How Much Anonymity does Network Latency Leak?

NICHOLAS HOPPER
EUGENE Y. VASSERMAN
ERIC CHAN-TIN
University of Minnesota - Twin Cities

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. We present two attacks on low-latency anonymity schemes using this information. The first attack 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 high confidence. The second attack requires more resources but allows a malicious website to gain several bits of information about a client each time he visits the site. We evaluate both attacks against two low-latency anonymity protocols – the Tor network and the MultiProxy proxy aggregator service – and conclude that both are highly vulnerable to these attacks.

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 [Chaum 1981], 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 [Moeller et al. 2005; Danezis et al. 2003] deliver messages after 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 [Kesdogan et al. 1998; Díaz and Serjantov 2003; Dingledine et al. 2007] on how to further strengthen such high-latency systems against various types of attacks.

In contrast, *low-latency* protocols such as Tor [Dingledine et al. 2004], I2P [jran-

dom et al. 2007], AN.ON [Federrath et al. 2006], Crowds [Reiter and Rubin 1998], 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 [Serjantov and Sewell 2003]. 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 [Syverson et al. 2000]. 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 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 [2005], 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 June 2008) 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 low-latency anonymous connection. While it has been suggested before that this information might be a potential avenue of attack [Back et al. 2001], we are not aware of any work (other than our preliminary work [Hopper et al. 2007]) 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 a series of experiments that measure the extent to which this information leakage compromises the anonymity of clients

using a low-latency anonymity scheme:

• **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 not clear how to use latency or round trip time (RTT) information to identify anonymous clients. We observe however, that if a client attempts to connect to two malicious servers (or make two connections to the same malicious server) using the same anonymous connection, 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 requires no active probing of the Tor network and has very minimal bandwidth requirements. It can use either standard HTTP, the most commonly mentioned Tor application, or persistent HTTP for higher accuracy. Thus it can be seen as a "lower cost" alternative to circuit clogging.

We report on an implementation and test of this attack in three scenarios: Using several hundred randomly chosen pairs of clients and randomly chosen pairs of servers from the PlanetLab wide area testbed [Chun et al. 2003], communicating over the deployed Tor network with the Privoxy proxy, our results suggest that we can classify pairs of connections with an equal error rate of roughly 17%. Using many of the same nodes but with the more sophisticated Polipo proxy interface to Tor, our results suggest that we can classify pairs with equal error rate of only 8%. Finally, we also test the attack against a single-hop open proxy service, and find that Tor is more vulnerable to this attack.

• **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 RTT information.

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. As a preliminary study, we performed a large scale survey that measured the latency to nearly 14000 servers with distinct routable IP address prefixes. Our results show that discovering the RTT between an unknown host and a random Internet host yields roughly 3.5 bits of information about the network location of the unknown host, or equivalently, reduces the number of likely hosts by a factor of $2^{3.5} \approx 11.3$. Learning additional RTTs yields increasingly less information, but our data shows that with 9 RTTs, at most 4 of our 14000 network locations will be probable matches, and on average less than 2.

• **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," [Ng and Zhang 2004; Dabek et al. 2004; Costa et al. 2004] which could be implemented using publicly available resources such as the ScriptRoute [Spring et al. 2003] service or `traceroute.org`.

We evaluate our attack using 995 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 Feb.-Mar. 2008). Our results suggest that a malicious server with a repeating visitor can uniquely identify a client's network location after an expected 50 visits to the website. If the client simply connects to a periodically reloading web page, this translates into an attack that locates the client in roughly 8 hours, on average.

We also evaluate our attack against a single-hop TCP proxy service to isolate the effects of encryption and multiple-hop relays. Our results suggest that on average 41 repeat visits via this service would suffice to uniquely identify a client's network location.

We stress that both of our 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 our methodology for measuring the latency of an anonymous connection. We present details of our passive linking attack and its evaluation in section 4. Section 5 presents the results of our analysis of information leakage by RTT data, estimating the average amount of information leaked by the RTT between two nodes. Section 6 discusses our client location attack against a single-hop proxy, while Section 7 extends the attack to Tor, and Section 8 discusses additional details about implementing our attack without direct access to RTT data. Finally, we discuss countermeasures and future work in section 9.

2. BACKGROUND AND RELATED WORK

2.1 The MultiProxy service

As a means to test the wider applicability of our techniques, beyond those reported in [Hopper et al. 2007], as well as isolate the effects of using a proxy rather than

directly accessing a server, we tested our attacks against both the Tor anonymity scheme, and the MultiProxy service. MultiProxy is a simple application that downloads a list of open TCP proxies from `multiproxy.org`. Then whenever the user requests a connection to server S, MultiProxy chooses one of the proxies on its list and connects to S through the proxy. We did not evaluate the MultiProxy application directly, opting for the simpler route of downloading the proxy list and directly choosing proxies from the list when needed by one of our tests.

2.2 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. [1] 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 list of three randomly chosen "entry guards". [2] 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

[1] Note that this analogy is not perfect: Tor's signalling is in-band, whereas traditional telephony networks have out-of-band signalling circuit construction

[2] In fact, each node may have more than three entry guards, to resist denial of service attacks; if less than three of its entry guards are available, a client finds a new guard node and appends it to the end of the list; first hops are always chosen randomly among the first three currently available guards on the list. Our attack does not depend on this behavior.

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 re-using the same circuit for prolonged periods of time, giving circuits a client-imposed maximum lifetime, after which new streams will not use the circuit.[3] We note, however, that existing streams are not migrated across circuits and may continue to use a circuit beyond this lifetime.

2.3 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.*[Back et al. 2001], 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.

Murdoch and Danezis [2005] 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[4]. 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 2100 as of June 2008 [tor 2008]). 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

[3]This value is configurable. In the version used in our tests – 0.1.1.26 – and all versions up to 0.2.0.26 released in May 2008, the maximum circuit lifetime is 10 minutes.

[4]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.

server-generated traffic pattern, the adversary can map the entire Tor circuit used by the client.

Øverlier and Syverson [2006] 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 Tor node uses a simple timing analysis (packet counting) to discover when this has happened.[5]

In another attack against Tor hidden services, Murdoch [2006] shows how to identify them based on clock skew, 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.* [2000] 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. Hintz [2002] 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 [Øverlier and Syverson 2006] – it relies on time intervals as opposed to timestamps. In this attack, discussed in [Serjantov and Sewell 2003; Wright et al. 2003; Back et al. 2001; Blum et al. 2004], the adversary estimates the load level of a node by measuring

[5]Tor introduced "guard nodes" for hidden services to mitigate this attack.

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 [Danezis 2003; Mathewson and Dingledine 2004] – 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.

3. MEASURING LATENCY THROUGH A PROXY

To validate the wide applicability of our attacks, we evaluated them against two low-latency anonymity networks: the MultiProxy proxy aggregator service, and Tor. Both of these networks slightly complicate the issue of measuring client to server latency because they are *proxy* rather than *tunneling* protocols: when the proxy (Tor exit node) X receives TCP packets from server Y, it acknowledges them immediately, then relays the data to the client. Thus the usual TCP mechanisms for estimating RTT only estimate the RTT from the server to the proxy, 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 [Oikarinen and Reed 1993], but since the most widely used application protocol in Tor seems to be HTTP, which does not explicitly support application-level ACKs [Fielding et al. 1999], this would restrict the scope of our attack. Instead we use a less-efficient but more widely-applicable approach targeted at web browsers.

3.1 General, inefficient procedure

Our basic attack works as follows: when server Y gets an HTTP request from a proxy 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 browsers to eventually make 1000 separate connections to Y[6]. For each of these connections, Y will get a SYN packet from the proxy X, and send a SYN/ACK packet; this packet is ACKed by X, and X notifies the client. (Over Tor, X sends a "RELAY_CONNECTED" cell to the client). When the client A receives the notification 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}, the round trip time between the client and the proxy. See Figure 1 for an illustration of this procedure in the context of Tor.

[6]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."

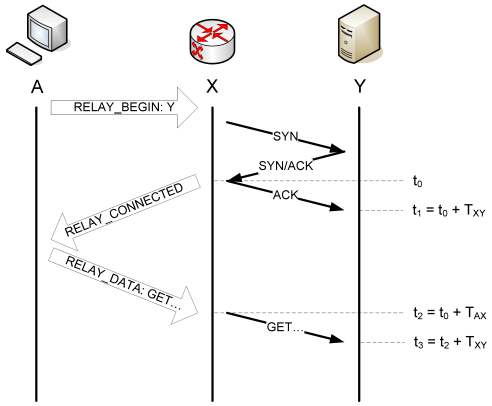


Fig. 1. Measuring Tor circuit time without application-layer ACKs: the estimate for T_{AX} is $t_3 - t_1$. We abuse notation and write T_{XY} for the one-way delay from X to Y.

3.2 A more efficient procedure

The previous method of serially measuring the latency of a circuit by using either an anonymity protocol-specific property of connection setup or measuring the time between serving a web page and getting a request for an embedded object, is very general in applicability but not very time-efficient: if the round-trip time of an anonymous connection is t seconds, then collecting n measurements requires nt seconds. This can limit the sample size quite severely in some instances: for example, Tor "recycles" circuits after 10 minutes, so a high-latency circuit might provide only a few dozen measurements before being recycled. In [Hopper et al. 2007] we used this method exclusively and found many such circuits; analysis excluding these experiments suggested that a more efficient timing method would lead to better results.

If we design our own application-level protocol, this can be achieved by pipelining. In addition to the general method described above, we implemented a very simple "application" in which the server periodically sends timestamps to the client, who echoes the stamps back to the server. If the interval between timestamps is ι, this allows collecting n measurements from a circuit with delay t in time $t + n\iota$. We used $\iota = 0.5$ seconds, allowing us to easily collect several hundred samples of even very slow Tor circuits.

At this point, a natural question to ask is whether this limits the applicability of the attacks we describe. After all, most Tor users will not access malicious websites using the "timestamp echo" application. However, many Tor users will connect to websites using either Firefox over the simple proxy interface to Tor, or an arbitrary web client over the more sophisticated Polipo [Chroboczek 2008] proxy. This is relevant because both Firefox and Polipo support concurrent connections and persistent HTTP.

Persistent HTTP combined with the *redirect* functionality in HTTP can be "abused" in order to provide a nearly identical functionality to our timestamping application as follows. The client's first request is answered with links to k

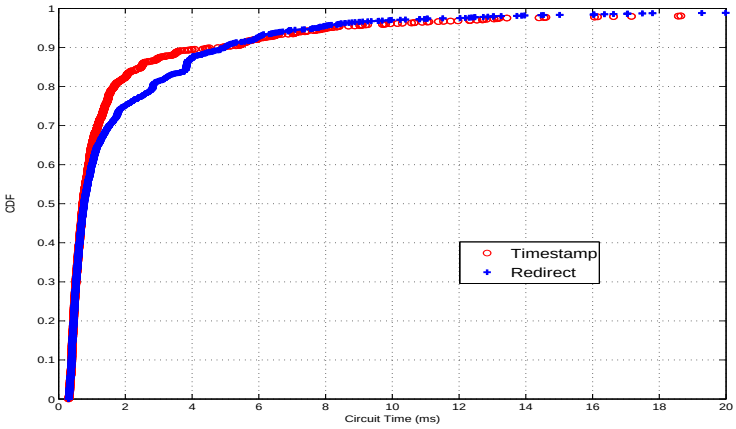


Fig. 2. CDF of circuit times measured by timestamping and redirection.

uniquely-named images with size 1×1, as before. Each image, however has a distinct DNS name for the server, so that the exit node must open k persistent HTTP connections to the server. When the proxy requests one of these images, the server responds with an HTTP/1.1 301 response – "Object Permanently Moved" – with a new URL on the same server encoding the current timestamp. This is followed by 19 additional 301 messages, at regular intervals. Each of these messages is an appropriate response to a GET request for the previous timestamped message. Although it is not required that a persistent HTTP client should read these subsequent messages and correctly interpret them, as long as the interval between messages is longer than the client's time to process the previous redirect and emit a new request then it will appear to the browser as if the circuit simply has very low latency. Thus we would expect the browser to process each of these "301 responses" as soon as it arrives and send a GET request for the new location right away.

We tested and confirmed that both the persistent Firefox browser (version 2.0.0.1) and the non-persistent command-line client `wget` over the Polipo proxy (version 1.0.4) have this behavior. To confirm that measurements collected with our timestamp application have similar statistical properties to measurements collected via redirection, we performed 100 runs of the following experiment. In each experiment, a PlanetLab node, acting as the client, constructs a Tor circuit and simultaneously connects to our server over HTTP and the timestamp application. The latency of each connection is measured at most 20 times, and then the connection is closed. In all, the timestamping application collected 1649 circuit times and the HTTP redirection server collected 1916 circuit times. The cumulative distributions of these samples are shown in figure 2. We compared 100 random points from each of these data sets using an exact two-sample Kolmogorov-Smirnov test [Panchenko 2006]; the p-value given the null hypothesis was 0.797, meaning that if the two samples are IID from identical distributions, we would expect to see samples as divergent

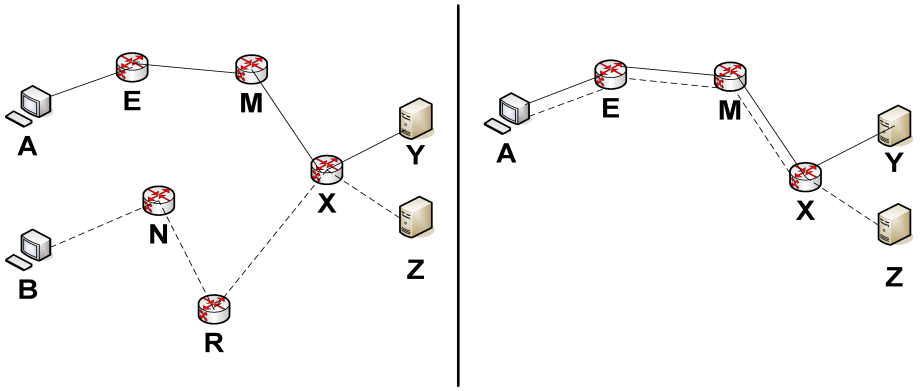


Fig. 3. Circuit linking scenario: client A connects via circuit E-M-X to server Y, and client B connects via N-R-X to server Z. Y and Z collude to determine if A-E-M and B-N-R are distinct (left) or identical (right) paths.

or worse 79.7% of the time. This gives us good confidence that our timestamping application produces results with high statistical similarity to an attack exploiting HTTP redirection.

Given the similarity between the measurement methods, we used the simple timestamping application to perform all pipelined measurements reported here, due to two factors. First, our application is much simpler to deploy and configure for distributed experiments, and results in a lower computational overhead on the PlanetLab testbed. Second, each timestamp generated by our application consumes only four bytes of bandwidth in each direction, whereas a complete HTTP request and response consumes roughly 400 bytes of bandwidth; thus our application resulted in reduced bandwidth depletion of the Tor network.

4. CONNECTION LINKING VIA LATENCY

The scenario behind our connection linking attack has two colluding servers, Y and Z, both accept connections from the same proxy 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. Figure 3 shows these scenarios in the context of Tor.

4.1 Attack Description

Our attack works as follows: we assume that Y communicates with client A over an anonymous connection terminating at node X, and Z communicates with B over an anonymous connection also terminating at 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 node U, then the idea behind our attack is to take several samples from both $T_{AY} - T_{XY}$ and $T_{BZ} - T_{XZ}$, and compare to see if they come from the same probability distribution. In the simple

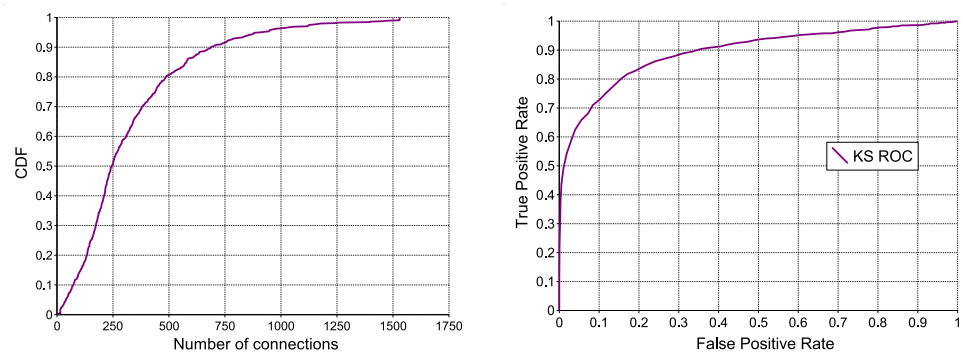
proxy system, we have $T_{AY} = T_{AX} + T_X + T_{XY}$ and $T_{BZ} = T_{BX} + T_X + T_{XZ}$, while the equivalent Tor circuits involve nodes E and M between A and X, and N and R between B and X, giving $T_{AY} = T_{AE} + T_E + T_{EM} + T_M + T_{MX} + T_X + T_{XY}$ and $T_{BZ} = T_{BN} + T_N + T_{NR} + T_R + T_{RX} + T_X + T_{BZ}$. If $A = B$, (and thus, over Tor, $E = N$, 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.

We used the *Kolmogorov-Smirnov*, or K-S, test to compare the sample sets in both attacks. The 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.

The K-S test has a tunable rejection region, giving a tradeoff between false positive and false negative error rates, so no single number characterizes the performance of the attack. Thus, we evaluate the attacks 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 1.0, while the nondiscriminating classifier has AUC 0.5. See Fawcett's tutorial [Fawcett 2006] for a more comprehensive treatment.

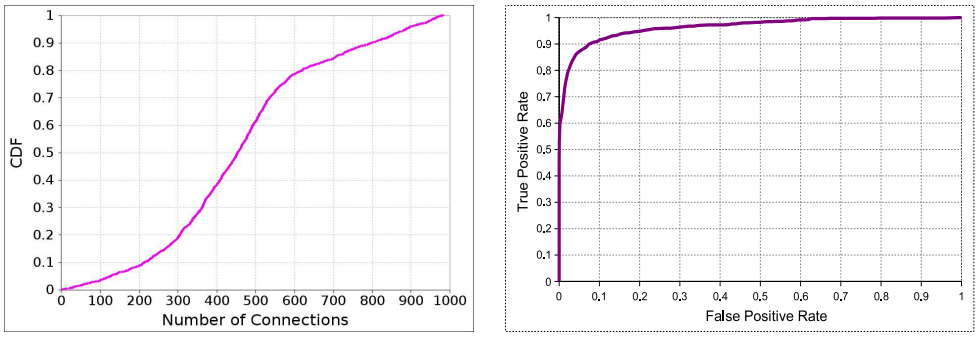
4.2 Evaluation

We tested the effectiveness of this attack using clients and servers from the PlanetLab wide-area testbed, in three scenarios: first, we tested our generic connection-timing method against Tor; second, we tested our timestamping application over Tor, and finally, for comparison, we tested our timestamping application over the MultiProxy network. Each evaluation consisted of multiple "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 exit point X from the anonymity scheme was chosen and each client established a connection to X. In the MultiProxy case, X was chosen from among the currently available proxies, while in the Tor case, X was chosen from among the high-bandwidth high-uptime exit nodes, and A and B independently chose entry and middleman nodes to build circuits terminating at X. After A and B established their respective connections, both clients connected to both servers, and the servers sampled these circuit times as described in Section 3. These four connection times were used in six comparisons: comparing samples from T_{AZ} to T_{AY} and T_{BZ} to T_{BY} gave two true positives, while comparing samples of T_{AZ} to T_{BZ}, T_{AZ} to T_{BY}, T_{AY} to T_{BZ}, and T_{AY} to T_{BY} gave four true negatives.



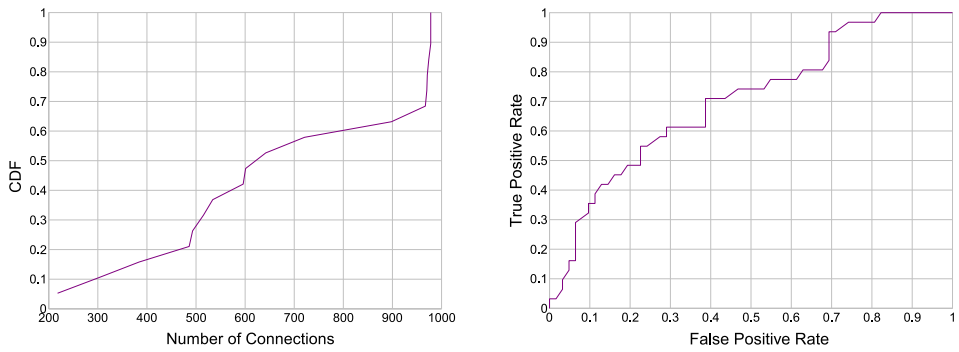
(a) (b)

Fig. 4. Client linking in Tor with generic timing. Figure (a) shows the cumulative distribution of timing samples per run. Figure (b) shows the ROC results; the area under curve is 0.89.



(a) (b)

Fig. 5. Client linking in Tor with pipelined timing. Figure (a) shows the cumulative distribution of timing samples per run. Figure (b) shows the ROC results; the area under curve is 0.96.



(a) (b)

Fig. 6. MultiProxy.com linking results with timestamping. Figure (a) shows the cumulative distribution of timing samples per run. Figure (b) shows the ROC results; the area under curve is 0.70.

Counting the number of misclassified streams for various threshold values allowed us to calculate false positive and false negative rates. One important note is that the current version of Tor recycles a used circuit after 10 minutes. Thus we set all of our experiments to stop after 10 minutes as well, regardless of whether the run had completed.

Linking Tor circuits with generic timing. Figure 4 summarizes the results of our Tor experiment using generic circuit timing. In all we collected 640 runs using this method, with a median sample size after 10 minutes of approximately 250, as shown in Figure 4(a). Overall the results are significantly better than random guessing. As Figure 4(b) shows, the attack has a reasonable tradeoff between false positives and false negatives, 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.

Linking Tor circuits with pipelined timing. 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 performed a set of 522 runs using our timestamping application to measure circuit times. For these runs, the performance of the attack was significantly improved: the equal error rate improved to 8% with AUC 0.96. The full results are shown in Figure 5.

Linking proxy connections. Tor circuits can potentially have a much higher level of distinguishability than connections over a simple proxy: if the variability of queueing latency at the proxy dominates the difference in network latencies, then it will be difficult to distinguish two clients using the same proxy, whereas in Tor the potentially idiosyncratic latencies of the different entry and middle nodes

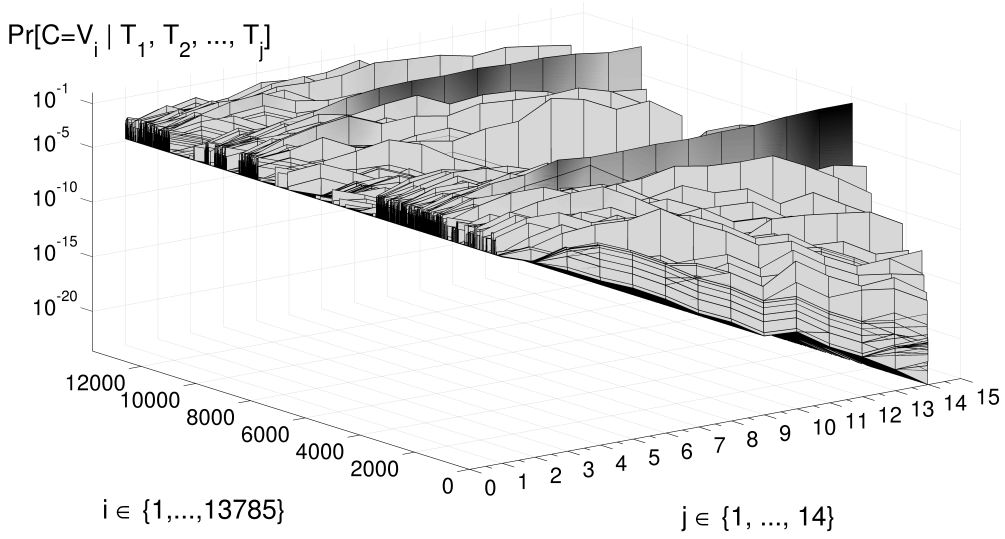


Fig. 7. Probability assigned to candidate locations 1-13874 after revelations 0-14, for one experiment. Note that the probabilities are plotted on a logarithmic scale.

contribute to the uniqueness of a circuit's timing distribution. To test the extent to which our circuit linking attack measures differences in queueing delays versus network latency, we performed a set of 20 runs using our timestamping application over the MultiProxy network. The results, shown in Figure 6 confirm that Tor circuit linking works well in part due to the additional diversity of Tor circuit times: the area under curve for MultiProxy connections was 0.70 with an equal error rate of 38%.

5. LATENCY WITHOUT NOISE

The possibility of using latency data in traffic analysis has been mentioned several times in previous works, apparently originating in a paper by Back *et al.* [2001]. 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 any topology where all host pairs have the same RTT (for example, a "star topology" where all hosts are connected by equal-length wires to a single switch) would leak no information about a host's location, while latency in a unidirectional ring would uniquely identify every host – 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

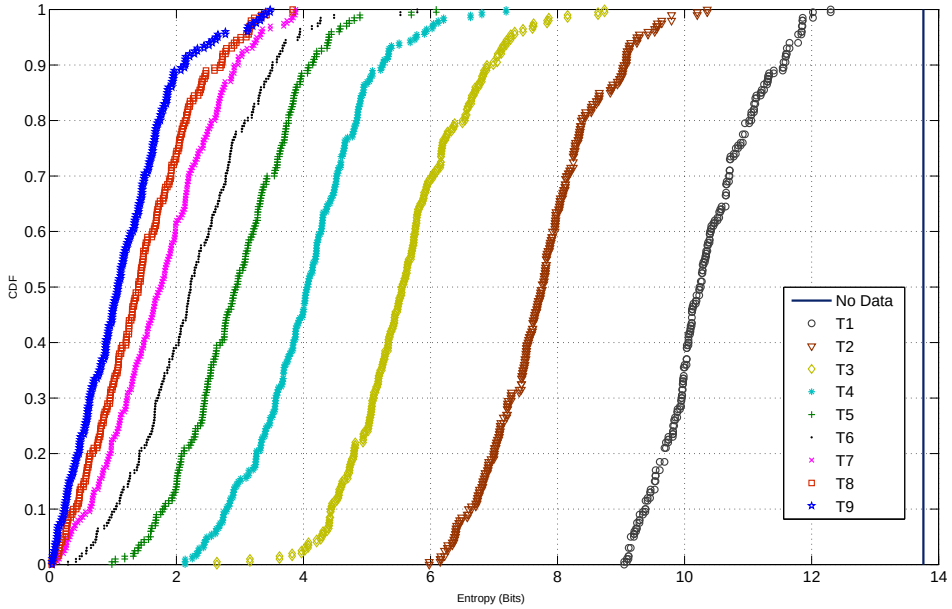


Fig. 8. Distribution of entropy of probability distribution $\Pr[V = C_i \mid T_1, \ldots, T_j]$, for $j \in \{0, ..., 9\}$.

clients using latency information.

More precisely, we imagine the following scenario: starting from a set of potentially distinguishable candidate network locations $\{C_1, C_2, \ldots, C_N\}$, the environment uniformly picks a "victim" location V, that the adversary will try to guess. Since V is chosen uniformly, initially the adversary has $H_0 = \log_2 N$ bits of uncertainty about the location of V. Then the environment uniformly chooses a "beacon" location B_1 and reveals B_1 and a small sample T_1 from the distribution on latency between B_1 and V to the adversary. The adversary can sample the latencies between B_1 and all of the C_is and update his "belief" distribution $\Pr[V = C_i]$ appropriately. His new uncertainty is $H_1 = \sum_i -\Pr[V = C_i|T_1] \log \Pr[V = C_i|T_1]$, and the information gained by this revelation is $H_0 - H_1$. As we repeat this process with randomly chosen B_2,B_3,B_4, and so on, the adversary updates his beliefs accordingly and becomes increasingly certain about the location V. The information leakage from m measurements is then $H_0 - H_m$, and if H_m is small then there is a small set of candidate locations that are likely for V.

To measure the information leakage of network latency on the Internet, we performed a large-scale survey in August 2007. Using the `routeviews` database of routable IP prefixes and reverse DNS queries, we identified 13874 DNS servers in distinct subnetworks. We then measured the round trip time to these servers from 14 randomly chosen PlanetLab hosts (at distinct sites). These hosts served as the "beacon" locations in the scenario above. At the same time, we took additional latency samples from the beacon hosts to 200 "victim" nodes chosen uni-

formly at random from the set of "beacon" nodes. We used the nonparametric Mann-Whitney-Wilcoxon (MWW) statistic for distribution shift and Bayes' Rule to compute the probability $Pr[V = C_i|T]$.

Figure 7 shows the result of one such experiment. In this figure, the righthand axis ranges over candidate locations $\{C_1, \ldots, C_{13874}\}$, the lefthand axis ranges over beacon hosts $\{B_1, \ldots, B_{14}\}$ and the vertical axis at point (C_i, B_j) depicts at logarithmic scale the quantity $\Pr[V = C_i \mid T_1, ..., T_j]$. The figure shows that initially the distribution is uniform across all candidate locations, and with further measurements, the victim's location becomes increasingly likely while other candidates become increasingly unlikely.[7]

Figure 8 summarizes the results for all 200 victims. We see that on average, the information gain from the first revelation is 3.5 bits, with decreasing information gain from subsequent revelations. Overall, all 200 experiments resulted in less than 2 bits of uncertainty after 14 revelations, with median uncertainty 0.35; The median information gain after 3 revelations was 8.2 bits, and the median information gain after 6 revelations was 11.5 bits. We note that in [Hopper et al. 2007] we used a different methodology to estimate the information gained from a single RTT, based on the MIT King data set [Gil et al. 2005], and arrived at a similar first-hop estimate of 3.6 bits.

Recall that when clients are identified with a network location, there will always be large sets of nodes that are essentially indistinguishable within this location. For example, typically all of the hosts within a routable IP prefix will be within a few routing hops of each other. Since there are currently about 200,000 routable IP prefixes, we estimate conservatively that there are at most 2^{18} distinguishable locations in the current Internet. Our study covered only the 14,000 prefixes for which we could identify reverse DNS servers located within the prefix, in order to avoid random scanning of IP addresses. Within this sample set, on average 9.4 RTTs are enough to identify one location. In order to estimate how this figure changes when considering the entire Internet, we randomly sampled 256, 512, 1024, 2048, and 8192 of these locations, and measured the average number of RTTs needed to identify a location within these sets. We found that the average increase in RTTs required when the number of locations doubled was 0.44, and that on average 9.3 RTTs were sufficient to identify one of 8192 locations. We can thus extrapolate a loose estimate of $9.3 + 0.44 \times (18 - 13) = 11.5$ RTTs (on average) to identify an Internet location.

6. MULTIPROXY CLIENT LOCATION VIA LATENCY

The previous results make it clear that knowing the latency between a client and several hosts will allow an adversary to identify or significantly reduce the set of possible locations of that client. In the context of an anonymity scheme, we translate this to a scenario in which a client wishes to access a server pseudonymously, for example by participating in a discussion forum or using a web mail service such as `hotmail` or `gmail`. Each time the client accesses the server, some information

[7] In this figure, a second candidate location is assigned a higher probability than the others. For the experiment, the victim was a host with a ".ru" domain name, the country code for Russia, while the other probable location was a host with a ".ua" domain, the country code for the Ukraine.

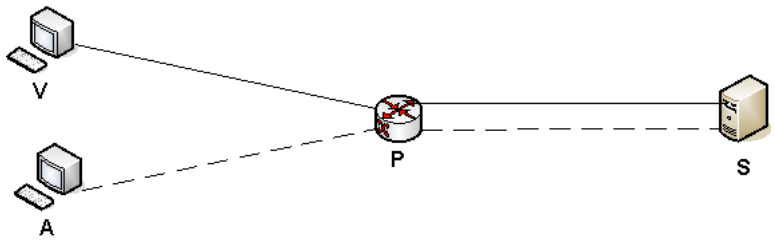
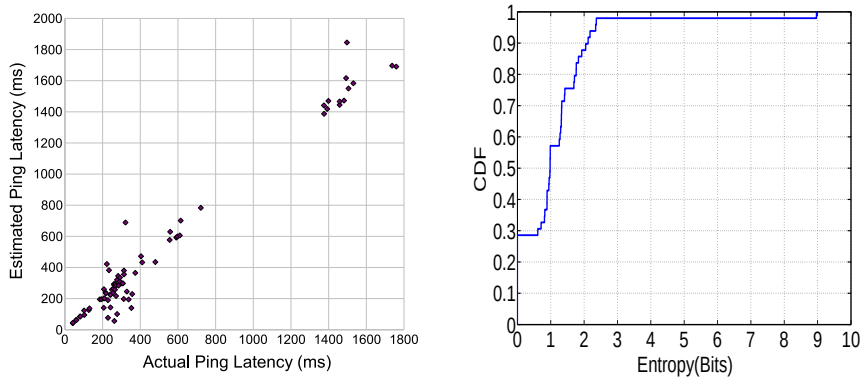


Fig. 9. MultiProxy client location: client V connects to server S through proxy P; A connects to S through P as well.



(a) (b)

Fig. 10. Results of MultiProxy client location. (a) Maximum likelihood estimates vs actual ping times (b) cumulative distribution of entropy

about its network location may leak, and the goal of the malicious server is to identify the client's probable network location with increasing confidence using as few of these linkable anonymous connections as possible.

In the context of a proxy aggregation service, this is relatively straightforward. As shown in Figure 9, when the client V connects to the server S through a proxy P, we can employ the timing methods of Section 3 to sample from the distribution $T_{VS} = T_{VP} + T_P + T_{PS}$. These samples include the queueing delay from P as well as the network latencies T_{PS} and T_{VP}. In order to sample from T_P, the malicious server uses a collaborating client A to connect through P, obtaining samples from $T_{AS} = T_{AP} + T_P + T_{PS}$. Using these samples, the server computes an estimate for the distribution T_{VP}. If this estimate is accurate, in that it has high mutual information with the correct distribution, it can be used in place of the "revealed" time T_1 from the previous section to update the adversary's beliefs on the network location of the victim.

To compute the probability distribution on T_1, we record the measured RTT T_{AP} from each sample taken from T_{AS}. Then for each candidate round trip time τ, we use the MWW test to compute the probability $\Pr[T_{VP} = \tau | T_{VS}, T_{AS} - T_{AP}]$. Computing the entropy of this distribution tells us how much uncertainty

the attacker has about T_{VP} given the observations, or how much information is lost by this method of measuring T_{VP}. We note that this method of computing information gain is different from the method employed in [Hopper et al. 2007], in which a specific set of candidate nodes were evaluated, and discarded if their estimated RTT did not lie in a confidence interval around the observed RTT; the present method gives a "modular" accounting of the information lost by the use of an anonymity scheme regardless of the set of candidate nodes.

6.1 Evaluation

We measured the information loss due to use of a single-hop proxy by performing an experiment using the PlanetLab network testbed and the MultiProxy aggregation service in January 2008. Our experiment collected data from 50 runs. In each run, random PlanetLab nodes were selected for the attacker, victim, and server, and a random open proxy was chosen from among the available nodes on the MultiProxy list. Then the attacker and the victim both used our timestamp application to connect to the server through the proxy, and up to 1000 timestamps were recorded for each connection, with the run terminating ending 10 minutes after initiation. At the conclusion of a run, we computed the conditional distribution on T_{VP} and also recorded the true round trip time between the victim and the proxy.

Figure 10 summarizes the results of these experiments. The average number of timestamps recorded per connection was 817 with an average connection round trip time of 787ms. The average entropy was 1.12 bits, with a median value of 0.9. Thus on average an attacker loses very little information about the client's location behind a single hop proxy. Furthermore, since the uncertainty is computed independently of any other data from the attack, runs with high uncertainty as to the Proxy-Victim latency can simply be discarded. Since 28% of the runs had essentially no uncertainty regarding the Proxy-Victim latency, discarding high-entropy runs will only increase the number of connections required by a factor of 3.57. Given our estimate that 11.5 RTT measurements are sufficient to uniquely determine the client's network location, this yields an estimated upper bound of $3.57 \times 11.5 = 41$ visits required. In the next section we consider how Tor's multiple hops and cryptographic overhead impact our attack.

7. TOR CLIENT LOCATION VIA LATENCY

The basic scenario of our client location attack, when extended to Tor, is shown in Figure 11. In this attack, the adversary consists of three 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.

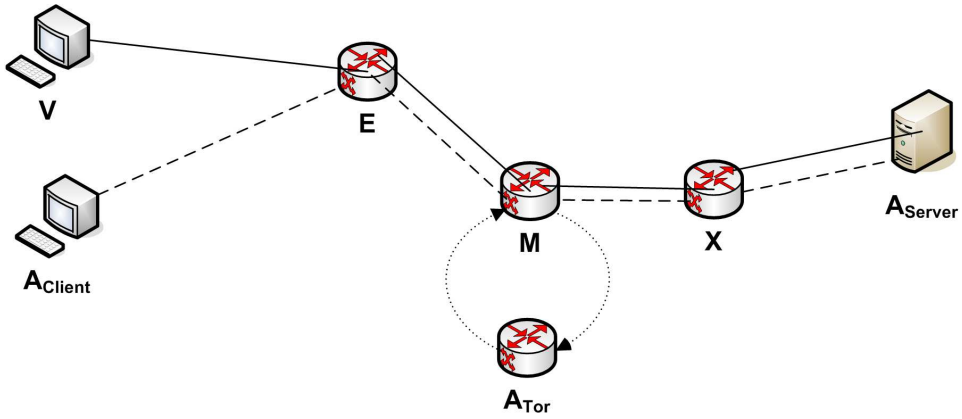


Fig. 11. Tor client location: client V connects to malicious server A via circuit E-M-X; A determines E-M-X and connects to A via E-M-X.

7.1 Attack Description

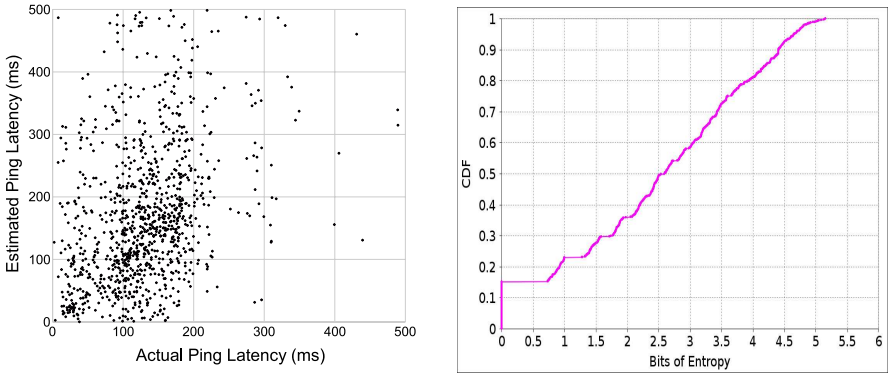
The basic idea of our client location attack is to try to measure – using a Tor connection – T_{VE}, the RTT between the victim V and the Tor entry node, E. The attacker then estimates, for several candidate victim nodes C, the RTT T_{CE}. Using these estimates, the attacker updates his beliefs about the victim node, and the attack is repeated. After several iterations, the attacker should have high confidence in a few candidate network locations for the victim.

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}. As in the previous section, these samples, along with the knowledge of the A_{Client}-entry node time T_{AE} can be used to compute a probability distribution on the value T_{VE}. If this distribution has high entropy, we can wait for another circuit using entry node E to improve our estimate of T_{VE}.

7.2 Evaluation

To measure the extent to which Tor circuits obscure the victim-entry node latency, we performed an experiment using the deployed Tor network in February and March



(a) (b)

Fig. 12. Results of Tor client location. (a) Maximum likelihood estimates vs actual ping times, victim to entry node (b) cumulative distribution of entropy per run

2008. Our experiment consisted of 995 runs, where in each run, we randomly selected PlanetLab nodes to act as A_{Client}, A_{Server} and V. Then a random Tor entry node, middleman node, and exit node were selected to act as E,M, and X. Both A_{Client} and V built circuits using these nodes and then connected to the timestamp application on A_{Server} using the constructed circuit. We collected up to 1000 timestamps for each circuit, terminating the experiment after 10 minutes, and also measured the RTT from A_{Client} to E. At the end of the run, we used these timestamps to compute a conditional probability distribution for T_{VE} as in the previous section, applying a "bandpass" filter that eliminated ping times with less than $\frac{1}{8}$ the probability of the most likely ping time, and then computed the entropy of the T_{VE} distribution. Note that since we are mainly interested in measuring the effects of latency, we do not perform the Murdoch-Danezis attack in our experiments.

Figure 12 summarizes the results of these runs. The total number of runs used was 995, and an average of 1261 (adversary and victim) timestamps were collected per circuit, with an average circuit RTT of 5597ms. The median information loss was 2.46 bits of entropy, with 23% of runs resulting in less than 1 bit of information loss about the victim latency T_{VE}.

To test the reason for the high variability in information loss, we computed the correlation between several properties of the circuit and the information loss. We found that the "relative spread", the difference between the 25th-lowest circuit RTT and the lowest circuit RTT expressed as a fraction of the lowest circuit RTT, accounted for 48% of the variation in information loss. Figure 13 shows the relative spread versus the information loss. Since the relative spread is a function of the delay of all nodes in a circuit and the number of samples, this highly suggests that information loss is independent of entry node. Thus, our experiments suggest that on average, the number of Tor circuit measurements required for a given expected information gain is roughly $1/.229 = 4.37$ times the number of direct RTT measurements required. Combined with our estimate of 11.5 RTTs required to identify a client's location, this yields an estimate of approximately $4.37 \times 11.5 = 50$ server

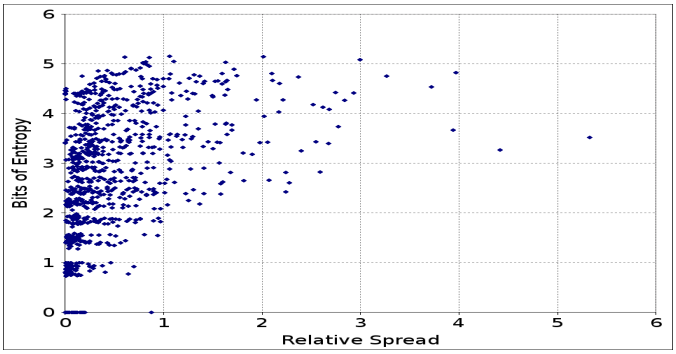


Fig. 13. relative spread vs information loss

accesses to identify a client's network location when no prior information is available.

8. ESTIMATING CANDIDATE RTTS

Once we have estimated the RTT from the victim to the Tor entry node E, the next step in our attack compares this measurement to the RTT between candidate nodes and E. So far we have evaluated the attack as if these are known quantities. 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 overall attack to work, we need a method to obtain (or at least estimate) the RTT between two hosts without the explicit cooperation of either. We measure the ability to do so using 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 [Ng and Zhang 2004; Dabek et al. 2004; Costa et al. 2004; Ledlie et al. 2007], using various coordinate systems and embedding algorithms. We chose to use the Vivaldi [Dabek et al. 2004] 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 [Spring et al. 2003] and `traceroute.org`.

To assess the correspondence between network coordinates and ping times, we performed the following experiment in February 2008. We chose a set of 100 random PlanetLab nodes and 1500 Tor nodes. The PlanetLab nodes were divided into 50 "attacker" nodes and 50 "victim" nodes. Then we measured the ping times between all attacker PlanetLab nodes, and the ping times from the attacker nodes to all of the Tor nodes and all of the victim nodes, and fit a coordinate system to these

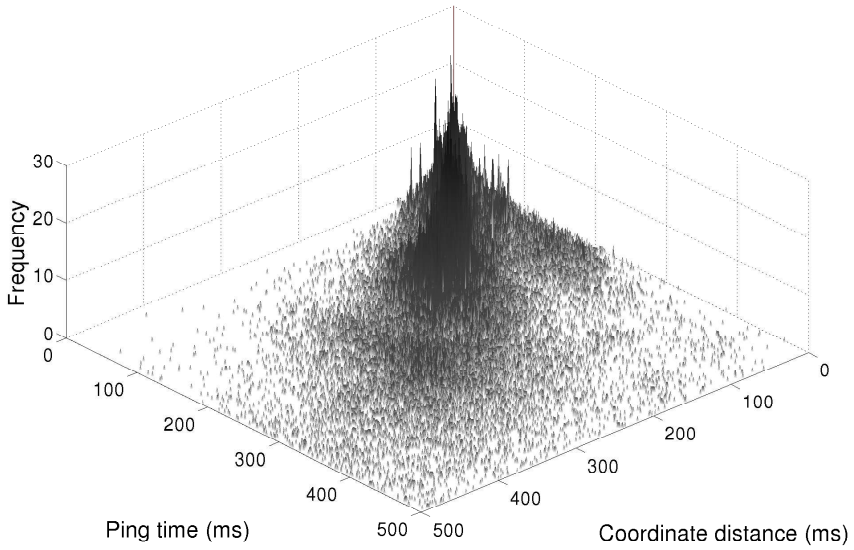


Fig. 14. Frequency (z) of coordinate distance (x) vs ping time (y) pairs.

times. For evaluation, we also measured the ping times between the victim nodes and the Tor nodes. Finally, we compared the coordinate distances from victim nodes to Tor nodes with measured ping times between these nodes.

Figure 14 illustrates the relationship between the victim coordinate distances and the measured ping times. We computed the frequency of each pair of measured RTT and predicted RTT, to produce an empirical probability distribution; the figure shows a sharp spike in probability following the $x = y$ line, indicating that network coordinates computed in this way strongly predict RTTs with little information loss.

Several alternate possibilities exist for the implementation of this step, that we did not evaluate empirically. One example is the King technique [Gummadi et al. 2002], 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.* [Gummadi et al. 2002] report that this technique has accuracy competitive with the GNP [Ng and Zhang 2004] 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.

9. DISCUSSION

Limitations. One limitation of our client location attack is that we assume 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.

Another limitation, of the client location attack but not the linking attack, is our reliance on the Murdoch and Danezis [2005] attack. The authors showed that their attack worked against 13 of 15 nodes that they evaluated in the much smaller Tor network from 2004, and had a duration of six minutes. As it turns out, our attack based on redirection can be made to work as long as the attack completes within a tolerable time period: the persistent streams opened by Firefox/Polipo will not be migrated by the Tor client, and the very first redirection response can contain the data bursts generated by the attack as a long sequence of unrecognized extension headers. However, it is not clear that any duration is sufficient to conduct the attack against the much larger and more heavily loaded Tor network of 2008. More research is needed to determine if this is the case.

Other Applications and Extensions. We evaluated our attacks in the context of both simple proxies and the Tor anonymity scheme, but we expect that they should be generalizable to other low-delay anonymity protocols. For example, peer-to-peer designs such as Crowds [Reiter and Rubin 1998], MorphMix [Rennhard and Plattner 2002] and I2P [jrandom et al. 2007], with lightly-loaded relays and multiple entry points, should 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 beyond persistent HTTP [Fielding et al. 1999] include IRC [Oikarinen and Reed 1993], and SIP [Rosenberg et al. 2002]. 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 [2006] 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, even if Murdoch-Danezis attacks are infeasible, 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 [Wong et al. 2006; Gueye et al. 2006]. 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. Other mechanisms which mitigate the Murdoch-Danezis attack by increasing the time required can defeat client location with general timing, but as discussed above, will not prevent client location using persistent HTTP with redirection. It is conceivable that there may be an adverse interaction between the conditions necessary for the Murdoch-Danezis attack to succeed and the conditions necessary for our attack to succeed; this could be determined by experiments that vary the network load on Tor while also carrying out our attack in full. If such an interaction exists, it could be possible to engineer a low-latency anonymity network such that conditions are always unfavorable for one part of the attack.

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. Such attempts do not prevent the measurement of circuit RTTs, nor, due to the variety of ways to measure internet RTTs and the abundance of "nearby" hosts on most networks, do they prevent the mapping necessary to peform the attack; they do, however, make this mapping slightly less convenient.

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 Roger Dingledine, Aaron Johnson, Yongdae Kim, Jon McLachlan, Cat Okita, Ivan Osipkov, Paul Syverson, 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.

REFERENCES

2008. TOR node status information. `https://torstat.xenobite.edu/`.

Back, A., Möller, U., and Stiglic, A. 2001. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proceedings of Information Hiding Workshop (IH 2001)*, I. S. Moskowitz, Ed. Springer-Verlag, LNCS 2137, 245–257.

Blum, A., Song, D., and Venkataraman, S. 2004. Detection of interactive stepping stones: Algorithms and confidence bounds. *Proc. of The Seventh International Symposium on Recent Advances in Intrusion Detection (RAID)*.

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

Chroboczek, J. 2003–2008. Polipo – a caching web proxy. http://www.pps.jussieu.fr/ jch/software/polipo/.

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

Costa, M., Castro, M., Rowstron, A., and Key, P. 2004. PIC: Practical internet coordinates for distance estimation. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*. IEEE Computer Society, Washington, DC, USA, 178–187.

Dabek, F., Cox, R., Kaashoek, F., and Morris, R. 2004. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, New York, NY, USA, 15–26.

Danezis, G. 2003. Statistical disclosure attacks: Traffic confirmation in open environments. In *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, Gritzalis, Vimercati, Samarati, and Katsikas, Eds. IFIP TC11, Kluwer, Athens, 421–426.

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

Díaz, C. and Serjantov, A. 2003. Generalising mixes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, R. Dingledine, Ed. Springer-Verlag, LNCS 2760.

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

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

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recogn. Lett. 27,* 8, 861–874.

Federrath, H. et al. 2006. JAP: Java anonymous proxy. `http://anon.inf.tu-dresden.de/`.

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

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

Gueye, B., Ziviani, A., Crovella, M., and Fdida, S. 2006. Constraint-based geolocation of Internet hosts. *IEEE/ACM Transactions on Networking (TON) 14,* 6, 1219–1232.

Gummadi, K. P., Saroiu, S., and Gribble, S. D. 2002. King: estimating latency between arbitrary Internet end hosts. *Proceedings of the second ACM SIGCOMM Workshop on Internet measurement*, 5–18.

Hintz, A. 2002. Fingerprinting websites using traffic analysis. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*, R. Dingledine and P. Syverson, Eds. Springer-Verlag, LNCS 2482.

Hopper, N., Vasserman, E. Y., and Chan-Tin, E. 2007. How much anonymity does network latency leak? In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*. ACM, New York, NY, USA, 82–91.

jrandom et al. 2007. I2P. `http://www.i2p.net/`.

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

Ledlie, J., Gardner, P., and Seltzer, M. 2007. Network coordinates in the wild. In *Proceedings of the 4th USENIX Symposium on Network Systems Design and Implementation (NSDI)*.

Mathewson, N. and Dingledine, R. 2004. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*. LNCS, vol. 3424. 17–34.

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

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

Murdoch, S. J. and Danezis, G. 2005. Low-Cost Traffic Analysis of Tor. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE Computer Society, Washington, DC, USA, 183–195.

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

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

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

Panchenko, D. 2006. *Lecture Notes of 18.443, Statistics for Applications*. MIT Open Courseware Project, `http://ocw.mit.edu/OcwWeb/Mathematics/18-443Fall-2006/CourseHome/index.htm`.

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

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

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

Serjantov, A. and Sewell, P. 2003. Passive attack analysis for connection-based anonymity systems. In *Proceedings of ESORICS 2003*.

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

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

Wong, B., Stoyanov, I., and Sirer, E. G. 2006. Geolocalization on the Internet through constraint satisfaction. *Proc. USENIX WORLDS*.

Wright, M., Adler, M., Levine, B. N., and Shields, C. 2003. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*.