How Much Anonymity does Network Latency Leak?

Nicholas Hopper, Eugene Y. Vasserman, Eric Chan-Tin
University of Minnesota
200 Union St SE
Minneapolis, MN 55455 USA
{hopper, eyv, dchantin}@cs.umn.edu

ABSTRACT

Low-latency anonymity systems such as Tor, AN.ON, Crowds, and Anonymizer.com aim to provide anonymous connections that are both untraceable by "local" adversaries who control only a few machines, and have low enough delay to support anonymous use of network services like web browsing and remote login. One consequence of these goals is that these services leak some information about the network latency between the sender and one or more nodes in the system. This paper reports on three experiments that partially measure the extent to which such leakage can compromise anonymity. First, using a public dataset of pairwise round-trip times (RTTs) between 2000 Internet hosts, we estimate that on average, knowing the network location of host A and the RTT to host B leaks 3.64 bits of information about the network location of B. Second, we describe an attack that allows a pair of colluding web sites to predict, based on local timing information and with no additional resources, whether two connections from the same Tor exit node are using the same circuit with 17% equal error rate. Finally, we describe an attack that allows a malicious website, with access to a network coordinate system and one corrupted Tor router, to recover roughly 6.8 bits of network location per hour.

Categories and Subject Descriptors

C.2.0 [**Computer Networks**]: General—*Security and protection*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*; E.3 [**Data**]: Encryption

General Terms

Security, Latency, Anonymity, Measurement

1. INTRODUCTION

The goal of every anonymous communication scheme is to allow users to communicate while concealing information about who communicates with whom. The notion of anonymous communication schemes was first introduced by Chaum [5], who proposed sending messages through a "Mix server" that mixes together messages from several senders before forwarding these messages to their destinations, concealing the relationships between senders and receivers. Since then, a wide variety of anonymity schemes have been proposed, yet all practical, deployed schemes rely to some extent on this idea of forwarding messages through "mixing" relays.

Current, widely-used anonymity schemes can be categorized as either high- or low-latency. High-latency systems like Mixmaster and Mixminion [10, 25] deliver messages at a significant delay - around 4 hours, on average - with the goal of ensuring anonymity against a strong adversary that can see all network traffic and control some nodes participating in the anonymity scheme. In order to ensure security against this type of adversary, these schemes implement countermeasures that increase delay, such as pool mixing, and consume additional bandwidth, such as cover traffic. There is a wide range of literature [11, 12, 22] on how to further strengthen such high-latency systems against various types of attacks.

In contrast, *low-latency* protocols such as Tor [13], I2P [21], AN.ON [15], Crowds [31], Anonymizer.com, and various commercial proxy aggregators, actively seek to limit processing delay and bandwidth overhead. Providing low-delay anonymity enables anonymous use of more interesting application services such as remote login and web browsing, but this functionality comes at the cost of reduced anonymity guarantees. In particular, most of these services are easily defeated by a global passive adversary using relatively straightforward attacks such as packet counting [34]. Furthermore, using these same attacks, an adversary that controls fraction f of the nodes in the system can trace fraction f of the connections made to colluding servers and fraction f^2 of all connections running through the system [36]. Thus, these systems focus on offering security against a "local" adversary, such as a small coalition of malicious servers that see only their own network traffic.

A "local" adversary is thus *extremely* limited, since he is unlikely to have access to any traffic of interest before it exits the anonymity service and arrives at his malicious servers. A natural question to ask is: What information, outside of the actual bits of data packets delivered to the adversary, does a low-latency anonymity service leak, and to what extent does this leakage compromise the anonymity offered by the service?

Several recent works have explored the impact of the local adversary's access to information about the timing of

CCS'07, October 29–November 2, 2007, Alexandria, Virginia, USA.

events in a low-latency anonymity scheme, such as packet arrival times. An example of this is the "circuit clogging" attack variant of Murdoch and Danezis [27], which relies on the observation that a sudden increase in the load of a Tor server will increase the latency of all connections running through it. Murdoch and Danezis show how a corrupt Tor node and web server can exploit this property to determine the nodes in a Tor circuit, i.e., the nodes that forward a given connection through the network. In the attack, the corrupt Tor node regularly sends packets on a loop through each Tor server, measuring the time the packets spend in transit. Then when the malicious server wishes to trace a connection, it modulates its throughput in a regular, on/off burst pattern. By correlating the delay at each Tor server against the timing of these burst periods, the attacker learns which nodes are in the circuit. Since the estimated number of Tor users (on the order of 10^5 as of April 2007) is less than the number of possible circuits (on the order of 10^8) seeing two connections that use the same circuit nodes is a strong indicator that the connections are from the same user. Thus at a minimum, timing information can leak the linkage between Tor connections.

In this paper, we make use of a similar observation: malicious servers acting as local adversaries can observe the *network latency* of a connection made over a Tor circuit. While it has been suggested before that this information might be a potential avenue of attack [3], we are not aware of any work reporting on the feasibility of performing an attack using this information, or even suggesting a concrete attack mechanism. As a consequence, it was not known whether leaking this information had any adverse effect on the anonymity provided by schemes like Tor. We address this issue by reporting on three experiments that measure the extent to which this information leakage compromises the anonymity of clients using a low-latency anonymity scheme:

• **Analysis of noise-free anonymity leakage.** Suppose that an anonymity service could impose *no delay at all* on a circuit, so that the only difference between a client connecting to a server normally and over the anonymity service would be that in the latter case, the client's IP address is somehow missing. This would represent the *best possible* case for an attack based solely on round-trip time (RTT) information. We analyzed the publicly available MIT King data set [17], a collection of pairwise RTTs between 1950 Internet hosts, to estimate the average amount of information that is leaked by knowing the RTT between a given host and an unknown host. We found that, on average, knowing the RTT to a host from one known server yields 3.64 bits of information about the host (equivalently, reduces the number of possible hosts from n to $n/2^{3.64} \approx 0.08n$).

Of course, many hosts on the Internet will be essentially indistinguishable by RTTs since they are located on the same subnet; without a more detailed study it is difficult to estimate the number of such equivalence classes. A reasonable estimate would seem to be the number of routable IP address prefixes, currently around $200,000$, or about 2^{18}. Thus on average, we estimate that an Internet host can be uniquely identified, up to RTT equivalence, by knowing its RTT to 5 other (randomly chosen) hosts. (Further work is necessary to more precisely determine the extent to which conditional entropy decreases with each measurement.)

• **A passive linkability attack.** When latency "noise" is introduced in the form of additional delays due to forwarding and mixing with other streams, it is no longer clear how to use latency or RTT information to identify anonymous clients. We observe that even in this scenario, if a client attempts to connect to two malicious servers (or make two connections to the same malicious server) using the same circuit, then the server-client RTTs of these connections (minus the RTT from the last node to the server) will be drawn from the same distribution, whereas other clients connecting to the server will have different RTTs.

Based on this observation, we develop an attack on Tor that allows two colluding web servers to link connections traversing the same Tor circuit. The attack uses only standard HTTP, the most commonly mentioned Tor application layer, and requires no active probing of the Tor network and has very minimal bandwidth requirements. Thus it can be seen as a "lower cost" alternative to circuit clogging.

We report on an implementation and test of this attack using several hundred randomly chosen pairs of clients and randomly chosen pairs of servers from the PlanetLab wide area testbed [6], communicating over the deployed Tor network. Our results suggest that we can classify pairs of connections with an equal error rate of roughly 17%, and the test can be tuned to support a lower false positive or false negative rate.

• **An active client-identification attack.** Finally, we show how latency information can be used to extend the reach of the Murdoch-Danezis clogging attack, allowing a malicious server to take advantage of repeated visits from a client to gradually locate the client, up to RTT equivalence. As with the clogging attack, our attack requires minimal resources – one corrupted Tor server, plus access to a "latency oracle" that can be used to estimate RTTs between Tor servers and nodes in the RTT equivalence class of a suspected client's location – and uses only standard protocols.

We show that a latency oracle can be implemented with a "network coordinate system," [7,8,28] which could be implemented using publicly available resources such as the ScriptRoute [35] service or `traceroute.org`.

We evaluate our attack using over 200 runs with randomly chosen client/server pairs from the PlanetLab wide area testbed, using randomly chosen circuits among the currently deployed Tor nodes (as of Jan./Feb. 2007). Our results suggest that a malicious server with a periodically reloading web page can recover, on average, about 6.8 bits of information about a client's location per hour. Thus a client's RTT equivalence class can be determined in 3 hours, on average.

We stress that both attacks are tested under real-world conditions against the deployed Tor network using a standard protocol (HTTP), and very little has been done to optimize these attacks for speed or accuracy. It is our expectation that we could make improvements in both of these categories by using less widely-supported tools, such as persistent HTTP over Tor. This would improve the performance of the attack, while simultaneously limiting its scope; we leave further investigations along these lines for future work.

These results have serious implications for the design of low-latency anonymity schemes. In particular, they suggest that, without new ideas for path selection, adding delay to a connection may be unavoidable for security considerations. In turn, this has implications for design decisions: for example, if latency must be uniformly high, then TCP tunneling over such services will provide extremely low bandwidth; or if the latency of circuits can be masked with noise in the short term, then circuit lifetimes may need to be shortened.

The remainder of this paper is organized as follows: in section 2, we give an overview of Tor, review the details of the Murdoch-Danezis attack, and survey related work. Section 3 presents the results of our analysis on the MIT King dataset, estimating the average amount of information leaked by the RTT between two nodes. We present details of our passive linking attack and its evaluation in section 4, and more details about our client-identification attack in section 5. Finally, we discuss countermeasures and future work in section 6.

2. BACKGROUND AND RELATED WORK

2.1 An overview of Tor

Tor is a low-latency and bandwidth-efficient anonymizing layer for TCP streams. Its growing popularity and the availability of a test-bed deployment have proven to be a fertile ground for research on implementing and attacking low-delay anonymity schemes.

Tor works similarly to a circuit-switched telephone network, where a communication path, or circuit, is first established, over which all communication during a given session takes place. Anonymity is achieved by establishing that circuit through three nodes: an entry node, an intermediary (middleman), and an exit node. Only the entry node knows the identity of the client contacting it, in the form of its IP address. The middleman node knows the identities of both the entry and exit nodes, but not who the client is or the destination he or she wishes to reach over the circuit. If the Tor server is an "exit" node, which provides a gateway between the Tor network and the Internet, it is responsible for making application-layer connections to hosts on the Internet, and serves as a relay between potentially non-encrypted Internet connections and encrypted Tor traffic. Thus, it knows the destination with whom the client wishes to communicate, but not the identity of the client. In this manner, no single node in the Tor network knows the identities of both communicating parties associated with a given circuit. All communications proceed through this encrypted tunnel.

Circuits are established iteratively by the client, who gets a list of Tor nodes and long-term keys from a directory service, selects a Tor node from that list (preferably one with high uptime and bandwidth), negotiates a communication key, and establishes an encrypted connection. To avoid statistical profiling attacks, by default each Tor client restricts its choice of entry nodes to a persistent set of three randomly chosen "entry guards". The circuit is then extended to additional nodes by tunneling through the established links. Link encryption, using ephemeral Diffie-Hellman key exchange for forward secrecy, is provided by SSL/TLS. To extend the circuit to another Tor node, the client tunnels that request over the newly-formed link.

Traffic between Tor nodes is broken up into cells of 512 bytes each. Cells are padded to that size when not enough data is available. All cells from the client use layered (or "onion") encryption, in that if the client wishes for a message to be passed to `example.com` via Tor nodes A, B, and C (C being the exit node), the client encrypts the message with a key shared with C, then again with a key shared with B, and finally A. The message is then sent over the previously-established encrypted tunnel to A (the entry node). A will peel off a layer of encryption, ending up with a message encrypted to B (note that A can not read this message, as A does not have the key shared between the client and B). A then passes on the message to B, who peels off another encryption layer, and passes the message to C. C removes the final encryption layer, ending up with a cleartext message to be sent to `example.com`. Messages can be any communication that would normally take place over TCP.

Since there is significant cryptographic overhead (such as Diffie-Hellman key exchange and SSL/TLS handshake) involved with the creation and destruction of a circuit, circuits are reused for multiple TCP streams. However, anonymity can be compromised if the same circuit is used for too long, so Tor avoids using the same circuit for prolonged periods of time, giving circuits a client-imposed maximum lifetime[1].

2.2 Attacks against Tor

Timing-based attacks. There have been a number of attacks mentioned in the literature that exploit the low latency of anonymity systems such as Tor. Several of these seem to have first been proposed, without implementation or evaluation, by Back *et al.* [3], including an earlier, more expensive, version of the Murdoch-Danezis clogging attack based on flooding nodes and looking for delay in the connection, and using network delays as a potential method to identify senders.

In [27], Murdoch and Danezis describe an attack that allows a single malicious Tor server and a colluding web server (or other service provider), to identify all three nodes of a Tor circuit used by a client for a given session (ideally, only the exit node's identity should be known to the service provider). However, this system does not identify the client directly, only its entry node into the Tor network[2]. The attack works as follows: when a client connects to the malicious web server, that server modulates its data transmission back to the client in such a way as to make the traffic pattern easily identifiable by an observer. At least one Tor server controlled by the adversary builds "timing" circuits through each Tor server in the network (around 800 as of January/February 2007 [1]). These circuits all have length one, beginning and terminating at the adversarial Tor node. By sending traffic through timing circuits to measure latency, the adversary is able to detect which Tor servers process traffic that exhibits a pattern like that which the attacker web server is generating. Since Tor does not reserve bandwidth for each connection, when one connection through a node is heavily loaded, all others experience an increase in latency. By determining which nodes in the Tor network exhibit the server-generated traffic pattern, the adversary can map the entire Tor circuit used by the client.

Øverlier and Syverson [30] discuss locating Tor *hidden services*. Hidden services allow a server to offer a service anonymously via Tor, by maintaining an open circuit to an "introduction point," which the client contacts through a circuit ending in a "rendezvous node," that the server also contacts through a fresh circuit. Their attack makes use of a malicious client and a single malicious Tor node. The main idea is to make many connections to the hidden server, so that it eventually builds a circuit to the rendezvous point using the malicious Tor node as an entry point. The malicious

[1]This value is configurable. In the latest version (0.1.1.26-alpha), the maximum circuit lifetime is 10 minutes.

[2]The client can be directly identified only if its entry node is also corrupted, but the success of this attack requires corrupting a proportionally large number of Tor nodes.

Tor node uses a simple timing analysis (packet counting) to discover when this has happened.

In another attack against Tor hidden services, Murdoch shows how to identify them based on clock skew [26], enabling us to estimate the load of a given Tor node (as temperature rises when CPU load increases) as well as the rough physical location of the node (as the temperature is generally higher during the day than at night, with a clear pattern visible if clock skew is measured over at least one 24-hour period). This attack may allow us to uniquely map a hidden service to a Tor node, if that node is in a geographically unique location compared to other Tor nodes. More importantly, this attack counters the reserved-bandwidth defense (used to make it harder for Tor nodes to determine other nodes' throughput), as CPU load indirectly measures throughput of a node (based on how busy it is). A node can defend against this by constantly running the CPU at 100%, but this may not be universally acceptable.

Syverson *et al.* [36] suggest that an adversary may de-anonymize any stream for which that adversary controls the entry and exit nodes. The probability of this occurrence in the short term (transient client connections) is $c(c-1)/r^2$, where c is the maximum number of nodes corruptable by the adversary in a fixed period of time, and r is the number of available Tor routers in the network. An adversary can determine if he or she controls the entry and exit node for the same stream by using a number of methods mentioned below, including fingerprinting and packet counting attacks.

Other attacks within Tor's threat model. In [20], Hintz shows how to determine the remote web site that a given stream is connecting to by fingerprinting the pattern of traffic carried by the stream. This attack requires maintaining an up-to-date catalog of website fingerprints, and comparing observed connections against this catalog. The fingerprint may be stable for certain websites where either the format or the content does not change much over time. This attack is particularly detrimental when mounted by a malicious entry node, since it allows for the de-anonymization of the client (who is directly connecting to the entry node) as well as the remote web server.

The packet-counting attack is a less time-precise version of the attacks in [30] – it relies on time intervals as opposed to timestamps. In this attack, discussed in [3, 4, 34, 38], the adversary estimates the load level of a node by measuring the packet flux across that node (the number of packets entering and the number emerging). This attack can be used as a starting point for other attacks mentioned above, such as detecting whether an adversary controls nodes that are part of the same circuit.

Attacks outside the Tor threat model. A well-known class of attacks against anonymity systems – called statistical disclosure, or long-term intersection attacks [9, 24] – also use coarse-grained timing, treating the entire anonymizing network (be it a single mix, a group of mixes, or another system) as a black box, and correlating traffic that enters and exits the system to determine communication patterns. This attack essentially learns about all of the communication relationships between the set of nodes it can monitor. Since Tor is mainly concerned with a "local adversary" that can only monitor the communications of its own nodes, the attack, while a serious consideration, is essentially outside of Tor's threat model.

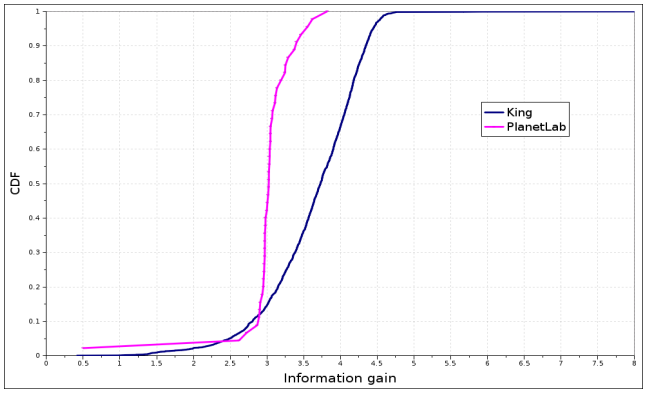

Figure 1: Cumulative distribution of expected information gain from RTT per host, for MIT King data set and PlanetLab nodes.

3. LATENCY WITHOUT NOISE

The possibility of using latency data in traffic analysis has been mentioned several times in previous works, apparently originating in a 2001 paper by Back *et al.* [3]. However, neither this work nor subsequent works seem to have addressed the basic question of *How much information does network latency leak?* Of course the answer is highly dependent on both the network topology – latency in a star topology would leak no information about a host's location – and the protocol in question, since it is conceivable that so much noise is added to the network latency that the signal is undetectable. In order to get an upper bound on the amount of information that can be leaked under the current Internet topology, we measured the amount of information about a host that can be gained given a precise estimate of its RTT to a randomly chosen host. Thus this evaluation represents a "best case" scenario for the adversary wishing to locate clients using latency information.

We performed our analysis on two different data sets. For the general Internet, we used the MIT King data set [17]. King [19] is a method for estimating the latency between two arbitrary Internet hosts, by making recursive queries through their associated nameservers. While this method has some flaws when used to estimate arbitrary latencies, it produces highly accurate estimates of round-trip times between the nameservers. The MIT dataset consists of multiple pairwise RTT measurements between 1950 randomly selected nameservers. Second, because our wide-area experiments use only PlanetLab nodes as clients, we analyzed the amount of information present in the RTTs between PlanetLab nodes.

Our analysis worked as follows: for each RTT we considered, there were several measurements; we calculated an 85% confidence interval for each of these by taking the average plus or minus one standard error. Then for each source host A, we computed the expected number of bits in the RTT to a random destination host B by counting, for each B, the number of hosts C such that the confidence intervals for AB and AC overlapped. Taking this count as N_B and the total number of hosts as N we computed the information gain for AB as $\log_2(N/N_B)$.

The results of our analysis for both data sets are shown in Figure 1. For the King data set, the average number of bits from RTT per host is 3.64, the median is 3.8, and the

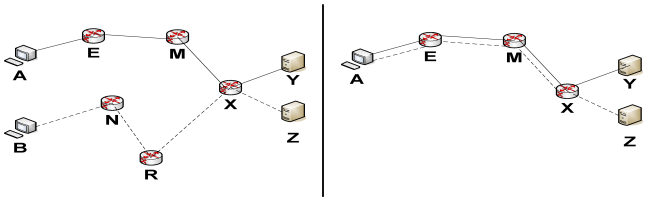

Figure 2: Circuit linking scenario: client A connects via circuit E-M-X to server Y, and client B connects via N-R-X to server Z. Y and Z collude to determine if A-E-M and B-N-R are distinct (left) or identical (right) paths.

10th percentile is 2.8. Thus 90% of Internet hosts leak 2.8 or more bits of location data by their RTT. The information gain from RTT among the PlanetLab nodes is more concentrated, with an average of 3.08 bits from RTT, a median of 3.02, and a 10th percentile of 2.89.

The results also reveal an interesting aspect of the King data set: of the ten hosts whose RTTs give the least information gain, all but two have Chinese domain names; in contrast, the 10 nodes with the highest amount of information per RTT appear to be located primarily in western or central Europe. We speculate that the high information gain for these European nodes is due to the fact that most Internet routes from Europe to Asia must transit through North America, so that there are more possibilities for network latency times to these nodes, while the need to transit one of a few bottleneck links from China adds enough "random" variability to mask the variation in latency between other locations external to China.

4. CIRCUIT LINKING VIA LATENCY

The basic setup of our circuit linking attack is shown in Figure 2. In this scenario, two colluding servers, Y and Z, both accept connections from the same Tor exit node, X. A truly unlinkable anonymity scheme should prevent the servers from being able to distinguish between the case that (a) two distinct clients have made the requests and (b) the same client makes both requests. Conversely, the goal of the servers Y and Z is to determine whether they are communicating with different clients or the same client.

4.1 Attack Description

Our attack works as follows: we assume that Y communicates with client A over a Tor circuit involving nodes E, M, and X, and server Z communicates with client B over a Tor circuit involving nodes N, R, and the same exit node X. If we denote by T_{UV} a random variable that denotes the RTT between nodes U and V, and by T_U a random variable that denotes the "queueing" time at Tor node U, then the idea behind our attack is to take several samples from both $T_{AX} = T_{AE} + T_E + T_{EM} + T_M + T_{MX} + T_X$ and $T_{BX} = T_{BN} + T_N + T_{NR} + T_R + T_{RX} + T_X$. If $A = B$, $E = N$, and $M = R$, then the sample sets should appear to come from the same probability distribution, and if not, they should appear to come from different distributions. The primary obstacles to overcome are how to obtain samples of the random variables T_{AX} and T_{BX}, and, to a lesser extent, what test to use on these samples.

The process of actually sampling the latency of a Tor circuit is somewhat complicated by the fact that Tor is a *proxy* protocol, rather than a *tunneling* protocol: when the exit

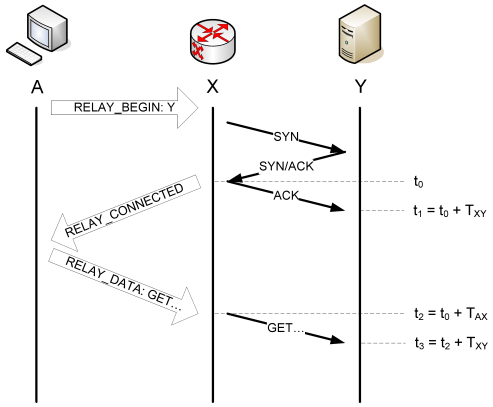

Figure 3: Measuring Tor circuit time without application-layer ACKs: the estimate for T_{AX} is $t_3 - t_1$. We abuse notation and write T_{XY} for the one-way delay from X to Y.

node X receives TCP packets from Y, it acknowledges them immediately, then buffers the results into cells, and relays the cells through the circuit. Thus the usual TCP mechanisms for estimating RTT only estimate the RTT from the server to the exit node, which will not help with our attack. One possible avenue of attack would be to explore application-level protocols that have explicit acknowledgements, such as IRC [29], but since the most widely used application protocol in Tor seems to be HTTP, which does not explicitly support application-level ACKs [16], this would restrict the scope of our attack. Instead we use a less-efficient but more widely-applicable approach targeted at web browsers.

Our attack works as follows: when Y (or Z) gets an HTTP request from a Tor node and decides to attack the connection, it responds with an HTML page with 1000 `<IMG height=1 width = 1 src= ...>` tags, pointing to uniquely named empty image files. This causes most existing browser / privoxy combinations to eventually make 1000 separate connections to Y[3]. For each of these connections, Y will get a SYN packet from X, and send a SYN/ACK packet; this packet is ACKed by X and X sends a "RELAY_CONNECTED" cell to the client. When the client A receives the "RELAY_CONNECTED" cell, it forwards an HTTP "GET" request to X, who forwards the request to Y. The time between the arrivals at Y of the "ACK" and "GET" packets is a sample from T_{AX}. See Figure 3 for an illustration of this procedure.

There are many methods for testing similarity of two sample sets. We used two different tests for our evaluation:

- The *comparison of means* test constructs a confidence interval for the mean of each sample population, under the assumption that the time to traverse a Tor circuit is some fixed time (based on wire speed) plus an exponentially distributed random variable; two sample sets are classified as identical if their confidence intervals overlap. The accuracy of this test depends both

[3]The amount of concurrency varies, but in the Firefox browser, for example, by default only 24 concurrent connection attempts are allowed; thus optimistically, these requests come in 42 "rounds."

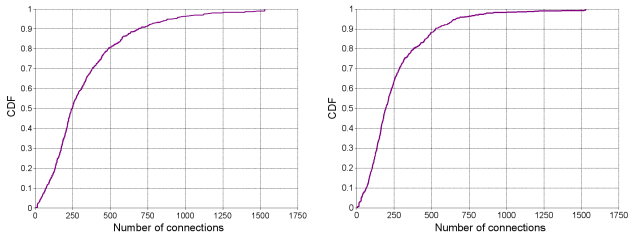

Figure 4: Cumulative distribution of sample size per run for clients A (left) and B (right).

on the degree to which this model is correct, and the width of the confidence interval used.

- The *Kolmogorov-Smirnov*, or K-S, test computes the largest difference in cumulative probability density between two sample sets, and classifies two sample sets as identical if this value is smaller than some rejection parameter. The K-S test is *nonparametric*, i.e. it makes no assumption about the distributions of the sample sets (except that the samples are i.i.d.), and can differentiate distributions based on "shape" and "location", but generally requires more samples than a parametric test with a correct model of the data.

Both tests have a tunable rejection region, giving a tradeoff between false positive and false negative error rates, so no single number characterizes the performance of either test. Thus, we evaluate them using Receiver Operating Characteristic (ROC) curves: each point on a classifier's ROC curve corresponds to the true positive and false positive rates for one setting of the rejection threshold. This curve illustrates the different tradeoffs between false positive and false negative rates for a classifier: a perfect classifier would correspond to the single point in the upper left corner, while a classifier that cannot distinguish between positive and negative examples will result in (a subset of) the line from $(0, 0)$ to $(1, 1)$. This tradeoff is sometimes summarized by calculating area under the ROC curve (AUC), where higher values indicate a "superior" classifier; the perfect classifier has AUC 1, while the nondiscriminating classifier has AUC 0.5. See Fawcett's tutorial [14] for a more comprehensive treatment.

4.2 Evaluation

We tested the effectiveness of this attack using clients and servers from the PlanetLab wide-area testbed. Our evaluation consisted of 641 "runs", where each run performed the following experiment. First, two random PlanetLab nodes were chosen to be the clients A and B, and two random PlanetLab nodes were chosen to be the servers, Y and Z; a random high-bandwidth, high-uptime Tor exit node was chosen for X and each client picked its own entry and middleman nodes. After A and B established their respective Tor circuits, both clients connected to both servers using the `wget` HTTP client, and the servers sampled these circuit times as described in Figure 3. These four circuit times were used in six comparisons: comparing samples from T_{AX} to T_{AY} and T_{BX} to T_{BY} gave two true positives, while comparing samples of T_{AX} to T_{BX}, T_{AX} to T_{BY}, T_{AY} to T_{BX}, and T_{AY} to T_{BY} gave four true negatives. Counting the number of misclassified streams for various threshold values allowed us to calculate false positive and false negative rates.

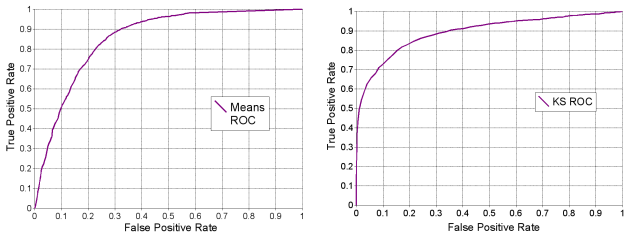

Figure 5: ROC curves for comparison of means tests (left, AUC 0.85) and K-S test (right, AUC 0.89).

One important note is that the current version of Tor recycles a used circuit after 10 minutes. Thus we set our experiments to stop after 10 minutes as well, regardless of whether the run had completed. Figure 4 shows the cumulative distribution function of number of circuit samples for both of the client nodes - the median number of samples after 10 minutes was approximately 200.

Because both tests we used have tunable rejection regions, there is no single statistic that completely summarizes the effectiveness of our classifier. Instead, by varying the rejection region, we produce the Receiver Operating Characteristic (ROC) curves for each of our tests, which summarize the tradeoffs each test supports, in Figure 5. For the simple comparison of means test (left), we find that the test supports an equal error rate of 22% and has a total area under curve (AUC) of 0.85. When dealing with a low base rate of true positives, the K-S test offers a much better tradeoff, supporting, for instance, a 37% false negative rate when the test is tuned to support a false positive rate of 5%, and achieving equal error rate of 17%; the K-S test has a total area under curve (AUC) of 0.89. In contrast, if Tor circuits were unlinkable, we would expect any linking test to have performance similar to the random classifier, which has an ROC curve consisting of a straight line from the origin to $(1, 1)$ and AUC of 0.5.

While these results suggest that latency data compromise the unlinkability of Tor connections to some degree, there is definitely room for improved attacks. To determine if more time efficient methods of sampling circuit RTT would improve performance, we also plotted the ROC curve for K-S tests where both circuits had at least 500 samples, shown in Figure 6. The total number of such tests[4] was 299, with 110 true positives and 189 true negatives. We found that these K-S tests had an equal error rate of 6% and AUC of 0.98. When the rejection region for the K-S test was set to 0.13, the classifier had a false negative rate of 11% with only a single false positive. These results strongly suggest that more efficient methods of obtaining circuit RTT samples will lead to stronger attacks.

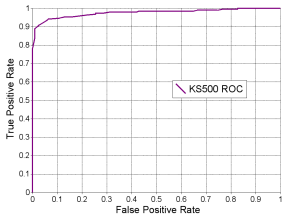

Figure 6: ROC with 500+ samples of both circuits.

5. CLIENT LOCATION VIA LATENCY

The basic scenario of our client location attack is shown in figure 7. In this attack, the adversary consists of three

[4]Recall that each *run* can result in as many as six *tests*.

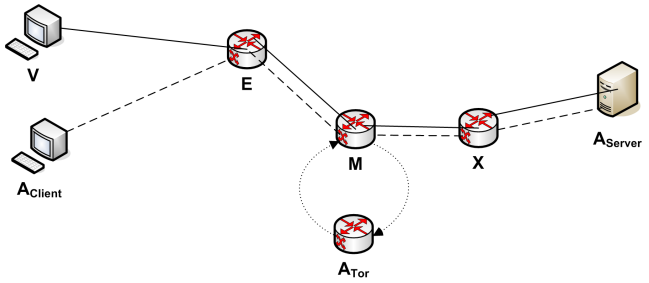

Figure 7: Client location: client V connects to malicious server A via circuit E-M-X; A determines E-M-X and connects to A via E-M-X.

logical entities, A_{Server}, a malicious web server; A_{Client}, a node posing as a Tor client; and A_{Tor}, a corrupted Tor server capable of carrying out the Murdoch-Danezis attack. The attack starts when the "victim" node V connects to A_{Server} over a Tor circuit consisting of nodes E, M, and X. A_{Server} and A_{Tor} collude to carry out the Murdoch-Danezis clogging attack and learn the Tor nodes in the circuit $E - M - X$. Thereafter, A_{Server} and A_{Client} collude to gain information about V's network location. The goal of the attack is, after several repetitions with different circuits, to identify V's network location with increasing precision.

5.1 Attack Description

The basic idea of our client location attack is to try to measure – using a Tor connection – T_{VE}, the RTT between the victim V and the Tor entry node, E. The attacker then estimates, for several candidate victim nodes C, the RTT T_{CE}. Candidates that lie outside the probable range for T_{VE} are discounted, and the attack is repeated. If the fraction of candidates in an iteration that are not discounted is c, then the information gain from that iteration is $-\log_2 c$. After several iterations, the list of remaining candidates should include only nodes with close network proximity to the victim. We now explain how we implement each step of this attack.

Measuring first hop latency. In section 4, we describe our technique for sampling the RTT of an entire Tor circuit, T_{VX}. Since $T_{VX} = T_{VE} + T_E + T_{EM} + T_M + T_{MX} + T_X$, the circuit time contains some information about T_{VE}, but does not directly measure the time of interest. In order to do this, we leverage the information gained in the Murdoch-Danezis attack: specifically, when V connects to A_{Server} via Tor, we assume that A_{Server} and A_{Tor} collude to discover the circuit nodes $E - M - X$ that V uses for the connection. Initially, this attack will only reveal the nodes in the circuit rather than their order, but since any given client uses only three entry nodes, and as the server, A_{Server} knows which node is the exit node, after several iterations it will be easy to infer the circuit order; before such time, the attacker can carry out the attack under both possible orderings and then eliminate the incorrect data later.

Given this information, A_{Client} can open a connection to A_{Server} using the *same* circuit nodes $E - M - X$. We measure the RTT of these connections as well, obtaining several samples from both T_{VX} and T_{AX}. Using these samples as a basis for estimating the "true" circuit times T_{VX} and T_{AX}, along with knowing T_{AE}, the RTT between A_{Client} and E, allows us to estimate T_{VE} by $T_{VX} - T_{AX} + T_{AE}$. We found that Tor circuit times carry enough noise, even with hundreds of samples, that the difference in mean measurements was not very reliable at predicting T_{VE}; however, taking the difference in minimum measurements for each circuit was a fairly reliable estimate.

Estimating candidate RTTs. Once we have estimated the RTT from the victim to the Tor entry node E, the next step is to compare this measurement to the RTT between candidate nodes and E. If we control either the candidate or E, we could compute this directly via `ping`, but doing so would make it easy for us to determine the victim, and is outside the Tor threat model. Thus for the attack to work, we need a method to obtain (or at least estimate) the RTT between two hosts without the explicit cooperation of either. Our attack measures this quantity via network coordinates.

Network coordinate systems were originally introduced in the context of peer-to-peer networks, for predicting which hosts will provide better routing or download service. The basic idea behind such systems is for each node to measure its RTT to several other nodes; using these RTTs, the entire network is embedded into a coordinate space such that given the coordinates of two nodes it is possible to predict the RTT between them. A number of such systems exist [7, 8, 23, 28], using various coordinate systems and embedding algorithms. We chose to use the Vivaldi [8] embedding algorithm, with four-dimensional Euclidean coordinates, due mainly to ease of implementation. The primary disadvantage of using network coordinates is that in order to be accurate without the cooperation of the candidate nodes, several nodes must be used for the service; however, several freely accessible resources provide RTT measurements from a group of hosts to arbitrary Internet hosts, including ScriptRoute [35] and `traceroute.org`.

Several alternate possibilities exist for the implementation of this step, that we did not evaluate empirically. One example is the King technique [19], which measures the latency between hosts A and B by asking the name server responsible for A's reverse DNS entry to do a recursive lookup for B's reverse DNS entry; Gummadi *et al.* [19] report that this technique has accuracy competitive with the GNP [28] network coordinate system and found that over 90% of name servers will carry out such recursive queries. Another possibility that we did not empirically evaluate is "asking" the entry node E to ping the candidate nodes by trying to extend a circuit from E to a service other than Tor running on a (node proximal to a) candidate node. If the attacker runs the same service on a corrupted node D and asks E to extend a circuit to D at the same time, then the time difference between error messages for the two requests should be a good estimator for the difference in RTT.

Eliminating candidates. Once we have estimated the true victim's RTT to E and the RTT between each of the candidate locations and E, we must decide for each candidate location whether it is consistent with the estimated T_{VE}. There is an inherent tradeoff in setting the threshold for what nodes to include: if too many candidates are included, the attack will take longer to complete, but if too few are included we could reject the true location of the victim due to noise in the estimated RTTs. Our approach is to construct an 85% confidence interval for T_{VE} (computed under the assumption that Tor circuit times are distributed according to an exponential distribution added to a mini-

	V	A_{Client}
Number of Runs	216	216
Connections/Run	685	1022
RTT Mean/Stdev (ms)	5078.24/4305.96	3817.2/1933.38

Table 1: Basic run statistics

mum network latency) and "conditionally" reject candidate locations with estimated latencies outside of the confidence interval. Locations are only "conditionally" rejected in that we continue to consider these candidate locations in later runs, and finally compute the most likely locations as those that fall within the confidence interval most often. In our empirical evaluations, we know the true location and when it is conditionally rejected we consider the run to give us no information gain, so that we do not overstate the information gain per experiment.

5.2 Evaluation

We measured the effectiveness of the client location attack (in terms of information gain per unit time) by performing an experiment on the PlanetLab wide-area testbed, during January and February 2007. The data collected from the experiment consisted of 216 runs. At the start of the experiment, two PlanetLab nodes were randomly chosen to play the roles of A_{Server} and A_{Client} as in Figure 7. For each run, a new victim node V was randomly selected from a set of about 100 North American PlanetLab nodes, and V and A_{Client} both built identical Tor circuits using three nodes selected randomly from around 60 high-uptime and high-bandwidth onion routers. V and A_{Client} both attempted to download 1500 1×1 images from A_{Server} over this circuit, with the experiment cut off after 10 minutes. Network coordinates were established by using 72 PlanetLab nodes to ping all other PlanetLab nodes (704 at the start of the experiment) and the entry node in the Tor circuit. As with the previous evaluation, A_{Server} recorded all connections from the Tor exit node at the network level, and to simplify distinguishing between the attacker and victim streams, different ports were used by each[5]. At the conclusion of a run, the number N of PlanetLab nodes that were candidate client locations was computed, and the information gain was computed as 0 if the true client was not included, and $\log_2(704/N)$ otherwise.

Basic statistics about the experimental results are shown in Table 1, and the cumulative distribution of information gain is shown in Figure 8. In terms of basic statistics, there is a large discrepancy in average connections per run between the victim and attacker circuits; we speculate that this is due to a lighter-than-average load on the PlanetLab host we chose for A_{Client}. In terms of information gain, the average number of bits obtained per run was 0.68, with a standard deviation of 0.86 and standard error 0.059. To assess whether the conditional information remains stable for additional runs, we also collected a set of 5 runs using a single client node V. Among these 5 runs, the average conditional information gain from a second run was 0.62 bits, with a standard error of 0.14. Translated to bits per unit time, the statistics suggest that we can recover 4.08 ± 0.702 bits of client location per hour.

[5]The port numbers – 5190 for the victim and 6667 for the attacker – were chosen arbitrarily among those commonly allowed by Tor exit nodes.

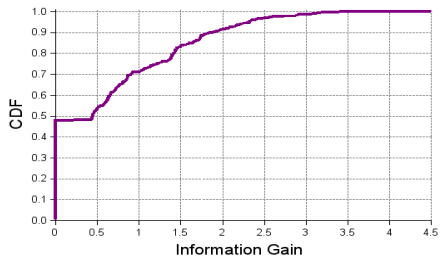

Figure 8: Cumulative distribution of information gain.

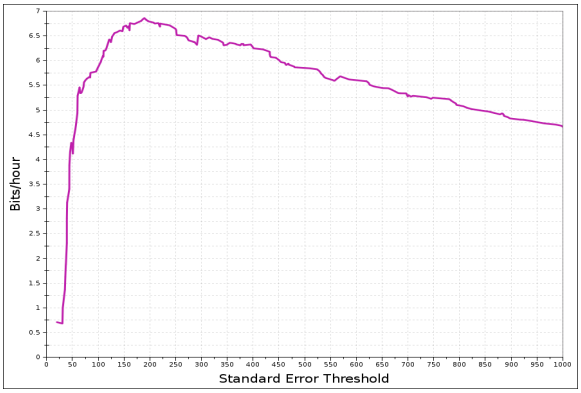

Figure 9: Expected bits per hour vs. 100-connection standard error threshold.

On closer examination of the data, we found that 100 out of the 216 runs yielded 0 bits of information about the client location. This includes the expected 30 runs in which the true client latency did not fall within the 85% confidence interval, but also 70 runs in which *no candidate locations were eliminated*. In these cases, the variability of the Tor circuit RTT was so high that every PlanetLab node's RTT to the entry node was within the confidence interval for T_{VE}. We hypothesized that it might be possible to detect early on that a run would be bad in this sense and discontinue the attack, dedicating the remaining time to a different client.

To test this, we looked at the correlation between standard error of a run after measuring the first 100 connections, which on average required less than 1 minute to collect, and after the run was completed. We found that standard error after 100 connections was an excellent predictor of ending standard error (coefficient 0.96, intercept -199.85, p-value 0.014). Thus, we can increase the bits per unit time obtained through our attack by setting a threshold value T such that, if the first 100 connections have standard error at least T, the attack is stopped with no output.

Setting the value of T requires optimizing a tradeoff between the probability of finding a good run (one where the standard error after 100 connections is less than T) and the information gain from a good run. Low values of T will cause an attacker to abort early on too many runs, while high values of T will cause the attacker to waste time on too many useless runs. Figure 9 shows the tradeoff between the value of T and bits/hour for our data set. Although the exact tradeoff curve will likely differ somewhat based on the distribution of candidate-to-entry node RTTs, we found

that setting the "early abort" threshold to roughly 200ms provided the best tradeoff for our data set. Intuitively, this makes sense: trace-based measurement studies suggest that 70% of Internet connections have median RTTs less than 200ms, and 80% have median RTTs less than 400ms [2], so a circuit with such a high standard error will almost always yield less than $-\log_2 0.8 = 0.3$ bits of information gain. In our measurements, 153 of 216 runs – about 70% – had a standard error greater than 200ms after the first 100 connections, and 115/216 runs (53%) had "early" standard error at least 400ms. Setting the threshold to this latter value yielded an estimated information rate of 6.25 bits per hour of work.

6. DISCUSSION

Limitations. There are several limitations in the attack as currently described. The most serious of these is the limited data on *conditional* information gain, that is, we cannot conclusively evaluate, from our data, how much additional information each run of an attack provides. This is due in part to limitations of our experimental method, which did not re-use clients; thus a "longitudinal" study is needed to more accurately assess conditional information gain. Another reason we could not accurately assess conditional information is due to the somewhat coarse-grained handling of information: our measure of information gain essentially always assumes a uniform distribution over "plausible" clients, so that repeated measurements in which a client is plausible but unlikely (for example, lies within two standard errors of our estimated RTT but not one) do not increase the information as measured by our experiment.

Another limitation is that our client location attack assumes that a user repeatedly accesses a server from the same network location. This assumption may sometimes be invalid in the short term due to route instability, or in the long term due to host mobility. It seems plausible that the attack can still be conducted when circuits originate from a small set of network locations, such as a user's home and office networks, but the attack would be of little use in case of more frequent changes in network location.

Other Applications and Extensions. We evaluated our attacks in the context of the Tor anonymity scheme, but we expect that they should be generalizable to other low-delay anonymity protocols. Indeed, we expect that single-hop proxy services will leak more information about the client-proxy RTT, allowing fairly precise linking attacks, although the strength of the client location attack will be somewhat diminished against services that have a single proxy server location. We also expect that peer-to-peer designs such as Crowds [31], MorphMix [32] and I2P [21], with lightly-loaded relays and multiple entry points, would yield cleaner RTT measurements, allowing the attacker to locate clients with higher precision. We are uncertain how the attacks presented here will interact with low-delay mix cascades such as `AN.ON`; in principle some network latency information should leak but we lack empirical data on the distribution of the noise in this scheme.

As we previously mentioned, we believe the speed and precision of our attacks can be increased by using a different measurement procedure when appropriate application-layer protocols are utilized. Examples of protocols that can be exploited to yield application layer acknowledgements include persistent HTTP [16], IRC [29], and SIP [33]. There is likewise still room for evaluation of alternative methods of implementing RTT oracles, and perhaps for a more sophisticated testing procedure that avoids the expense of querying the RTT oracle for every pair of Tor entry node and candidate location. Finally, it would be interesting to study the impact of various Tor parameters on our attacks, such as circuit lifetime, circuit length, and path selection.

Our attack may also be applicable to a recently proposed defense mechanism for hidden services, although it has not been tested. In particular, Øverlier and Syverson [30] have described an attack on Tor hidden services that exploits the ability to make many requests to a hidden service, so that eventually the hidden service connects to a malicious Tor router as the first hop. They recommend using a small set of trusted "entry guards" as first hops to prevent the attack. However, using essentially the same techniques, a malicious Tor node and hidden service client should be able to recognize when it is the second hop router, and obtain very precise estimates of the hidden server's RTT to each of its guard nodes. These estimates can be compared against candidate locations as in our client location attack, and if there are sufficiently few and widely spread candidates compared to the number of entry guards, it should be possible to locate the hidden server. Thus one layer of entry guards should not be considered sufficient to protect a hidden server's location.

Finally, several systems have recently been developed to geolocate an Internet client given its RTTs from a set of landmark nodes [18, 37]. In cases where candidate clients cannot be associated with IP addresses, it may be possible to apply these techniques to our attack, leaking information about the client's physical location.

Mitigation. There are a number of techniques and best practices that can reduce the attacker's probability of success in the client location attack. For example, onion routers can minimize the success probability of the Murdoch-Danezis attack by allocating a fixed amount of bandwidth to each circuit, independent of the current number of circuits, and doing "busy work" during idle time; this may be an undesirable tradeoff between anonymity and efficiency but will prevent the client location attack from succeeding. Outside of such considerations, Tor nodes can prevent being used as RTT oracles by refusing to extend circuits to nodes that are not listed in the directory; they can drop ICMP ECHO_REQUEST packets in order to raise the cost of estimating their network coordinates; and if Tor node administrators have control over their DNS or reverse DNS hosts, they can ensure that recursive lookups from "outside" nodes are disabled. Tor clients and their network administrators can likewise drop ping packets and deny other attempts to learn their network coordinates to the necessary accuracy.

Both client location and circuit linking can be prevented by adding sufficient delays to make the RTT and timing characteristics of Tor servers independent of the underlying network topology; this can be accomplished by delaying the forwarding of data at the client. Alternatively, we can note that in our evaluation, about half of the circuits we sampled already had enough timing noise to essentially defeat the client location attack. Given the limited time period over which a Tor circuit is available for sampling, it may be an effective countermeasure for each node to introduce high-variance random delays in outgoing cells. Selecting delays from an identical distribution at each Tor node would also make the timing distributions from different circuits look more alike, possibly thwarting the circuit-linking attack.

Of course, if the only way to thwart attacks based on latency and throughput is to add latency and restrict throughput, this would have serious implications for the design of low-latency anonymity systems and the quality of anonymity we can expect from such schemes. We believe that our attacks are effective enough to motivate searching for other possible countermeasures. One interesting possibility is to make the Tor path selection algorithm latency-aware, by incorporating some notion of network coordinates into directory listings. Clients could then construct circuits with the explicit goal of having an RTT close to one of a small number of possibilities. Doing so could help reduce the high average circuit RTTs we observed (5 sec), reduce the effectiveness of latency-based attacks, and allow clients to explicitly trade-off some anonymity for better efficiency. However, more research is clearly needed to understand the security implications of such an approach.

Acknowledgments. The authors wish to thank Yongdae Kim, Jon McLachlan, Cat Okita, Ivan Osipkov, and Peng Wang for helpful comments and discussions about this work. This research was partially supported by the National Science Foundation under CAREER grant CNS-0546162.

7. REFERENCES

[1] TOR (the onion router) servers. `http://proxy.org/tor.shtml`, 2007.

[2] Aikat, J., Kaur, J., Smith, F. D., and Jeffay, K. Variability in TCP round-trip times. In *IMC '03: Proc. 3rd ACM SIGCOMM conference on Internet measurement* (New York, NY, USA, 2003), pp. 279–284.

[3] Back, A., Möller, U., and Stiglic, A. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proc. Information Hiding Workshop (IH 2001)* (April 2001), LNCS 2137, pp. 245–257.

[4] Blum, A., Song, D., and Venkataraman, S. Detection of interactive stepping stones: Algorithms and confidence bounds. *Proc. 7th Intl Symposium on Recent Advances in Intrusion Detection (RAID)* (2004).

[5] Chaum, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 24*, 2 (1981), 84–88.

[6] Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev. 33*, 3 (2003), 3–12.

[7] Costa, M., Castro, M., Rowstron, A., and Key, P. PIC: Practical internet coordinates for distance estimation. In *ICDCS '04: Proc. 24th Intl. Conf. on Distributed Computing Systems* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 178–187.

[8] Dabek, F., Cox, R., Kaashoek, F., and Morris, R. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04* (New York, NY, USA, 2004), ACM Press, pp. 15–26.

[9] Danezis, G. Statistical disclosure attacks: Traffic confirmation in open environments. In *Proc. Security and Privacy in the Age of Uncertainty, (SEC2003)* (Athens, May 2003), IFIP TC11, Kluwer, pp. 421–426.

[10] Danezis, G., Dingledine, R., and Mathewson, N. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *SP '03: Proc. 2003 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2003), IEEE Computer Society, p. 2.

[11] Díaz, C., and Serjantov, A. Generalising mixes. In *Proc. 3rd Privacy Enhancing Technologies workshop (PET 2003)* (March 2003), R. Dingledine, Ed., Springer-Verlag, LNCS 2760.

[12] Dingledine, R., et al. Anonymity bibliography. `http://freehaven.net/anonbib`, 1999 – 2007.

[13] Dingledine, R., Mathewson, N., and Syverson, P. F. Tor: The second-generation onion router. In *Proc. 13th USENIX Security Symposium* (August 2004).

[14] Fawcett, T. An introduction to ROC analysis. *Pattern Recogn. Lett. 27*, 8 (2006), 861–874.

[15] Federrath, H., et al. JAP: Java anonymous proxy. `http://anon.inf.tu-dresden.de/`.

[16] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. IETF RFC 2616: Hypertext transfer protocol – HTTP/1.1. `http://www.ietf.org/rfc/rfc2616.txt`, 1999.

[17] Gil, T. M., Kaashoek, F., Li, J., Morris, R., and Stribling, J. The "King" data set. `http://pdos.csail.mit.edu/p2psim/kingdata/`, 2005.

[18] Gueye, B., Ziviani, A., Crovella, M., and Fdida, S. 2006. Constraint-based geolocation of internet hosts. *IEEE/ACM Trans. Netw.* 14, 6 (Dec. 2006), 1219-1232.

[19] Gummadi, K. P., Saroiu, S., and Gribble, S. D. King: estimating latency between arbitrary Internet end hosts. *Proc. 2nd SIGCOMM Workshop on Internet measurement* (2002), 5–18.

[20] Hintz, A. Fingerprinting websites using traffic analysis. In *Proc. 2nd Privacy Enhancing Technologies workshop (PET 2002)* (April 2002), R. Dingledine and P. Syverson, Eds., Springer-Verlag, LNCS 2482.

[21] jrandom, et al. I2P. `http://www.i2p.net/`, 2007.

[22] Kesdogan, D., Egner, J., and Büschkes, R. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proc. Information Hiding Workshop (IH 1998)* (1998), Springer-Verlag, LNCS 1525.

[23] Ledlie, J., Gardner, P., and Seltzer, M. Network coordinates in the wild. In *Proc. 4th USENIX Symposium on Network Systems Design and Implementation* (April 2007).

[24] Mathewson, N., and Dingledine, R. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proc. 4th Privacy Enhancing Technologies workshop (PET 2004)* (May 2004), vol. 3424 of *LNCS*, pp. 17–34.

[25] Moeller, U., Cottrell, L., Palfrader, P., and Sassaman, L. IETF draft: Mixmaster protocol version 2. `http://www.ietf.org/internet-drafts/draft-sassaman-mixmaster-03.txt`, 2005.

[26] Murdoch, S. J. Hot or not: Revealing hidden services by their clock skew. *13th ACM Conference on Computer and Communications Security (CCS)* (2006).

[27] Murdoch, S. J., and Danezis, G. Low-Cost Traffic Analysis of Tor. In *SP '05: Proc. 2005 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 183–195.

[28] Ng, T. E., and Zhang, H. A network positioning system for the Internet. *Proc. USENIX Conference* (2004).

[29] Oikarinen, J., and Reed, D. IETF RFC 1459: Internet relay chat protocol. `http://www.ietf.org/rfc/rfc1459.txt`, 1993.

[30] Øverlier, L., and Syverson, P. Locating Hidden Servers. In *SP '06: Proc. 2006 IEEE Symposium on Security and Privacy (S&P'06)* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 100–114.

[31] Reiter, M. K., and Rubin, A. D. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security 1*, 1 (1998), 66–92.

[32] Rennhard, M., and Plattner, B. Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection. In *WPES '02: Proc. 2002 ACM workshop on Privacy in the Electronic Society* (New York, NY, USA, 2002), ACM Press, pp. 91–102.

[33] Rosenberg, J., et al. SIP: Session Initiation Protocol. IETF RFC 3261, `http://tools.ietf.org/html/rfc3261`, 2002.

[34] Serjantov, A., and Sewell, P. Passive attack analysis for connection-based anonymity systems. In *Proc. ESORICS 2003* (October 2003).

[35] Spring, N., Wetherall, D., and Anderson, T. Scriptroute: A public Internet measurement facility. *USENIX Symposium on Internet Technologies and Systems (USITS)* (2003), 225–238.

[36] Syverson, P., Tsudik, G., Reed, M., and Landwehr, C. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies: Proc. Workshop on Design Issues in Anonymity and Unobservability* (July 2000), H. Federrath, Ed., Springer-Verlag, LNCS 2009, pp. 96–114.

[37] Wong, B., Stoyanov, I., and Sirer, E. G. 2006. Geolocalization on the internet through constraint satisfaction. In *Proc. USENIX WORLDS 2006*.

[38] Wright, M., Adler, M., Levine, B. N., and Shields, C. Defending anonymous communication against passive logging attacks. In *Proc. 2003 IEEE Symposium on Security and Privacy* (May 2003).