used by malicious or compromised nodes to identify the rest of the circuit, or as the first step in a follow-on trace-back attack. Our intuition is that by modulating the bandwidth of an anonymous connection (e.g., when the destination server, its router, or an entry point is under our control), we create observable fluctuations that propagate through the Tor network and the Internet to the end-user's host. To that end, we employ LinkWidth, a novel bandwidth-estimation technique. LinkWidth enables network edge-attached entities to estimate the available bandwidth in an arbitrary Internet link without a cooperating peer host, router, or ISP. Our approach also does not require compromise of **any** *Tor nodes. In a series of experiments against the Tor network, we show that we can accurately identify the network location of most participating Tor relays.*

1 Introduction

Network anonymity systems were first introduced as early as 1981 [7]. However, it was until much later that a practical software system that enables its users to communicate anonymously on the Internet was introduced. Tor [8] employs an application agnostic Onion Routing scheme [11] to anonymize the traffic. The strength of the scheme is the use of strong, layered cryptography combined with a wide network of relay nodes located across different administrative domains.

Although state-of-the-art, like all current low-latency anonymity systems, Tor is vulnerable to traffic analysis attacks. These attacks can expose the identity of a Tor client or of the proxy nodes used in an anonymous Tor circuit when an adversary can manipulate the traffic entering and leaving that circuit [1, 3, 12, 17]. In order to be successful, these attacks assume that one or more of the Tor relays or active network components (*e.g.,* routers) that participate in the client's relay are known and can be manipulated by the adversary. Here, it is important to note that knowledge of the Tor circuit would make water-marking and all such attack techniques much more effective and practically deployable. Indeed, if the Tor circuit is exposed, the adversary would have the option to inject the watermark via any device on any of the administrative domain the packet traverses, including different ISPs and countries.

In this paper, we describe a technique for identifying the Tor nodes participating in a circuit of interest. For doing so, we introduce and use *LinkWidth*, a novel *single-end controlled* available bandwidth estimation tool. LinkWidth uses an edge-attached host to estimate the available bandwidth on an arbitrary network link *without* direct access either to that link itself or to an appropriately positioned cooperating host. An adversary may use LinkWidth to detect induced traffic fluctuations on the anonymous Tor circuit to a server of interest. These fluctuations can be created by the server itself (if it is cooperating with the attacker), by a router or node close to the server (*e.g.,* when collaborating with the server's ISP or otherwise hijacking/compromising/legally compelling use of such a node), or by launching a targeted network denial of service attack against the appropriate link(s), router(s) or the server itself. In our experiments, we assumed that the attacker controls the server.

However, any other scheme for causing "large enough" traffic variations, as defined later in the paper, would suffice. Our scheme empowers an attacker with access only to a few high-bandwidth edge hosts to identify the Tor nodes participating in a circuit. This knowledge can be used in a number of follow-on attacks, as it enables the violation of a key property of Tor and similar anonymity networks: no entity other than the user should have complete knowledge of the circuit path (*i.e.,* the identity of the proxies). We stress that we do not assume that the attacker has access to large numbers of routers, network infrastructure nodes (*e.g.,* DNS or DHCP servers), or Tor nodes, nor do we exploit software vulnerabilities that inadvertently expose the true network identity of the user.

To evaluate our measurements, we built a prototype of LinkWidth and evaluated its effectiveness in detecting small variations in available bandwidth through a series of experiments in a lab environment. We also launch an attack against a number of circuits through Tor, aiming to expose the intermediate Tor relays. Even with our limited resources, we could successfully probe 26 separate Tor circuits with a true positive rate of 59.46% and true negative rate of 10%. Details of our experiments and their outcomes are documented in Section 4.

Possible countermeasures to our attack include shorter circuit lifetimes, limited traffic smoothing by Tor nodes, striping over multiple parallel circuits to access the same server, and preventing the use of long-lived TCP connections. We note that the use of longer Tor circuits does not appear to make the attack more difficult [4].

The novel contributions of this paper include:

• An attack against Tor and similar anonymity systems, wherein a bandwidth-provisioned adversary can expose the network identity of all Tor relays.

• *LinkWidth*, a novel single-end available-bandwidth estimation technique.

• An demonstration of our proxy-discovering attack against Tor.

2 Related Work

Onion-routing anonymizing networks [24] use multi-hop encrypted communications to protect sender and/or receiver anonymity. Tor [8] extends the existing onion routing scheme by adding support for integrity protection, congestion control, and location-hidden services through rendezvous points. Tor can be used for both *initiator* and *responder* anonymity. Initiator anonymity allows a client to hide its true identity from a server. Responder anonymity allows a server to provide a TCP service without revealing its IP address. Tor circuits are formed using three Onion Routers (ORs) (by default). The first hop is called the *Entry Node*, the second the *Middleman*, and the third the *Exit Node*. A Tor Client, known as Onion Proxy (OP) in Tor terminology, uses the public-keys (Onion Keys) of the three ORs to to establish shared secrets with them. The OP fragments the data into 512 byte units called *cells*. These cells are encrypted incrementally using the shared secrets of the Exit Node, the Middleman and the Entry Node. This technique, common for many anonymizing networks and mixes, is known as *Telescopic Encryption*. Each of the three ORs "peels-off" the headers off the Tor cell it receives and forwards it to the next OR along the circuit. The Exit Node decapsulates and decrypts the Tor cell retrieving the payload which is sent encapsulated into a regular TCP/IP header to the intended destination. Since an OP selects different ORs for every new circuit, the destination receives packets from different Tor Exit Node each time a new circuit is established.

Tor trades off resistance against certain adversaries with better performance characteristics relative to more robust privacy mechanisms, such as MIXes [6] and DC-nets [5, 26]. An adversary observing all links in an onion routing network can record arrival and departure times for all messages in the network and use statistical methods to determine exactly who is communicating with whom [20, 22, 27]. However, Tor is considered "secure enough" in practice for semi-interactive traffic, *e.g.* web sessions, because few entities are believed to have the ability to act as global passive adversaries. This is precisely the assumption that our attacks work.

There have been prior efforts in using network latency to attack Tor. Perhaps the closest prior work is the one from Murdoch *et al.* [17]. They focus on using network latency for determining if a relay node is a part of a specific Tor circuit. Their method requires a server to send pseudo-random data as fast as allowed by the underlying network to the victim client. The adversary uses a modified Tor Proxy for establishing single-hop circuits (rather than the default 3 hops) through the victim Tor relay, back to itself. The corrupt server sends traffic to the client having a particular "on-off" pattern. The adversary attempts to observe the variation in one-way delay through the victim Tor relay due this induced network traffic fluctuation. Higher correlation between these induced fluctuations and the observed one-way latency distortions gives a better probability that the victim Tor relay is the one which is a part of the victims Tor circuit. In contrast, our approach is much less invasive and does not require the inclusion of a malicious Tor relays, padding, extra traffic, and nominal network conditions (*i.e.,* no congestion).

Hopper *et al.* [12] go a step ahead and try to use a combination of this technique and pairwise round-trip times (RTTs) between Internet nodes as input to statistical measures to correlate Tor nodes to probable clients. In addition, their method can be extended to application-layer RTT estimates (rather than TCP RTT estimates). *However we argue that RTT is a temporal network parameter which cannot be considered a constant.*

Previous work also disregarded traffic filtering and shaping either at the end hosts or at network edges. Our scheme uses TCP packets for probing. Where TCP is filtered or rate-limited, we emulate the same behavior using ICMP. Unlike previous work, we assume least control over various network elements. All such traffic analysis attacks may fail if the Tor relays perform traffic engineering by controlling the outgoing traffic rate and burst length.

Bandwidth Estimation Prior research in bandwidth measurement has taken two major forms [21]. One focuses purely on the measurement of bottleneck bandwidth for IP payload. Tools such as Pathchar [14], Pathrate [9] and Pchar [10] measure the bottleneck bandwidth. These techniques rely on the *Packet Pair Technique* [15]. A pair of packets, sent back-to-back to the destination, "spreads" in time. This spreading in time, known as *received dispersion*, is inversely proportional to the bottleneck link capacity. The capacity is thus measured as $B = L/T$. In this formula, L is length of the second transmitted packet (in bits). T, the dispersion, is measured as the latency between the reception of the last bit of the first packet and the last bit of the second packet. The *Packet Train Technique* extends the Packet Pair Technique by sending a train of packets. The use of more packets minimizes the error due to noise and cross traffic. A detailed discussion of various packet pair and packet train techniques and their comparison can be found elsewhere [19].

The other major family of measuring techniques focuses on the estimation of end-to-end throughput, typically for use with transport-layer protocols such as TCP. The transport-protocol mechanics are geared towards optimizing in-order and correct delivery of messages in the presence of unreliable links without under-utilizing the end-to-end path capacity. Therefore, it is important for TCP to determine the number of bits correctly received since the previously received acknowledgment. Tools such as Iperf [25] and abget [2] and Sprobe [23] come close to measuring throughput.

3 Approach

Our attack on Tor relays leverages *LinkWidth*, our single-end measurement and estimation technique for bandwidth estimation. Below we present a more detailed description of how we identify the Tor nodes participating in a given circuit:

• An adversary continuously senses the available bandwidth in the up-links of all Tor relays. These may be the immediate up-link, or some other link that carries all traffic to and from the Tor node.

• When an anonymous user contacts a server of interest and requests data (*e.g.,* a web page), the traffic from the server to the Tor exit node is artificially modulated. This modulation can be done by the server itself, or by an upstream router that is under the control of the attacker. The modulation can be as simple as temporarily queuing all traffic and then releasing it in a high-volume burst, or may involve a unique throughput pattern. The goal of the adversary is to detect this pattern with high confidence as it manifests itself in the three Tor relays.

Alternatively, an adversary interested in identifying *all* users accessing a server that is not under his control may launch a network denial of service attack against the server or one of its up-links, causing a back-off in TCP connections and hence an increase in available bandwidth in these links traversed by those connections. This scenario requires more resources (in terms of bandwidth) on the part of the attacker. Lacking these, we decided to focus on the malicious/compromised server/router scenario. We note that we require *at most* one such router, and its identity/location is independent from that of the client.

Our technique works well only when we use a well-provisioned probing node is at a network "vantage" point with respect to the victim Tor relay. Stated simply, this would mean that the bottleneck in the path connecting the adversary to the victim relay should be the latter. However, we posit that nodes in major ISPs, government organizations and universities do have such capabilities. The original Tor model does aim to mitigate such *traffic analysis* attacks. We only try to approximate a global passive adversary (which may be anyways a difficult task due to limited control over network links and resources). We end up with a "pseudo" global passive adversary while still managing to only target a vulnerability *within* the Tor threat model.

In the next subsection, we provide a in-depth explanation of the measurement methods that we use as part of our bandwidth-estimation tool: *LinkWidth*.

3.1 LinkWidth

LinkWidth is a tool that allows us to estimate available and capacity bandwidth on a path, without additional support or active collaboration from a remote host or any device in the network. LinkWidth transmits TCP SYN packets to a remote host on a closed TCP port. The receiver (a router or end-host) replies with a TCP packet where either RST and ACK flags (closed port) or SYN and ACK (open port) are set. Where TCP packets are filtered and/or rate limited due to security considerations, we can use ICMP ECHO_REPLY messages from the receiver to signal correct reception of probe packets (by sending ICMP ECHO_REQUEST packets instead of TCP SYNs). This is described in more detail in our technical report [4].

To measure end-to-end TCP capacity, the sender emulates the TCP Westwood sender by sending *cwin* packets. $cwin - 2$ TCP RST packets (called *load packets*), are "sandwiched" between two TCP SYN Packets (called the *head measurement packet* and *tail measurement packet* respectively). These TCP SYN packets, sent to closed ports, evoke TCP RST+ACK reply packets. Correct reception of the train of $cwin + 1$ packets is determined by two TCP RST+ACK packets from the receiver (due to the head and tail measurement packets). Each correct reception of the TCP RST+ACK pair causes *cwin* to be increased either exponentially (Slow Start phase) or linearly (Congestion Avoidance phase). Since we do not rely on an established

TCP connection, the only way to signal a packet loss is by coarse timeout. After sending the train, the sender initializes a timer to wait for the two expected ACKs. The expiration of the timeout causes the readjustment of the *cwin* and *ssthresh* parameters inside a timeout event handler method.

We use TCP RST packets to avoid generating unnecessary replies, either in the form of TCP RST or ICMP Destination Host/Net Unreachable packets, which could potentially interfere with our forward probe traffic. The time dispersion between two consecutive TCP RST+ACK replies due to the head and tail measurement packets are stored as t_n and t_{n-1}. Thus the capacity/bandwidth is measured as:

$$b_k = (cwin * L)/(t_n - t_{n-1})$$

Here, b_k is the measured "instantaneous" bandwidth (measured throughput), $cwin * L$ is the total data sent (in bits) for the entire train, t_n and t_{n-1} are the times of reception of the two TCP RST+ACK reply packets. The successful reception to a previous train determines how many packets we send in the current train. Our method is a direct extension of the packet train method.

The throughput measurement is a slight modification of the capacity measurement. The TCP RST packets are replaced by TCP SYN packets. The time of reception of the TCP RST+ACK due to the first TCP SYN packet is stored in the variable *first*. Thus, for any value of *cwin*, if any m replies are received correctly (such that $1 \leq m \leq cwin$), this indicates that the throughput is:

$$b_k = (m * L)/(T_m - \mathit{first})$$

where t_m is the time when the m^{th} reply is correctly received. LinkWidth reports the measurement as BWE.

The arrangement of packets differs when using ICMP packets. We replace the head and tail TCP SYN packets with ICMP ECHO packets. The load packets continue to be TCP RST packets. Correct reception of the train is indicated by reception of ICMP ECHO_REPLY packets at the sender. A similar modification is used for measuring throughput: the receiver waits to see how many ICMP ECHO_REPLY response packets it receives before estimating the throughput.

We developed a prototype of LinkWidth for GNU/Linux. To avoid incurring packet delays due to kernel resource scheduling, we bypassethe regular protocol stack and send our own TCP and ICMP packets crafted using the Raw Socket API. The coarse timeout is implemented using the standard POSIX API function *settimer()*. The expiration of the timer is indicated by raising a $SIGALRM$ signal.

4 Detecting Proxy Nodes

We employ LinkWidth to detect induced traffic fluctuations in Tor relays participating in a circuit. We demonstrate how an adversary may use LinkWidth for linking Tor relays to Tor clients and/or servers of a communication.

We have performed extensive measurements in a controlled lab-environment to measure the effectiveness of LinkWidth in detecting small bandwidth fluctuations. The results of our controlled experiments are included in a technical report [4].

We use LinkWidth to do traffic analysis for uncovering the Tor relays participating in a communication. This is achieved by measuring the fluctuation of available bandwidth or throughput of the probable Tor relays participating in a circuit, induced by a colluding client or server, from a network "vantage" point. Further, the information of relays participating in a circuit may be used to launch trace-back attack to determine Tor clients or Hidden servers participating in a communication. This might require a well provisioned adversary, with possibly multiple vantage point and a map of the network. He/She should be observing induced fluctuation on network routers connecting the Tor end-points to their respective Entry Nodes[1].

Figure 1 illustrates how an adversary probes the Tor relays involved in a circuit. In our experiments, we use LinkWidth to probe Tor relays that may possibly be part of Tor circuits. An adversary with sufficient bandwidth resources can simultaneously probe all (or a large fraction of) the advertised Tor nodes. To detect the participating Tor relays, the adversary can collude with server and induce fluctuations in bandwidth of existing anonymous circuits (basically TCP connections). *The goal is to demonstrate that an adversary is able to detect these induced fluctuations in bandwidth in the Tor relays whenever the client downloads the file from the server.*

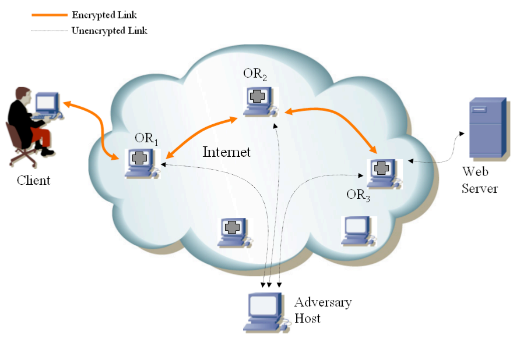

Figure 1. Adversary probing the fluctuation in available bandwidth of ORs participating in a Tor circuit

To bypass the default restrictions on Middleman Node selection, we modified the Tor Client *version 0.1.2.18* to enable the establishment of circuits in which the users can

[1]Optimizing the adversary's search of links to probe is a different problem in itself.

select all the ORs manually[2].

In our experiments, the client, the web server, and the probing host used by the adversary are all scattered in separate geographic locations and networks within the US[3]. The web server offers a 100 MByte file; a relatively large file for providing adequate delay to the adversary for perceiving the changing network congestion at each value of the client-server traffic bandwidth. The available bandwidth of the client-server TCP connection is shaped using the Linux Traffic Controller [13]. Because the client selects the Tor relays from the publicly available Tor status page [16], the adversary also has knowledge of all the available Tor nodes. Therefore, the adversary can detect the Tor relays in the circuit by measuring bandwidth variations whenever the client communicates to the web server through the Tor circuit.

We quantify the effectiveness of detecting the ORs involved in Tor circuits by creating 26 distinct Tor circuits. We probed these Tor relays from different network locations. The results, summarized in Table 1, indicate how many of the Tor relays in each circuit reported fluctuation in available bandwidth (and thus were correctly identified by the attacker).

Relays/Circuit Detected	**# of Circuits**
3	8
2	8
1	4
0	6

Table 1. Number of Tor relays per Tor circuit where available bandwidth fluctuation is correctly detected.

In our experiments, we successfully identified all the relay nodes in 8 out of the 26 circuits that we probed. For 8 circuits, we were able to identify 2 of the 3 participating Tor relays. There were 4 circuits in which only one 1 of the relays was detected. Finally, there were 6 circuits in which we could detect fluctuation in none of the relays involved. Amongst the 6 undetected, were 4 relays which filtered all probe traffic. Using these results we can compute the true positives and false negatives:

Total number of nodes probed $= 74$ (N) (not counting the 4 which filtered all probe traffic)
Total number of nodes in which bandwidth fluctuation was observed $= 44$ (T)
True Positives $(T/N) = 44/74$ (59.46%)
False Negatives $(1 - T/N) = 30/74$ (40.54%)

To determine the false positives, we repeated the same experiment: we generated 10 different "3-hop" Tor circuits and for each of these circuits, we selected ORs not participating in any of these circuits and probed them for available bandwidth fluctuations induced by the colluding server. The results from these experiments are summarized as follows:

Total number of nodes probed $= 30$ (N)
Total number of non-participating nodes which report fluctuation $= 3$ (F)
False Positives $(F/N) = 3/30$ (10%)
True Negatives $(1 - F/N) = 27/30$ (90%)

In the first set of experiments, we observed bandwidth fluctuation in some relays which were not part of our client's anonymous circuit. However, on repeated attempts of the same experiment, we saw no fluctuation in available bandwidth; thereby detected no false positives.

In laboratory conditions, our system can successfully detect 50 Kbps fluctuations in bandwidth, or approximately 6 KBytes/second. In a "noisy" network environment with cross-traffic, our detection ability is limited to variations that are at least 30–40 KBytes/second.

Most of the ORs filter and/or rate limit TCP SYN packets to closed ports. In the presence of such filtering, we use the ICMP-based emulation of LinkWidth. Probes using ICMP are prone to error due to difference in dispersion of the replies from those of the forward probe traffic (when the probes reach the destination). Moreover, prioritized packet-scheduling and forwarding of data packets over control packets, in modern IP routers, can result in erroneous measurements. This refers only to probe packets that use ICMP.

From our experience of real-world Tor circuits, we have learnt that correct detection of the exact pattern of increasing or decreasing available bandwidth is contingent upon various factors. The most restricting appears to be the number and quality of the attackers' network vantage points. If this condition is not met, then accurate bandwidth measurement may not be possible in all cases (depending on link utilization). In practice, our use of a few vantage points in academic institutions in the US seems to provide sufficient bandwidth to conduct our attack against a large subset of the Tor network. For an attack against all of the Tor relays, we would require access to nodes in the different major geographical areas hosting Tor relays (Europe, Asia)

[2]Selection of the intermediate relay nodes cannot be controlled in the standard Tor distribution. There are, however, source-code options that can unlock the intermediate relay selection.

[3]We avoid selecting nodes in Europe and Asia so as to avoid intercontinental Internet links which may a times act as bottlenecks. An adversary would need nodes in these geographical locations from which to probe the relevant Tor relays. While this is generally feasible, we do not currently have such capabilities.

such that we do not have to probe over trans-Atlantic or trans-Pacific long-haul links that affect the accuracy of our measurements.

5 Discussion

In the attack presented so far, we avoid crossing the continent in our search for Tor relays. The trans-continental links are a bottleneck in many instances, and we do not have network vantage points located outside the US. To make things worse, out of the approximately 150 Exit Nodes within the US that were available at the time of our experiment, less than 100 provide adequate quality of service and exit policies that allow our client's traffic. Unfortunately, PlanetLab [18] hosts cannot provide accurate traffic measurements because we cannot control kernel scheduling of network resources. Consequently, they cannot be used as neither network vantage points nor colluding entities.

To obtain high confidence detection results, we need to observe traffic from Tor nodes with throughput of at least 30–40 KBytes/sec (approximately 300 Kbps). In addition, packet loss, traffic filtering and shaping, intermediate network bottlenecks, and operating system and networking device driver issues play an important role in measurement-based network monitoring. TCP SYN packets are also filtered by some Tor relays. In most instances where TCP probes were rate-limited or filtered, we rely on ICMP probes. However, as mentioned earlier in Section 4, modern IP routers prioritize data traffic (*e.g.,* TCP and UDP) over control traffic (*e.g.,* ICMP). Thus ICMP-based emulation may be affected due to erroneous values of end-to-end dispersion.

The effectiveness of the Tor attacks presented earlier is constrained by the limitations of the traffic-based measuring techniques. These limitations could be leveraged to create countermeasures to our attack. First, a Tor client can use parallel circuits in a round-robin fashion to access the same server; this would diffuse the ability of the server to generate detectable traffic variations, since traffic spikes would be distributed across all parallel connections. The use of shorter circuit lifetimes may prevent an adversary from detecting Tor relays with high confidence. Traffic smoothing by Tor relays is another potential countermeasure. Use of Tor circuits with more relay nodes, leading to longer network paths, does not appear to make the attack appreciably more difficult. However, it can significantly affect the client-perceived latency and throughput of the connection [4], making it a particularly unsuitable countermeasure.

6 Conclusion

We propose a new technique for uncovering the identity of Tor relays participating in an anonymous circuit. Our scheme works by artificially inducing traffic fluctuations in the traffic sent by the server to the anonymous client. This can be achieved by colluding with the server, controlling an upstream router, or possibly through a targeted DDoS attack. We detect these perturbations as they traverse the network using single-end bandwidth measurements. From our experiments on 26 Tor circuits we showed that we were able to expose Tor relays with a true positive rate of 59.46% and a true negative rate of 10%. This knowledge can then be used in follow-on attacks, either against the anonymous client or against the Tor network itself. For accurately detecting the relevant nodes, it is essential for an adversary to be at a "vantage point" in the network such that either the bottleneck link itself or the disturbance in cross traffic caused by the server is enough to distort LinkWidth's probes. As future work, we plan to extend our technique to produce a full trace-back to the Tor client and Hidden Servers using our single-end bandwidth estimation technique.

References

[1] D. Agrawal and D. Kesdogan. Measuring Anonymity: The Disclosure Attack. *IEEE Security & Privacy*, 1(6):27–34, November/December 2003.

[2] D. Antoniades, M. Athanatos, A. Papadogiannakis, E. P. Markatos, and C. Dovrolis. Available Bandwidth Measurement as Simple as Running wget. In *Proceedings of Passive and Active Measurements (PAM)*, March 2006.

[3] K. Borders and A. Prakash. Web Tap: Detecting Covert Web Traffic. In *Proceedings of the* 11^{th} *ACM Conference on Computer and Communications Security (CCS)*, pages 110–120, October 2004.

[4] S. Chakravarty, A. Stavrou, and A. D. Keromytis. Approximating a Global Passive Adversary Against Tor. Computer Science Department Technical Report cucs-038-08, Columbia University, August 2008.

[5] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[6] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communincations of the ACM*, 24(2):84–90, February 1981.

[7] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Psuedonyms. *Communications of the ACM*, 1981.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *In Proceedings of the* 13^{th} *USENIX Security Symposium*, pages 303–319, August 2004.

[9] C. Dovrolis and R. Prasad. Pathrate. `http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/pathrate.tar.gz`, 2004.

[10] A. B. Downey. Using pathchar to Estimate Internet Link Characteristics. In *Proceedings of ACM SIGCOMM*, August 1999.

[11] D. Goldschlag, M. Reed, and P. Syverson. Onion Routing for Anonymous and Private Internet Connections. *Communications of the ACM*, 43(2):39–41, 1999.

[12] N. Hopper, E. Y. Vasserman, and E. Chan-Tin. How Much Anonymity does Network Latency Leak? In *Proceedings of ACM CCS*, October 2007.

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

[14] V. Jacobson. PATHCHAR. `http://www.caida.org/tools/utilities/others/pathchar/`, 1997.

[15] S. Keshav. Congestion Control in Computer Networks. UC Berkely Technical Report TR-654, September 1991.

[16] J. B. Kowalski. TorStatus. `http://anonymizer.blutmagie.de:2505/`.

[17] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *IEEE Symposium on Security and Privacy*, pages 183–195, May 2005.

[18] PlanetLab. `http://www.planet-lab.org/`.

[19] R. Prasad, M.Murray, C. Dovrolis, and K. Claffy. Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. In *Proceedings of IEEE Network*, August 2003.

[20] 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*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[21] M. Y. Sanadidi. Bandwidth Estimation Techniques , A Tutorial Presentation. In *SBRC 2002:Brazilian Symposium on Computer Networks Date*, Buzios, Brazil, May 2002.

[22] A. Serjantov and P. Sewell. Passive Attack Analysis for Connection-Based Anonymity Systems. In *Proceedings of ESORICS*, October 2003.

[23] Stefan, Saroiu, and Krishna. Sprobe: Another tool for measuring bottleneck bandwidth. In *Proceedings of InfoComm 2002*, 2002.

[24] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion Routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, May 1997.

[25] A. Tirumala, F. Qin, J. Dugan, J. Feguson, and K. Gibbs. IPERF. `http://dast.nlanr.net/projects/Iperf/`, 1997.

[26] M. Waidner. Unconditional Sender and Recipient Untraceability in Spite of Active Attacks. In *Proceedings of EUROCRYPT*, pages 302–319, April 1989.

[27] X. Wang and D. S. Reeves. Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Interpacket Delays. In *Proceedings of the* 10^{th} *ACM Conference on Computer and Communications Security (CCS)*, pages 20–29, October 2003.