Low-Cost Traffic Analysis of Tor

Steven J. Murdoch and George Danezis
University of Cambridge, Computer Laboratory
15 JJ Thomson Avenue, Cambridge CB3 0FD
United Kingdom
{Steven.Murdoch,George.Danezis}@cl.cam.ac.uk

Abstract

Tor is the second generation Onion Router, supporting the anonymous transport of TCP streams over the Internet. Its low latency makes it very suitable for common tasks, such as web browsing, but insecure against traffic-analysis attacks by a global passive adversary. We present new traffic-analysis techniques that allow adversaries with only a partial view of the network to infer which nodes are being used to relay the anonymous streams and therefore greatly reduce the anonymity provided by Tor. Furthermore, we show that otherwise unrelated streams can be linked back to the same initiator. Our attack is feasible for the adversary anticipated by the Tor designers. Our theoretical attacks are backed up by experiments performed on the deployed, albeit experimental, Tor network. Our techniques should also be applicable to any low latency anonymous network. These attacks highlight the relationship between the field of traffic-analysis and more traditional computer security issues, such as covert channel analysis. Our research also highlights that the inability to directly observe network links does not prevent an attacker from performing traffic-analysis: the adversary can use the anonymising network as an oracle to infer the traffic load on remote nodes in order to perform traffic-analysis.

1 Introduction

Anonymous communication networks were first introduced by David Chaum in his seminal paper [10] describing the mix as a fundamental building block for anonymity. A mix acts as a store-and-forward relay that hides the correspondence between messages it receives and sends. Several mix based architectures have been proposed and implemented to anonymise email, most notably Babel [26], Mixmaster [30] and the newer Mixminion [15]. Their latency is tolerable for email, but is unsuitable for interactive applications such as web browsing.

Other systems, based on the idea of a mix, were developed to carry low latency traffic. ISDN mixes [33] propose a design that allows phone conversations to be anonymised, and web-mixes [6] follow the same design patterns to anonymise web traffic. A service based on these ideas, the Java Anon Proxy (JAP)[1] has been implemented and is running at the University of Dresden. These approaches work in a synchronous fashion, which is not well adapted for the asynchronous nature of widely deployed TCP/IP networks [8].

The Onion Routing project has been working on stream-level, low-latency, high-bandwidth anonymous communications [35]. Their latest design and implementation, Tor [18], has many attractive features, including forward security and support for anonymous servers. These features, and its ease of use, have already made it very popular, and a testing network, available for public use, already has 50 nodes acting as onion routers (as of November 2004).

Tor aims to protect the anonymity of its users from non-global adversaries. This means that the adversary has the ability to observe and control some part of the network, but not its totality. Similarly, the adversary is assumed to be capable of controlling some fraction of Tor nodes. By making these assumptions, the designers of Tor believe it is safe to employ only minimal mixing of the stream cells that are relayed, therefore lowering the latency overhead of the communication.

This choice of threat model, with its limitation of the adversaries' powers, has been a subject of controversy in the anonymity community, yet most of the discussion has focused on assessing whether these restrictions of attackers' capabilities are 'realistic' or not. We leave this discussion aside and instead show that traffic-analysis attacks can be successfully mounted against Tor even within this very restricted threat model.

Our attacks are based on the notion that the timing signature of an anonymised stream can be used to track it in the

[1] http://anon.inf.tu-dresden.de/

Tor network, since the low latency hardly distorts it. The adversary is not global so he cannot observe this timing signature directly in the network. Instead the adversary sends his own traffic streams through a node and monitors the latency. He uses the fact that the traffic volume of one stream influences the latency of other streams carried over the same Tor node. To assess if one of the target streams is carried over a Tor node, the adversary routes a stream that he can observe, over the same node, and measures the changes in latency. Besides tracing the route of an anonymous communication, our traffic-analysis attacks can also be used to link transactions together. Our techniques allow any user to perform traffic-analysis on the whole network, and thereby approximate the capabilities of a global passive observer.

This might at first seem to be a typical example of information leakage, or covert channels, that have for decades haunted the community working on multi-level secure systems. Yet there is a fundamental difference: anonymous communication relies on traffic from many different sources being mixed together. Therefore the established solution to covert channels, of separating the different streams to avoid timing information leakage, would completely ruin the anonymity properties of the system. For this reason, novel techniques will have to be devised to cope with our attacks.

The results we present should be seriously considered by designers of anonymous communication systems. They concern a widely deployed, popular, and well used system that represents the state of the art in both research and implementation. Few systems of such a standard have been deployed (Freedom [23, 9, 4] and JAP [6] being the others), which has made practical experimentation to verify the effectiveness of theoretical attacks very difficult. Also, our attacks highlight an architectural flaw that leads to information leakage, and this could affect other designs of anonymity systems. The parallels that this problem has with covert channels in multilevel security brings the field of anonymous communications closer to more traditional computer security disciplines. The approach of performing covert channel analysis to assess the security of an anonymous communication system was pioneered in Moskowitz *et al.* [31, 32]. Our attacks show that this is not simply a theoretic model, but techniques from this community can be effective in practice in degrading the anonymity provided by real systems.

2 Understanding The Onion Router (Tor)

The Onion Router (Tor) [18] is the culmination of many years of research by the Onion Routing project [35, 24, 39]. Not only is it a completely new design and implementation, but it reflects a shift from the traditional threat models anonymous communication systems have tried to protect against. We first describe the Tor architecture and then introduce the threat model considered in the Tor design.

2.1 Architecture

The Tor network can be used to transport TCP streams anonymously. The network is composed of a set of nodes that act as relays for a number of communication streams, hopefully from different users. Each Tor node tries to ensure that the correspondence between incoming data streams and outgoing data streams is obscured from the attacker. Therefore the attacker cannot be sure about which of the originating user streams corresponds to an observed output of the network.

The Tor architecture is similar to conventional circuit switched networks. The connection establishment has been carefully crafted to preserve anonymity, by not allowing observers to cryptographically link or trace the route that the connection is using. The initiator of the stream creates a *circuit* by first connecting to a randomly selected Tor node, negotiating secret keys and establishes a secure channel with it. The key establishment uses self-signed ephemeral Diffie-Hellman key exchange [16] and standard Transport Layer Security (TLS) is further used to protect the connections between nodes and provide forward secrecy. All communications are then tunnelled through this circuit, and the initiator can connect to further Tor nodes, exchange keys and protect the communication through multiple layers of encryption. Each layer is decoded by a Tor node and the data is forwarded to the next Onion router using standard route labelling techniques. Finally, after a number of Tor nodes are relaying the circuit (by default three), the initiator can ask the last Tor node on the path to connect to a particular TCP port at a remote IP address or domain name. Application layer data, such as HTTP requests or SSH sessions, can then be passed along the circuit as usual. Since we are not attacking the cryptographic components of Tor we will not go into any further details on this subject. Interested readers should consult the Tor specification [17].

TCP streams travelling through Tor are divided and packaged into cells. Each cell is 512 bytes long, but to cut down on latency it can contain a shorter useful payload. This is particularly important for supporting interactive protocols, such as SSH, that send very small keystroke messages through the network.

Controversially, Tor does not perform any explicit mixing. Cells are stored in separate buffers for each stream, and are output in a round-robin fashion, going round the connection buffers. This ensures that all connections are relayed fairly, and is a common strategy for providing best effort service. Importantly, when a connection buffer is empty, it is skipped, and a cell from the next non-empty connection buffer is sent as expected. Since one of the objectives of

Tor is to provide low latency communications, cells are not explicitly delayed, reordered, batched or dropped, beyond the simple-minded strategy described above.

Tor has some provisions for fairness, rate limiting and to avoid traffic congestion at particular nodes. Firstly, Tor implements a so-called token bucket strategy to make sure that long-term traffic volumes are kept below a specified limit set by each Tor node operator. Since the current deployment model relies on volunteer operators, this was considered important. Yet this approach, on its own, would not prevent spikes of traffic from being sent, and propagating through a connection. These spikes of data would, of course, be subject to the maximum bandwidth of each node, and could saturate the network connection of some Tor nodes.

To avoid such congestion, a second mechanism is implemented. Each stream has two windows associated with it, the first describes how many cells are to be received by the initiator, while the other describes how many are allowed to be sent out to the network. If too many cells are in transit through the network – and have not already been accepted by the final destination – the Tor node stops accepting any further cells until the congestion is eased. It is important to note that this mechanism ensures that the sender does not send more than the receiver is ready to accept, thereby overfilling the buffers at intermediary Tor nodes. It also makes sure that each connection can only have a certain number of cells in the network without acknowledgement, thus preventing hosts from flooding the network. Tor does not, however, artificially limit the rate of cells flowing in any other way.

Finally, it is worth mentioning that each Tor circuit can be used to relay many TCP streams, all originating from the same initiator. This is a useful feature to support protocols such as HTTP, that might need many connections, even to different network nodes, as part of a single transaction. Unused Tor circuits are short-lived – replacements are set up every few minutes. This involves picking a new route through the Tor network, performing the key exchanges and setting up the encrypted tunnels.

2.2 Threat model

The principal objective of an adversary attacking an anonymous communication system is to link the initiators of connections with their respective communication partners and vice versa. For example, an adversary observing a web request coming out of the Tor network might be interested in determining its originator. Similarly, an attacker observing a connection into the Tor network would be interested in knowing which remote machine it is ultimately accessing. A secondary objective of the attacker is to link transactions, namely network connections, so as to establish that they are from the same initiator. This could allow an adversary to profile the initiator, by observing patterns in his communication habits.

Tor aims to protect against a peculiar threat model, that is unusual within the anonymous communications community. It is conventional to attempt to guarantee the anonymity of users against a global passive adversary, who has the ability to observe all network links. It is also customary to assume that transiting network messages can be injected, deleted or modified and that the attacker controls a subset of the network nodes. This models a very powerful adversary and systems that protect against it can be assumed to be secure in a very wide range of real world conditions.

Tor, on the other hand, like some other designs, most notably MorphMix [36] and Tarzan [21, 20], assumes a much weaker threat model. It protects against a *non-global* adversary that can only observe a fraction of the network, modify the traffic only on this fraction and control a fraction of the Tor nodes. Furthermore, Tor does not attempt to protect against *traffic confirmation attacks*, where an adversary observes two parties that he suspects to be communicating with each other, to either confirm or reject this suspicion. Instead, Tor aims to make it difficult for an adversary with a very poor *a priori* suspicion of who is communicating with whom, to gain more information.

It could be claimed that the weaker threat model makes Tor insecure and incapable of protecting the anonymity of users against powerful real-world adversaries. In particular, while real world adversaries are not omnipotent, they do have the ability to be *adaptive*, i.e. select where to monitor the network based on previous observations. This monitoring can be performed on deployed TCP/IP or telephone networks using the lawful interception capabilities integrated in most modern routing equipment [40]. Access to these capabilities is, or at least should be, restricted to authorised parties only.

The importance of our attacks is that an adversary can extract information about the path of a Tor connection without stepping outside the threat model considered by Tor, and the methods used are accessible to any Tor user. Therefore we show that even relatively weak adversaries can perform traffic-analysis, and get vital information out of Tor. This means that even non-law-enforcement agencies can significantly degrade the quality of anonymity that Tor provides, to the level of protection provided by a collection of simple proxy servers, or even below.

3 Attacking The Onion Router

An attacker aims to gain some information about who is communicating with whom through the Tor network. We will present an overview of the techniques that an attacker can use to trace the communication and the constraints introduced by the restrictive Tor threat model. These lead to

the theoretical exposition of our attacks, the practical results of which are presented in Section 4.

3.1 Traditional traffic-analysis

Traffic-analysis is extracting and inferring information from network meta-data, including the volumes and timing of network packets, as well as the visible network addresses they are originating from and destined for. In the case of anonymous communications, an adversary would use this data to perform traffic-analysis with the aim of tracing who the originator or the ultimate destination of a connection is – therefore violating the anonymity properties that the system is designed to provide. We assume that Tor intermediaries, through the use of encrypted tunnels, effectively hide the bit patterns of data travelling though a Tor connection. Therefore an attacker cannot use any information from the content to trace the stream and has to resort to traffic-analysis.

Traffic-analysis can be performed at different levels of granularity. The first class of attacks considers the anonymous network as a "black box" and only consider the times when users are initiating connections, and connections are being relayed to network services outside the Tor network. Dogan Kesdogan *et al.* [27] were the first to show how repeated communications would eventually be revealed even if the anonymous channel was otherwise perfect. A statistical variant of these attacks, later presented [13], and validated through simulations [29], is more general and can be applied to a wider variety of anonymous communication channels.

Both these attack families are very powerful and would uncover repeated patterns of communication through Tor. For example, the disclosure and statistical disclosure attacks could, in the long run, reveal if a particular user connects every day to a set of web sites through Tor. An analysis of how long this would take can be found in Mathewson *et al.* [29] and Agrawal *et al.* [2]. Yet, to effectively mount such attacks, an adversary is required to observe a large fraction of the network in order to log who is accessing it and which outside services are used. This attacker is outside the threat model that Tor tries to protect against and therefore cannot be considered to break Tor as such[2].

A second category of attacks works at a much finer granularity. They inspect the traffic within the anonymous communication network, and further, the actual shape (load) of the traffic on each network link. Earlier work by the Onion Routing project drew attention to the fact that overall traffic patterns in connections are not particularly distorted by each Onion Router that relays them [39]. Therefore, a global observer would be able to correlate the timing and volume of incoming and outgoing streams in order to trace the route an onion-routed connection is taking though the network.

In Danezis [14] these finer granularity attacks are presented in detail, and a theoretical framework is developed to assess their effectiveness. In practice, an attacker observes a stream of data that is to be traced, for example, the reply of a web server to the initiator of a request. This stream of data can be represented as a function of traffic volume by time. The function is convolved with an exponential decay function: the result is a *template* that predicts how the stream will look in the anonymous network. All links of the network are then compared to assess if they match (more precisely, could have been generated by) the target template. Each network link will have a degree of similarity to the template that can be used to classify it as being after the first, second or third node on the path of the connection. Similar attacks have also been presented in Fu *et al.* [42] and Levine *et al.* [28].

The obvious problem of these attacks is that, as presented, the adversary observes all nodes, network links and is also to be able to record traffic meta-data at a much finer granularity than required for the disclosure attacks above. As a result, these attacks use a global passive adversary, which is not considered within the Tor threat model. At the same time, it is important to highlight that these attacks are robust [29]: when less, partial or lower resolution data is available they will take longer, and require more evidence until they provide the attacker with the same degree of certainty, but in the long run they will still work. Therefore, an attacker that controls part of the network, as Tor assumes, might still be able to trace some communications at random. However this is not very appealing to an attacker because of the amount of interception effort and analysis required.

An attack worth mentioning has been presented by Serjantov *et al.* [38]. They notice that by doing simple packet counting on lone connections, they can follow the anonymised streams. Their attack is appealing, since packet counting can be performed easily and cheaply by most routing equipment. Others have also looked at detecting *stepping stones* (relays) for intrusion detection [7, 41].

3.2 Traffic-analysis of Tor

As we have seen, traditional traffic-analysis techniques rely on vast amounts of data. The conventional wisdom has been that such data can only be gathered by a global passive observer, which lies outside the Tor threat model. The key contribution of our work is the realisation that such

[2]How realistic these attacks are is a completely different subject, that requires careful consideration of the size and topology of the anonymous communication network. In the case of Tor, a fully-connected network, an attacker would have to be able to know all the meta-data associated with the TCP connections to and from all Tor nodes. Given their small number (at time of writing, approximately 50) this might not be such a large effort. In the case of JAP [6], which arranges all relays in a cascade, only two nodes have to be under surveillance when applying disclosure or statistical disclosure attacks. In the latter case we judge them to be a real threat.

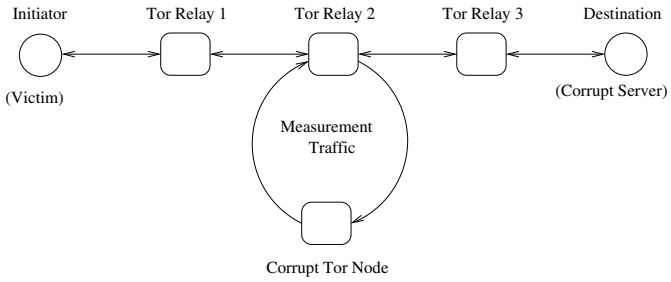


Figure 1. The attack setup

observation capabilities are not necessary to perform these attacks. The ability to route over the anonymous communication network, that anyone has, can be used to estimate the traffic load on specific Tor nodes accurately enough to perform traffic-analysis. Therefore adversaries with very modest capabilities can still detect the path that target connections are using through the Tor network.

Mix systems rely on the fact that actions, be it relayed messages, stream cells or connection startups, from different users, are processed by one party, the mix, in such a way that they become unlinkable to their initiator. In the case of Tor, multiple connections, from different users have to be relayed by the same Tor node, for any of them to be provided with any anonymity at all[3]. Since the relayed streams are processed and transported over a particular Tor node, they interfere with each other. This is due to the fact that they consume shared resources on a single machine – such as processor time and network bandwidth.

Some mixing strategies try to correlate streams in order to make them indistinguishable. The best example is the threshold mix batching strategy that waits until a particular number of messages have arrived and outputs them all at once. Tor does not use any particular batching strategy, since it would increase the latency of the communication. Instead, cells from different streams are sent out in a round robin fashion. When a stream has no cells available it is skipped and the next connection with cells waiting to be delivered is used. This means that the load on the Tor node affects the latency of all connection streams that are routed through this node. A similar increase in latency is introduced at all layers. As expected, the higher the load on the node, the higher the latency.

The simple observation that higher load, even due to one extra connection, on a Tor node will result in higher latency of all other connections routed through it can be used by an attacker. By routing a connection through specific Tor nodes, and measuring the latency of the messages, the adversary can get an estimate of the traffic load on the Tor node, that is, the superposition of the traffic load resulting from all relayed connections. This, in turn, can be compared with a known traffic pattern to assess if it is present in, and therefore relayed through the node, using conventional traffic-analysis techniques [14].

Any Tor user can perform these measurements and try to estimate the load on a Tor server. On the other hand, a Tor node is not subject to some restrictions that apply to clients (e.g. bandwidth limits), therefore, without loss of generality, we consider that the attacker controls a corrupt Tor node. This is in accordance with the Tor threat model, and allows us to ignore whether the node to be measured is also an exit node or not. This corrupt Tor node creates a connection that passes through another Tor node, whose traffic load is to be measured. This connection is then filled with *probe traffic*, that measures the latency of the connection and therefore estimates the load on the target Tor node. This should allow the detection of transient traffic signals transiting through the measured Tor node.

The adversary could observe a connection to or from the Tor network and use the technique outlined above to detect which nodes it is being relayed through. We propose a more powerful variant of this attack: we assume that the adversary controls a network server that the user to be traced is accessing. This falls within the Tor threat model, and to some extent it is its *raîson d'étre*: users should be able to access network services, that might be interested in identifying them, in an anonymous manner. This corrupt server sends to the user, though the Tor connection, data modulated in a very specific traffic pattern. In our experiments we have used a pattern that consists of sequences of short (a few seconds) bursts of data. Since the attacker knows the input pattern to the Tor network, he can construct a template, and use it to detect whether the traffic volume in Tor nodes is correlated with it.

Figure 1 illustrates the setup necessary for the attacks. In the next section we will present the results of setting up and performing such an attack against the operational Tor network.

3.3 Traffic-analysis methodology

The goal of an attacker is, based on timing data from all nodes on the network, to identify which nodes are carrying the traffic with the pattern injected by the corrupt server. For each node, we performed a test where the stream from the corrupt server went through the target node, and one where the stream did not go through the target node. For the test to be considered a success, the correlation between the traffic modulation and probe latency in the case where the victim stream did go through the target node should be higher than the case where it did not. If this was not the case then either the victim stream did not sufficiently affect the

[3] Acquisti *et al.* go as far as claiming that a multitude of users that do not trust each other have incentives to share the same anonymous network since their traffic is then all mixed together [1].

probe traffic (causing false negatives), or that "echos" of the victim stream propagated through the network and affected the probe stream (causing false positives).

The correlation performed was very simple: the template formed by the modulated traffic from the corrupt server was multiplied with the probe data and the sum of the result was evaluated. In more detail, the template function from the corrupt server $S(t)$ is:

$$S(t) = \begin{cases} 1 & \text{if server is sending at sample number } t \\ 0 & \text{otherwise} \end{cases}$$

The data from the probe is expressed as $L(t)$ which is the measured latency of the target Tor node (in μs) at sample t. $L'(t)$ is the normalised version of the probe data, formed by dividing $L(t)$ by the mean of all samples.

The correlation, c, is the sum of the product between $S(t)$ and $L'(t)$, divided by the number of samples where the corrupt server was sending:

$$c = \frac{\sum S(t) \times L'(t)}{\sum S(t)}$$

A number of improvements could be made to this technique, by using better models of the effect on latency. One obvious addition is to shift the template in time by an estimate of the latency, or to convolve it with an exponential-decay function. Also, quantities other than simple latency could be used, such as a moving variance measure. We have had satisfactory results with the simple technique, and so we leave the design of efficient and optimal transient signal detectors for traffic-analysis for future work.

4 Experimental setup and results

In order to validate the feasibility of the traffic-analysis attacks, we built and evaluated a simple version of our approach. The probe computer used was a standard 800 MHz PC running the Debian GNU/Linux 3.0 operating-system. Tor version 0.0.9 was set up as being a client only (in Tor terminology, an *Onion Proxy*) and modified to choose routes of length one (not including itself), rather than the default of three. In addition to the modified Tor software, the corrupt Tor node consisted of a TCP client and server, both written in C and carefully crafted to avoid the timing properties being interfered with by runtime services such as garbage collection. The interface between the TCP client and the Onion Proxy is achieved using `socat` [37] to connect to the SOCKS interface of Tor. The targeted Tor node then connects back to TCP server running on the same machine.

At regular intervals (in our experiment, every 0.2 seconds) the probe client sent data containing the current system time in microseconds (as reported by `gettimeofday()`) and optional padding. The TCP socket used was configured with the `TCP_NODELAY` option to disable the Nagle algorithm and ensure the data was sent immediately. Also, the TCP stream establishment and each segment sent included a nonce, to avoid port scans from interfering with our results. The probe server recorded the time the segment was sent, and also when the segment was received, and saved both to a file. While this approach limits us to only probing Tor nodes that allow outgoing TCP streams (exit nodes), it could be generalised to all nodes if the attacker controlled a Tor server, even one which was not approved by the Tor directory server maintainers.

The corrupt server was simulated by a TCP server which would send pseudorandomly generated data at as fast a rate as allowed by Tor, for a pseudorandom time period (for our experiment between 10 and 25 seconds), then stop sending for another period (between 30 and 75 seconds). The times at which it stopped and started sending were stored in a file for later analysis. The victim was simulated by a TCP client which would receive data and record the time at which each buffer of data was received and logged this. We recorded the receipt of data to evaluate how much the timing signature of the data was being distorted by Tor, however this data would not be available to an attacker and so was not used in our correlation calculations. The Tor Onion Proxy on the victim node was unmodified since it would not be controlled by the attacker. Again `socat` was used for the interface between the victim client and Tor. The non time-critical parts of the experiment, such as the starting and stopping of programs and the collection of results from the remote machines, were written in Python. The probe server used was hosted in the University of Cambridge Computer Laboratory. The victim and corrupt server were run on PlanetLab [11] nodes in two separate US institutions. The full layout of our system is shown in Figure 1.

In each experimental run, targeting nodes in turn, the procedure was as follows. The probe server would be set to monitor the target node, then after 4 minutes the victim stream would be created so that its first hop would be the node monitored (i.e., the furthest away from the corrupt server, so the timing pattern would be the most distorted). Monitoring by the probe server would continue for another 4 minutes after the victim stream was closed. In order to test for false positives, this test was then repeated, except the victim stream was sent on a path that excluded the target node.

4.1 Results

Data from probing 13 Tor nodes[4] was collected and processed as described in section 3.3 using GNU R [34]. The

[4]Out of the 50 Tor nodes that made up the network, at the time, five were not included so as to check for false positives, and the rest did not carry the probe or victim stream due to being down or because of exit-policy restrictions.

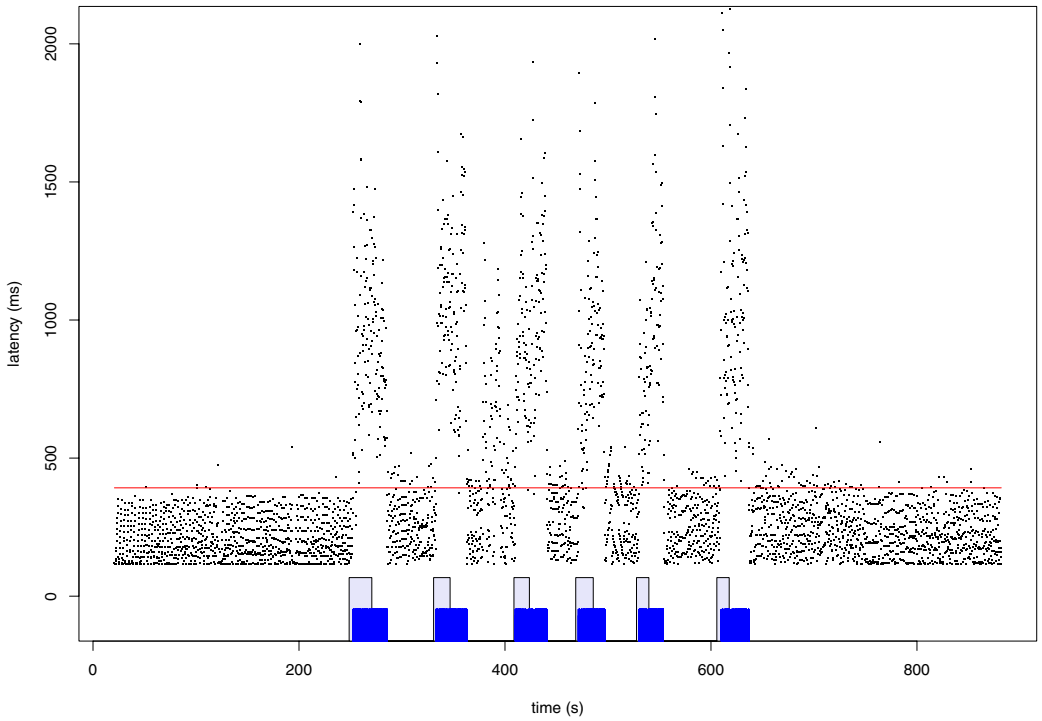


Figure 2. Probe results showing good correlation (Node K)

correlation quality varied, however for all but 2 nodes it correctly differentiated the case where the node was carrying the victim traffic and the case where the traffic flowed through other nodes.

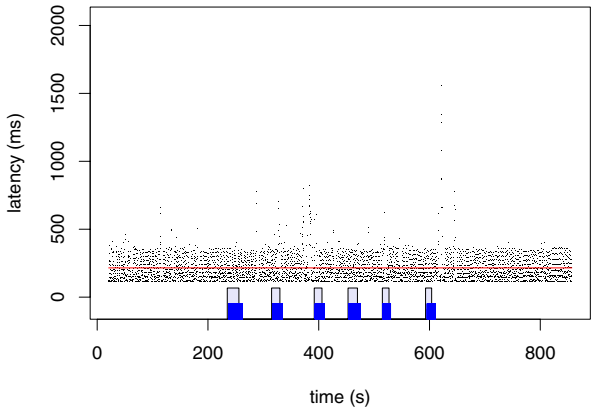
Figure 2 shows a good correlation between probe data in victim traffic. The dots indicate the latency of the probes and the pattern of the victim stream sent is shown at the bottom. The victim stream received is overlaid to show how the pattern is distorted by the network. Finally, the horizontal line indicates the mean latency during the sample period. In contrast, Figure 3(a) shows the same node being monitored when the victim stream was being routed elsewhere. Figure 4 shows a summary of the correlation over all nodes. For each node, the left bar shows the correlation when the victim stream was travelling though the node (false negative test) and the right bar shows the correlation when it was not (false positive test). The two incorrect results (E and M), where the correlation was higher when the traffic was not being sent through the nodes, are highlighted with diagonal shading lines.

None of the results from the false positive test show any obvious correlation to the traffic pattern, which suggests that "echos" of load are not significantly propagated through the network. This means that it should be possible to increase the accuracy of the test simply by running the test for longer than the 6 minutes in our experiments. Other options would be to increase the sampling frequency or to improve the correlation function as suggested in Section 3.3. There appears to be significant room for improvement, as shown in Figure 3(b) which was not correctly identified as being correlated, despite showing visible similarity to the traffic pattern.

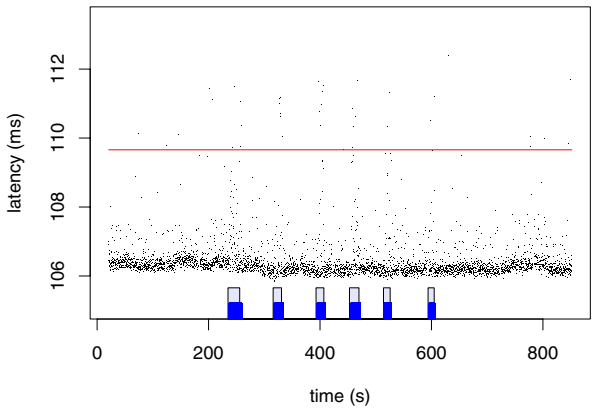
5 Discussion

Our experiments clearly show that Tor streams retain their timing characteristics as they travel through the anonymising network. Furthermore, these characteristics affect other streams in such a way that it is possible to observe them without direct access to the Tor nodes. We have shown that, as a result, it is possible for an attacker to discover which Tor server is being used to inject the traffic stream, and degrade the anonymity provided into being equivalent to a collection of simple proxy servers.

The fact that the timing characteristics of streams are not substantially altered, and can be used to identify the Tor nodes carrying them, comes as no surprise. The low latency requirement of Tor does not allow it to shape the traffic in any way, and would require large amounts of cover traffic for these characteristics to be hidden. Since the attack relies on the indirect measurement of the stream traffic characteristics, a simple minded cover traffic strategy – that only filled the links with cover traffic – would not work. The cover traffic should not only fill the links, to confuse a di-



(a) Probe results without traffic pattern (Node K)



(b) False negative (Node E)

Figure 3. Results without positive correlation

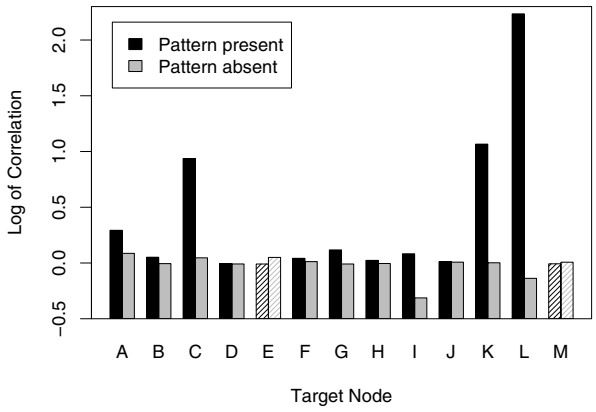


Figure 4. Summary of correlation

rect observer, but also make sure that it confuses indirect measurements as presented in this paper. When designing such a cover traffic strategy it is also important to keep in mind Wei Dai's attack [5]: an adversary can try to fill the victim node with their own traffic, trying to eliminate all the cover traffic. This is very similar to the indirect measurement of traffic load that we have performed, and shows that Tor would have to use cover traffic all the time, and not simply when there is not enough genuine traffic to fill all the links.

The interference between the timing characteristics of different streams is both a benefit for anonymity and at the same time, a vehicle for attack. One would hope that streams on the same Tor node would interfere with each other to such a degree that it is impossible to differentiate them from each other, therefore making it difficult for an attacker to know which one to trace, which is not the case. This perfect interference should create 'echos' of the traced stream throughout the network and cause any traffic-analysis to produce a lot of false positives. Nevertheless, streams relayed by the same Tor node interfere with each other just enough to leak information to an adversary controlled stream and thus allow the measurement of the load of the node. In some sense, Tor exhibits the worst possible behaviour: not enough interference to destroy individual stream characteristics, yet enough to allow the remote measurement of the node's load.

Two strategies could be employed to protect Tor: *perfect interference* and *non-interference*. Perfect interference amongst all streams relayed through the same node means that the output streams all have the same shape. This means that the adversary will have a very difficult time determining which output stream corresponds to the input stream to be traced. Since Tor relies on a sequence of relays, it would be interesting to study how long paths would need to be so that streams would interfere with each other in such a way that all the outputs of the network would have the same characteristic. Note, that since the vehicle of this entanglement is traffic streams, one needs to assess how many other streams have been *touched*, by being relayed through the same node, and therefore might become indistinguishable with. A second strategy for implementing perfect interference is to shape the stream traffic into another random shape, either the same for all streams or different for each of them, yet unlinkable to any particular input stream. Causality means that this shaping can only be done by delaying the packets (you cannot send a packet received at time t out in the network at time $t-1$). Therefore any traffic shaping strategy will inevitably increase the latency of the communication.

Non-interference between streams can also be used to protect against our attacks. This would eliminate the covert channel we use to remotely infer the timing of streams on

Tor nodes. This property could be very difficult to implement in practice. All streams share a lot of common resources: the Tor packet scheduler, the TCP/IP stack, the physical network and the CPU of the Tor node. There is an established methodology for discovering and eliminating covert channels [22], and it is recognised as a difficult problem. Even hardened systems exhibit covert channels of >1 bit/s. These might be tolerable for multilevel secure systems, but would be devastating for anonymous communication systems – in a few seconds an adversary could distinguish the victim's communication amongst all of the streams. This is, because there are inherently fewer actors to identify in an anonymous communication system than possible cryptographic keys or possible documents in a multilevel system.

5.1 Linkability attack

A variant of our attack can also be used to determine whether two streams coming out of the same Tor node belong to the same initiator. Remember that Tor uses the same connection to route many streams from the same initiator – we can use this property to test whether two streams coming out of the Tor network and accessing two corrupt servers belong to the same user. We determine, using the main attack presented, the Tor nodes that route the two streams. While the probability that two different initiators use the same exit node is $1/N$, the probability that the full path of three nodes is the same, given that each node was chosen randomly, is only about $1/N^3$. Therefore the fact that two streams use the same path strongly indicates that they belong to the same initiator. Testing whether a second stream belongs to the same initiator as an already traced stream, is cheaper than performing the analysis to start with. The attacker already knows the two nodes on the path of the first stream and can just test them to confirm that the second stream belongs to the same connection and initiator.

This attack is especially interesting since it shows that Tor makes it easier to link two events to the same initiator than a simple proxy. These events exhibit a particular signature, that a simple proxy does not have, namely their path through Tor, which can be uncovered using our attacks. If our attacks are not eliminated, augmenting the length of the Tor path, conventionally thought to increase security, would make it even more vulnerable to this attack. The longer the common chain of Tor nodes two connections share, the less likely it is that they belong to different users. The same is true for the total number of nodes: it is conventionally believed that more nodes is better for anonymity, but a larger population of nodes makes common chains less common and allows for more precise identification. The fact that empty connections are short lived, and that a stream can exit at any node in the path might make such attacks slightly less reliable, but does not solve the underlying problem.

5.2 Variants of our attack

The attack we have presented so far relies on a probe stream being routed through a Tor node to detect the timing of a modulated communication stream from a corrupt server. Using the principle that timing information leaks from one stream to the other, we could conceive quite a few variants of this attack.

Firstly, we could modulate the probe traffic that is sent to the victim Tor node in a loop and try to detect the effects on requests sent by the initiator of the anonymous communications. In cases where the traffic is mainly from the victim to the server, the corrupt server does not have much opportunity to significantly modulate the traffic load, so this may be the only option. The difficulty with this approach is that the normal method of probing all Tor nodes in the network simultaneously is problematic, since the modulation of the victim stream will be the combination of the load induced on all three of the Tor nodes along the path.

An alternative would be to probe each Tor node in turn, but for a given stream lifetime, this would reduce the probe duration and thus accuracy. Instead, the attacker could probe all nodes, but using a different, "orthogonal" pattern for each node, so the resulting combination observed can be decomposed into the original components. An adaptive attack could be mounted by, for example, probing all nodes in the network briefly and observing the result. While this short test will have a poor accuracy, it could be used to eliminate some nodes known not to be on the path. The remaining nodes could be probed again (possibly with a longer pattern) further eliminating more nodes. This process is repeated until only three nodes remain. Another option is to probe some fraction of the nodes at one time; if the resulting stream is affected then at least one node on the path must be in that fraction, if not then all nodes in that group can be eliminated from consideration. The above techniques could be combined together.

If the attacker does not have total control over the corrupt server and can only monitor the link but not modify the load, then there are still variants of our attack that can be used. One is to use the probe-modulation variant above. Another is to take advantage of a known traffic pattern observed on the corrupt server. Since this pattern cannot be optimised, the attack may take longer to reach a result, but the traffic may still be suitable for inducing an observable effect on the intermediary Tor nodes. One could mount attacks without any monitoring if the traffic being sought has known characteristics, which can be observed on the Tor nodes it is being sent through.

If an attacker can neither directly observe nor change the traffic on the corrupt server, it may be possible to infer the

load by the attacker using the server and observing the response time, in a similar way to how the Tor nodes are monitored. An attacker could also alter the load of the destination server by modulating a denial of service (DoS) attack on it. When the DoS attack is running, the load of the victim connection should be decreased and so decrease the load of the Tor nodes on the path it is travelling. Recent research [25] has shown that by exploiting the TCP back-off algorithm, it is possible to mount an effective and difficult to trace denial of service attack without large resources. Techniques similar to this could also be used in the probe-modulation variant and to design better patterns for the corrupt server to send, so the influence on other Tor connections through each node is maximised.

The above attacks allow the nodes used to relay a particular stream to be identified, which already severely degrades anonymity. In order to identify the initiator, the attacker must look at incoming connections to all three nodes. If resources are limited, then it would be desirable to identify the entry node, to target monitoring. This could be done by estimating how much the induced traffic pattern is shifted as it travels through the network. We did not perform this because our probe sampling frequency was too low (every 0.2 seconds) to show effects on the scale of typical Tor latency. However, once an attacker has found the three nodes on the connection path, he could probe these at higher frequency, to watch for the precise timing of the pattern. Another possibility is to look at the distortion of the induced pattern. As it progresses through the network, noise will be added, so it is likely the node showing the 3rd best correlation is the entry node.

5.3 Attack costs

Our attack is feasible for the adversary anticipated by the Tor designers and can be mounted without direct access to the communication links of the Tor nodes. To reliably perform the attacks, each Tor node in the network should be observed all the time. Assuming there are N Tor nodes, we therefore require N probe streams going through them – a set of machines, or a single machine connected to the Internet with a low-latency connection suffices. This means that the cost of the attack is $\mathrm{O}(N)$ since it increases linearly with the number of nodes in the Tor network.

Note that higher volumes of traffic in the Tor network would make the quality of the observation poorer and degrade the performance of the attack. Therefore, there is a hidden cost that has yet to be estimated, which rises with the number of streams relayed by each node. At the same time, any increase in latency that might hinder the attacker, by making the remote measurements less precise, will inevitably also increase the latency of genuine Tor traffic. Therefore we are again confronted with the choice of increasing the security of the system, versus keeping the latency as low as possible.

Aside from observing the nodes, an adversary is assumed to have the ability to modulate the replies of a dishonest accessed server. The simplest way of doing this is by deceiving anonymous users and making them access an attacker controlled server. This way, arbitrary data streams can be sent back and forth, and get detected. Where Tor is used to access an HTTP [19] (web) service, the attacks can be mounted much more simply, by including *traffic-analysis bugs* within the page, in the same way as web bugs [3, 12] are embedded today. These initiate a request for an invisible resource that, due to the HTTP architecture, can have an unconstrained traffic shape and characteristic. The attacker can then simply try to detect them, using our attack as described.

5.4 Understanding the traffic artifacts

As described earlier, our attack is based on the fact that the traffic characteristics of streams are hardly affected by Tor, and that these characteristics leak into other streams sufficiently so that they can be remotely estimated. It is interesting to study how these processes are taking place in practice.

Streams interfere with each other at all levels. At the highest level, Tor routers relay a set of streams using a non-blocking polling strategy presented in Figure 5. Each of the relayed streams is polled to see if any data is available to be relayed. If data is available, it is processed, otherwise the next stream is considered. This strategy in itself ensures that a different stream being relayed will delay the probe stream, and leak information about the latency of the node.

Aside from the polling strategy, streams relayed by Tor share the operating-system resources, the TCP/IP stack, the network and the hardware of the Tor node. The operating-system scheduler could influence the timing of the streams by allocating more resources when the node relays more traffic, or less when then node is mostly waiting for more input. Memory management operations could also take more time if data is being routed. The TCP protocol would back-off if the link is congested. Finally the network has a fixed capacity, and has to be shared amongst connections. All of these contribute to the latency of the probe data being influenced by the volume of data relayed by the Tor node. Figure 2 illustrates this. It is clear that the probe data (top) can be used to infer the volume of the actual traffic sent (bottom).

Other traffic patterns have been observed in the measurement data that are not yet fully explained. These could be artifacts of the measurement technique, that by its indirect nature can only act as an estimate of the load, or genuine latency introduced by the remote Tor node. We present here

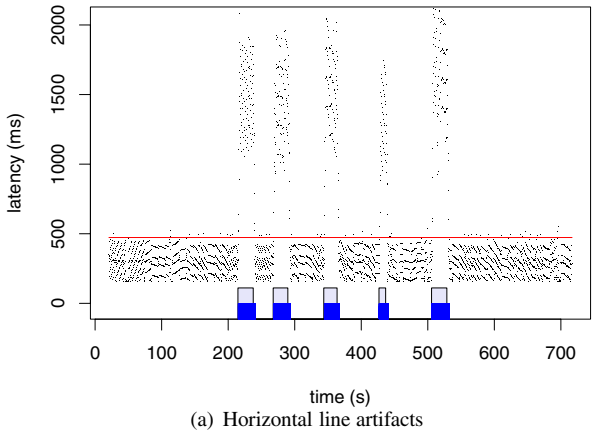
```
/* Tor main loop */
  for(;;) {
    timeout = prepare_for_poll();
    ...
    /* poll until we have an event,
       or the second ends */
    poll_result = tor_poll(poll_array, nfds, timeout);
    ...
    /* do all the reads and errors first,
       so we can detect closed sockets */
    for(i=0;i<nfds;i++)
      /* this also marks broken connections */
      conn_read(i);

    /* then do the writes */
    for(i=0;i<nfds;i++)
      conn_write(i);

    /* any of the conns need to be closed now? */
    for(i=0;i<nfds;i++)
      conn_close_if_marked(i);
    ...
  }

/* Read from connection */
static void conn_read(int i) {
  ...
  if(!(poll_array[i].revents & (POLLIN|POLLHUP|POLLERR)))
    if(!connection_is_reading(conn) ||
       !connection_has_pending_tls_data(conn))
      return; /* this conn should not read */
    ...
    connection_handle_read(conn) < 0) {
    ...
}
```

Figure 5. The Tor 0.0.9 polling code

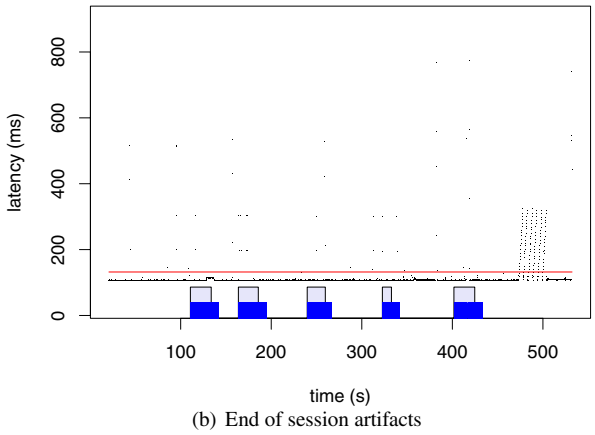
two examples that could be used to perform traffic-analysis, if they were linked with particular states of the Tor nodes.

Figure 6(a) shows the results of probes against an exit node in the Tor network. Again, the top graph represents the latency over time of probe traffic, while the bottom represents the times the corrupt server was sending data. Note that the latency of the probes seems to be quantised into four or five bands – even when a high volume of traffic is injected. The quantisation could be explained by the lack of precision or quantisation of the measurement process. Another explanation is that the bands are formed by measuring the node when one, two, three or four other streams are being served at the time. This seems to match with the experimental data: only four clusters are visible when the corrupt server is not relayed, and five when the stream is present. This technique could be developed to extract information about the number of streams relayed – and in turn used to infer the beginning and termination of a stream.

Figure 6(b) illustrates a completely different type of traffic pattern. After the last burst of traffic from the corrupt server (bottom) the latency of the probe traffic exhibits a very peculiar pattern, it goes up six times each time falling back into the average latency. This event has been observed many times in conjunction with the closure of a Tor connection, and could be due to time devoted in tearing down connections. If such events can be observed, the connection tear down could be tracked through the network to gain information about the route of a connection.



(a) Horizontal line artifacts



(b) End of session artifacts

Figure 6. Artifacts under investigation

Aside from the precise load information extracted from the probe traffic, these secondary traffic artifacts could also be used to perform traffic analysis and assess which Tor server is being used to relay the target traffic. Therefore a strategy to eliminate information leakage into other streams should also try to eliminate these artifacts.

6 Conclusions

We have presented an attack against Tor, a deployed and well used, anonymising protocol. This attack can be performed by a modest adversary, using powers well within the restricted Tor threat model. In fact, we show that the anonymising network itself can be used to route probe traffic and gather information otherwise available only to a global passive adversary.

In November 2004 we performed extensive experiments on current Tor nodes and found them to be susceptible to

our attack. This does not give us the ability to trace the actual originator of the communication, since we do not have the ability to observe who is connected to a Tor node. Nevertheless our attacks greatly degrade the anonymity provided by Tor, by allowing adversaries to discover the path of a Tor connection and thereby reducing the protection to the level provided by a collection of simple proxy servers. We expect the same attack to be usable against other low-latency anonymising network designs, since none of them have been specially hardened against it.

Furthermore, since Tor reuses the same path for multiple streams within a short time interval, our attacks allow different operations to be linked to the same initiator with greater certainty. The observable path of each stream can act as an identifier or identity that links streams amongst themselves and to the initiator – a property that makes Tor weaker than a simple proxy when it comes to protecting the unlinkability of actions.

We discussed some strategies that could be used to protect Tor against our attacks. They all, to some degree, involve an increase in the latency of the communication. They also highlight the need for a full covert-channel analysis of such anonymising networks, to assess whether any information that could be used for traffic-analysis is leaked to other streams that are potentially observable by an adversary.

This attack brings the field of anonymous communications even closer to more traditional computer security disciplines. On one hand we show that the literature on covert channel analysis and elimination is directly applicable and necessary to truly secure Tor. On the other hand, our attack relies on using Tor nodes as oracles that disclose their load – therefore not requiring a global observer. Similar techniques have been used in the past in breaking cryptographic protocols, by using and combining the services they provide. It is the first time that such techniques are applied for traffic-analysis of anonymous communication systems.

Acknowledgements

Paul Syverson and Roger Dingledine, part of the team that designed Tor, have provided us with very useful feedback and information concerning the architecture of Tor and the true impact of our attacks. This work would not have been possible without the dedication of the volunteers running Tor nodes.

George Danezis is supported by the Cambridge-MIT Institute (CMI) project on 'Third generation peer-to-peer networks' and part of this work was done while visiting MIT CSAIL and the Brown University Watermyn Coop. Steven J. Murdoch is supported by a scholarship from the Carnegie Trust for the Universities of Scotland.

References

[1] A. Acquisti, R. Dingledine, and P. F. Syverson:. On the economics of anonymity. In R. N. Wright, editor, *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 84–102. Springer, 2003.

[2] D. Agrawal, D. Kesdogan, and S. Penz. Probabilistic treatment of mixes to hamper traffic analysis. In *IEEE Symposium on Security and Privacy*, pages 16–27, Berkeley, CA, USA, May 2003. IEEE Computer Society.

[3] A. Alsaid and D. Martin. Detecting web bugs with bugnosis: Privacy advocacy through education. In *Privacy Enhancing Technologies (PET 2002)*, San Francisco, CA, May 2002.

[4] A. Back, I. Goldberg, and A. Shostack. Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., May 2001.

[5] A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In I. S. Moskowitz, editor, *Information Hiding workshop (IH 2001)*, volume 2137 of *LNCS*, pages 245–257. Springer-Verlag, April 2001.

[6] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 115–129. Springer-Verlag, July 2000.

[7] A. Blum, D. Song, and S. Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004*, Sophia Antipolis, France, September 2004.

[8] R. Bohme, G. Danezis, C. Diaz, S. Kopsell, and A. Pfitzmann. Mix cascades vs. peer-to-peer: Is one concept superior? In *Privacy Enhancing Technologies (PET 2004)*, Toronto, Canada, May 2004.

[9] P. Boucher, A. Shostack, and I. Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

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

[11] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3), July 2003.

[12] R. Clayton, G. Danezis, and M. G. Kuhn. Real world patterns of failure in anonymity systems. In I. S. Moskowitz, editor, *Information Hiding, 4th International Workshop*, volume 2137 of *LNCS*, pages 230–245. Springer-Verlag, April 2001.

[13] G. Danezis. Statistical disclosure attacks. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.

[14] G. Danezis. The traffic analysis of continuous-time mixes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, May 2004.

[15] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *IEEE Symposium on Security and Privacy*, Berkeley, CA, 11-14 May 2003.

[16] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[17] R. Dingledine and N. Mathewson. Tor spec. Technical report, The Free Haven Project, October 20 2004. `http://www.freehaven.net/tor/cvs/doc/tor-spec.txt`.

[18] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[19] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, Network Working Group, June 1999.

[20] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In V. Atluri, editor, *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 193–206, Washington, DC, November 2002. ACM.

[21] M. J. Freedman, E. Sit, J. Cates, and R. Morris. Introducing tarzan, a peer-to-peer anonymizing network layer. In P. Druschel, M. F. Kaashoek, and A. I. T. Rowstron, editors, *International workshop on Peer-to-Peer Systems (IPTPS)*, volume 2429 of *LNCS*, pages 121–129, Cambridge, MA, March 2002. Springer-Verlag.

[22] V. D. Gligor. *A Guide to Understanding Covert Channel Analysis of Trusted Systems*. National Computer Security Center, 1993. NCSC-TG-030, Version 1.

[23] I. Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, UC Berkeley, December 2000.

[24] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[25] Guirguis, Mina, Bestavros, Azer, and I. Matta. Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources. In *Proceedings of ICNP'04: The 12th IEEE International Conference on Network Protocols*, Berlin, Germany, October 2004.

[26] C. Gülcü and G. Tsudik. Mixing E-mail with Babel. In *Network and Distributed Security Symposium — NDSS '96*, pages 2–16, San Diego, California, February 1996. IEEE.

[27] D. Kesdogan, D. Agrawal, and S. Penz. Limits of anonymity in open environments. In F. A. P. Petitcolas, editor, *Information Hiding workshop (IH 2002)*, volume 2578 of *LNCS*, pages 53–69, Noordwijkerhout, The Netherlands, 7-9 October 2002. Springer-Verlag.

[28] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright. Timing attacks in low-latency mix-based systems. In A. Juels, editor, *Proceedings of Financial Cryptography (FC '04)*. Springer-Verlag, LNCS 3110, February 2004.

[29] N. Mathewson and R. Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, LNCS, May 2004.

[30] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol version 2. Technical report, Network Working Group, May 25 2004. Internet-Draft.

[31] I. S. Moskowitz, R. E. Newman, D. P. Crepeau, and A. R. Miller. Covert channels and anonymizing networks. In *Workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, October 2003.

[32] I. S. Moskowitz, R. E. Newman, and P. F. Syverson. Quasi-anonymous channels. In *Communication, Network, and Information Security (CNIS 2003)*, New York, USA, 10–12 December 2003.

[33] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In W. Effelsberg, H. W. Meuer, and G. Müller, editors, *GI/ITG Conference on Communication in Distributed Systems*, volume 267 of *Informatik-Fachberichte*, pages 451–463. Springer-Verlag, February 1991.

[34] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. ISBN 3-900051-07-0 `http://www.R-project.org/`.

[35] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[36] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[37] G. Rieger et al. socat – multipurpose relay. `http://www.dest-unreach.org/socat/`.

[38] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. In *European Symposium on Research in Computer Security (ESORICS 2003)*, Gjovik, Norway, 13–15 October 2003.

[39] P. F. Syverson, G. Tsudik, M. G. Reed, and C. E. Landwehr. Towards an analysis of onion routing security. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 96–114, Berkeley, CA, USA, 25-26 July 2000. Springer-Verlag.

[40] J. Young and E. M. On obtaining "lawful interception" documents. `http://www.quintessenz.org/etsi`.

[41] Y. Zhang and V. Paxson. Detecting stepping stones. In *9th USENIX Security Symposium*, August 2000.

[42] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, May 2004.