Revisiting Circuit Clogging Attacks on Tor

Eric Chan-Tin, Jiyoung Shin and Jiangmin Yu
Department of Computer Science
Oklahoma State University
{chantin, jiyoung, jiangmy}@cs.okstate.edu

***Abstract*—Tor is a popular anonymity-providing network used by over $500,000$ users daily. The Tor network is made up of volunteer relays. To anonymously connect to a server, a user first creates a circuit, consisting of three relays, and routes traffic through these proxies before connecting to the server. The client is thus hidden from the server through three Tor proxies. If the three Tor proxies used by the client could be identified, the anonymity of the client would be reduced. One particular way of identifying the three Tor relays in a circuit is to perform a circuit clogging attack. This attack requires the client to connect to a malicious server (malicious content, such as an advertising frame, can be hosted on a popular server). The malicious server alternates between sending bursts of data and sending little traffic. During the burst period, the three relays used in the circuit will take longer to relay traffic due to the increase in processing time for the extra messages. If Tor relays are continuously monitored through network latency probes, an increase in network latency indicates that this Tor relay is likely being used in that circuit. We show, through experiments on the real Tor network, that the Tor relays in a circuit can be identified. A detection scheme is also proposed for clients to determine whether a circuit clogging attack is happening. The costs for both the attack and the detection mechanism are small and feasible in the current Tor network.**

***Keywords*—Tor, Anonymity, Attack, Circuit Clogging, Privacy.**

I. Introduction

Anonymity and censorship-resistant systems are becoming more prominent due to the recent unrest in Egypt and other countries [1]–[3]. To avoid prosecution and identification by oppressive authorities, citizens of these countries require a system that allow them to maintain their anonymity on the Internet. Various anonymization-providing services exist nowadays [4]–[10], with Tor [9], [10] being one of the most popular ones with over $500,000$ users [11]. Tor is known as a "low-latency" service as interactive applications such as web-browsing, chat, and remote connections (VPN and ssh for example) can be used on top of Tor.

A client, who wants to connect to a remote server anonymously, uses Tor as a proxy. All the connections and messages go through Tor first, then to the server. Thus, the server believes that the connection is coming from Tor, which hides who the real client is. The Tor system is made up of a network of relays. Each *relay* is a volunteer machine. The client picks three relays from the network to form a circuit: the *entry* node, *middle* node, and *exit* node. The client establishes a connection with the entry node, then using the entry node as a proxy, extends that connection to the middle node, and finally, extends the same connection to the exit node. Then, the client hops through each of the relays to connect to the server. In this case, the server believes the connection is coming from the exit relay. Only the entry relay knows who the client is, but believes the destination is the middle relay.

The Tor project started around 2004 and has been growing in popularity since. It is used by citizens of oppressive regimes, dissidents, whistle-blowers, journalists, and governments' military for anonymous communication. Currently, there are over $500,000$ users [11] in Tor and over $3,000$ relay nodes [11]. The premise of anonymity relies on the three relays used by the client to be non-colluding (especially the entry and exit nodes). Moreover, the identity of the three relays used by a client to connect to a server is hidden. If an adversary could somehow identify the three relays used by a client, this breaks some of the anonymity of the client as it reveals which three Tor relays the client chose. It has been shown in [12], [13] that after the Tor relays in the circuit have been identified, the identity of the client is also leaked. Thus, deanonymizing the three relays used by a client is the first step towards identifying which client is communicating with which server. This has a huge impact on Tor as the anonymity of any Tor user can be compromised.

There exists many attacks [14]–[22] in the literature on deanonymizing the client and the three Tor relays, such as timing attacks, network flow attacks, and circuit clogging attacks. In this paper, we revisit the *circuit clogging* attack described in [17] and show that it is still applicable in the current Tor network. The three relays used in Tor is called a circuit. In a circuit clogging attack, the premise is that a client creates a circuit and connects to a server using that circuit. The server or parts of the content of the server (for example an advertising frame) is malicious. The malicious content alternates between sending a lot of data and sending very little data. If there were a direct connection between the client and the server, there were no issue with the extra data. Since each Tor relay can be part of multiple circuits, when even one circuit is busy delivering a lot of data, that Tor relay will slow down. In this particular attack, the goal is to identify the three relays used by a client. Every relay in the Tor network is continuously monitored, for example, by creating a 1-hop circuit or through system pings. When a relay slows down during the period of sending burst data (sending a lot of data), its monitoring will show a spike in network latency. The three Tor relays that show an increase in network latency in the monitoring are most likely the three relays used in the circuit by the client.

The first described circuit clogging attack [17] on Tor was performed a few years ago when Tor was still in its infancy. There were only 50 Tor relays at the time, of which 13 were used to demonstrate the attack. Now, there are over $3,000$ relays and Tor is heavily used for web traffic and bulk downloads, such as BitTorrent [23] or peer-to-peer file download traffic. We revisit this attack and show that this circuit clogging attack to identify the three nodes in a Tor circuit is still applicable today. Although the authors in [19] claimed that this attack is no longer possible, their experiments were limited. We performed a full-scale experiment with real Tor nodes. Our proposed circuit clogging attack and network experiments are slightly different from the experimental setup in [17] or [19], as shown in Section III. Section V shows that a distinct pattern can be observed for the Tor relays in the circuit versus Tor relays not in the circuit. The pattern shows that during the burst period, an increase in network latency can be observed, while during the "sleep" period, the network latency decreases to normal. Through experiments performed on the real Tor network, all the Tor relays used in a circuit under a circuit clogging attack can be identified over 78% of the time. The bandwidth required to perform the circuit clogging attack is less than 30KB/second, and the bandwidth needed to monitor all the $3,000$ Tor relays is 391KB/second.

A detection scheme for clients is also proposed. The detection mechanism probes all the user's created circuits for timing information. If it detects a high and unexpected increase in network latency, the user can disconnect from the server and destroy the affected circuit. Experiments indicate that the proposed scheme can detect over 85% of attacks. The scheme also incurs a very low overhead, requiring less than 3KB/second extra bandwidth to operate. The scheme proposed detects when a possible circuit-clogging attack is occurring; it does not prevent circuit-clogging attacks.

The rest of the paper is organized as follows. Section II gives a more detailed description of how Tor works. An outline of related work is also provided in Section II. In Section III, we describe the specific circuit clogging attack from [17] and how our experiment is set up. A possible solution to identify when a clogging attack is taking place, is given in Section IV. The results of our experiment are shown in Section V. Finally, in Section VI, we discuss future work and conclude.

II. Background

We provide a more detailed description of how Tor works, as well as, a threat model for the attack. The different attacks on Tor to identify the client or the circuit are then outlined.

A. Tor

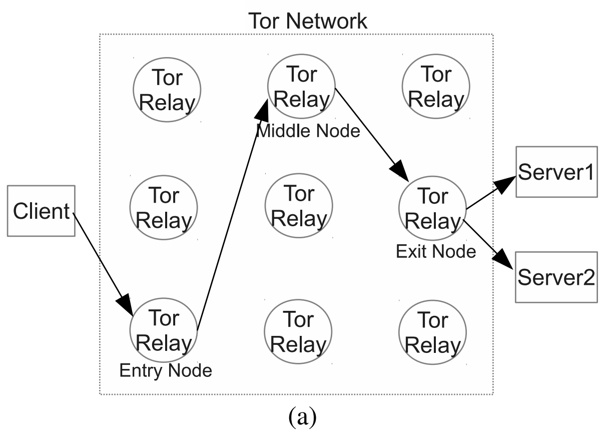
Tor [9], [10], released around 2004, is the second generation onion router [24]. The Tor network consists of three main entities: directory servers, relays, and clients. The directory servers are trusted and they keep track of all the relays in the network. Every relay contacts the directory servers to register itself as a relay and upload its key and configuration, such as open ports, and advertised bandwidth. The directory servers regularly form a consensus of all the relays, and sign the consensus document. The servers also monitor the relays' advertised bandwidths. Each client contacts the directory servers to download the consensus document to obtain a list of all the relays and their status. A client can also serve as a relay. The directory servers are currently hard-coded in Tor, whereas the relays are volunteer machines.

Tor works as follows. A client contacts the directory servers and downloads the consensus document of all the active relays. The client then needs to construct a circuit to connect to servers on the Internet. To contact a server, the client proxies the connection through a circuit. To build a circuit, the client randomly selects three Tor relays: an *entry* node, a *middle* node, and an *exit* node. The client first establishes an encrypted connection with the entry node. The client then extends that circuit, by going through the entry node, to establish an encrypted connection with the middle node. The client further extends the circuit by establishing another encrypted connection with the exit relay. Encryption and authentication of each relay are possible since every relay's key is part of the consensus document. Once the circuit is built, the client uses it to proxy connections over, to contact Internet servers. Circuits are built using onion routing, such that the connection from the client to each of the three relays is onion-encrypted. None of the relays see what the client is sending to the other relays. For example, the entry node cannot tell that the client is extending the circuit to a particular exit node, as the message to the middle node is encrypted with the middle node only. Figure 1 shows graphically how circuits and onion encryption work.

Anonymity is achieved since no entity in the network knows who the client and server are. The entry node only knows that the client is communicating with the middle node. The middle node knows that a machine (entry node) is communicating with another machine (exit node). The middle node cannot tell that it is the middle node of a circuit. Similarly, the exit relay knows that a machine (middle node) is communicating with a server. Finally, the server believes that the connection is coming from the exit relay. To prevent an adversary from potentially controlling a large fraction of entry nodes in all circuits, a client randomly selects three Tor relays as its entry guard nodes. This means that the client will only pick one of these three entry guards as its first entry node in any circuit the client creates. The middle and exit nodes in the circuit are still randomly selected from the consensus document. It is noted that there are over $3,000$ relays in the Tor network, with about 800 of these relays marked as exit relays.

Due to the heavy cost, in terms of encryption and processing, in creating circuits, a client can use one circuit for multiple connections (or streams) to different servers. By default, each circuit is used for at least 10 minutes before it is recycled and a new one created. If a connection is still active in a circuit, that circuit is not destroyed but no new connections are created through that circuit. Circuits can be created in parallel to increase efficiency. Rate limiting is applied end-to-end using TCP. However, Tor also implements its own rate limiting. Every connection is associated with a token bucket. Once a connection runs out of tokens, no new packets can be accepted or delivered until previous packets have been received.

In Tor, every packet is a cell, and each cell's size is 512 bytes. By default, for each relay, the size of a circuit is $1,000$ cells while the size of a stream is 500 cells. Within each circuit, cells are scheduled in a First-In-First-Out (FIFO) fashion. There are currently two scheduling algorithms that can be used within each relay to decide which circuit's cells get processed next. The original algorithm uses round-robin to select the next



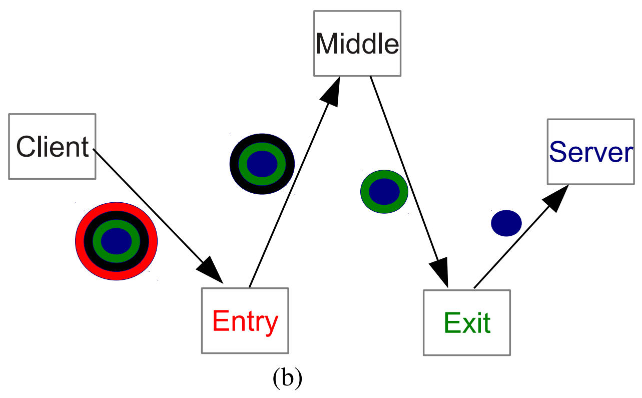


Fig. 1. (a) Circuit creation in Tor: the client randomly selects three Tor relays in the Tor network to be its entry, middle, and exit nodes. The same circuit can be used for different connections (called streams) to different servers. (b) Onion routing and encryption: A client connects to the server through Tor. The client wants to send a message (blue) to the server. The client first adds a layer to the message, encrypted with the exit node's key (green). Then the client adds a second layer, encrypted with the middle node's key (black), and finally, that whole message is encrypted with the entry node's key (red). At each step, each relay peels off its layer and forwards the message to the next relay.

cell to process. The round-robin is performed among all the circuits. The newer algorithm uses an exponential weighted moving average (EWMA) to determine which circuit to process the next cell from. Each cell is weighted exponentially based on the number of cells in each circuit. Circuits with fewer cells thus have higher priority. This scheduling algorithm is the recommended one from the consensus document.

B. *Threat Model*

The threat model in this paper is the same as that used in the original Tor paper [10] and in the current literature on Tor. The adversary is only local to each entity or ISP in Tor, and is assumed to not have global capabilities. The adversary can be either passive or active. Attackers can drop packets, relay packets, modify packets (note that cells in Tor are encrypted), delay packets, and passively eavesdrop on all packets. An adversary can listen on the communications at multiple relays, clients, and/or servers, but not all of them. Moreover, the Tor relays, clients, and servers can be malicious but the directory servers are honest and trusted.

C. *Related Work*

The current literature contains many publications describing attacks on Tor and other anonymity-providing services [12], [15]–[20], [25]. Most of these attacks use some sort of timing analysis to determine which relays, clients, servers, or paths are used. The simplest attack in Tor, is for the adversary to be in control of both the entry and exit nodes in a circuit. Using timing information, the adversary can correlate that the two nodes are actually part of the same circuit, and from there, can deduce that a particular client is communicating with a particular server. If the adversary controls f relays out of n relays, then the probability that the entry and exit nodes selected belong to the adversary is $(\frac{f}{n})^2$. This probability is even lower with the introduction of entry guards.

In 2005, Murdoch and Danezis [17] showed a low-cost traffic analysis attack on Tor, which allowed them to identify the three relays used in a circuit. The attack is a variant of the circuit clogging attack, where the adversary attempts to overload the three relays used, which in turn, increases the network latency timing of these three relays. Since all other relays are unaffected, an increase in timing of network latency of three nodes, leads the attacker to conclude that these three nodes are the three relays in the circuit. This attack assumes that the server or part of the content being served is controlled by the adversary. From contacting the directory servers, the adversary also knows all the Tor relays in the network. It can then monitor all the Tor relays for timing information. The adversary runs a probe client and server. It then constructs a one-hop (one-node) circuit to each Tor relay. The probe client uses that circuit to connect to the probe server. The probe keeps each circuit alive and periodically performs a measurement of the network latency of each circuit. Once the malicious server receives a connection from Tor, it starts the circuit clogging attack. The attack consists of a period of low traffic, followed by a period of high traffic, followed by a period of low traffic, and so on. During the period of high traffic, the three relays in the circuit become overloaded as they have to process more cells. This, in turn, leads to an increase in network latency measured by the probe. As all other relays are unaffected, the three relays, experiencing an increase in network latency, are likely the three nodes in the circuit. Once the circuit is identified, the attack can be further extended [12], [13] to narrow down the list of possible clients which created that circuit. The authors of [12], [13] measured the network latency of the whole circuit to calculate the network latency between the entry node and the server. From there, it was trivial to calculate the network latency between the victim and the entry node. The list of possible victims can then be narrowed down using that extra information (network latency) obtained.

However, the investigation of this low-cost circuit clogging attack was originally performed in 2005, when Tor consisted of only 50 relays and few users. The primary use of Tor then, was for remote connections, web traffic, and anonymous chats. Experiments were performed using 13 out of the 50 relays. Nowadays, Tor consists of thousands of relays, with hundreds of thousands of users. Usage of Tor is also more diverse, consisting of short and small web traffic and longer and more bandwidth-intensive file downloads, such as BitTorrent [23]. It is widely believed that this attack is no longer possible due to the changes in the Tor network mentioned above, such as increased traffic and more diverse traffic. As far as we know, there was no specific countermeasure implemented, other than a better scheduling algorithm for circuits. Evans *et al.* [19] presented a variant of the clogging attack using *long paths*. In that paper, they replicated the Murdoch and Danezis attack and showed that it was no longer applicable in the Tor network of 2008, due to the noise in the Tor network and the loss of the effects of the attack in the noise.

We show that a variant of the circuit clogging attack from the 2005 paper [17] is still possible today. This allows an adversary to identify the three relays used in a circuit, and can be a stepping stone to actually identify clients of the Tor network. We note that in our proposed attack, only two out of the three relays are public Tor relays; the third one is under the adversary's control. Our attack is more comprehensive than previous attacks [17], [19] as the experiments include all the Tor relays, and the periods of on/off attack are longer. The attack is also performed on the real Tor network. The experiment setup is also slightly different, as shown in the next section.

III. Attack Design

We now describe our circuit clogging attack. It is very similar to the clogging attack from Murdoch and Danezis [17]. The four entities are the *client* (victim), *burst server*, *probe*, and *Tor*. The client refers to any user connecting to servers anonymously, using the Tor network. *Tor* represents the three relays used in the circuit created by the client. The three Tor relays are randomly selected from the real Tor network. The burst server is a malicious server the client connects to, either directly or indirectly. For example, the burst server can serve advertisement content when the client visits a major popular website. The goal of the burst server is to introduce enough traffic in the connection with the client to identify the three Tor relays used in the circuit. The probe entity is controlled by the same adversary that controls the burst server; the objective of the probe is to perform network measurements of all the Tor relays. The probe measures the time to route a message through each Tor relay in the network. For consistency, the measurement is performed by creating a one-hop circuit through each Tor relay and measuring the network latency. The probe entity consists of three components, all hosted on the same physical machine: a probe client, probe server, and a public Tor exit node. Although the Tor exit node is a public exit relay, its exit policies are restrictive to only allow exit traffic to the probe server. Moreover, this exit node advertises low bandwidth and is regularly turned off to prevent it from gaining the "fast" and "stable" flags in the consensus document, which in turn, decreases the probability that it will be used by other circuits. Since actual one-hop circuits are not allowed in Tor, this exit node is needed to simulate a one-hop circuit. Since the exit node is hosted on the same physical machine as the probe client and server, and is not used by other circuits, the overhead introduced is small and consistent among all the network probe measurements. The burst server machine and the probe machine are time synchronized so that the increase in

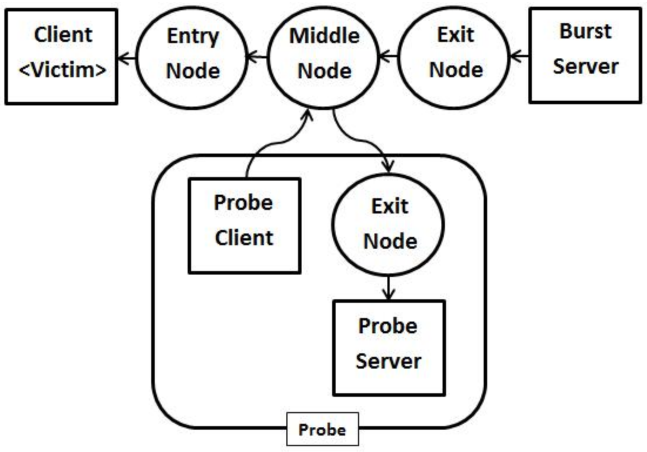


Fig. 2. The layout of our proposed attack experiment. The burst server could be a physical server or part of the content served by another server, for example an advertisement server. The circuit created by the client consists of real Tor relays. The probe's three components are hosted on the same physical machine. The probe exit node is set up such that it only accepts exit connections to the probe server.

network latency measurements can be correlated with the time of the burst traffic period. All the entities of our experiments are shown in Figure 2.

The burst server introduces burst of traffic to the circuit, in an attempt to disrupt the timing information measured by the probe. The burst server alternates between a period of burst traffic and a period of low traffic. Ideally, the Tor network traffic is constant enough that introducing noise or extra messages temporarily will increase the network latency of the three Tor relays measured by the probe. If this experiment is performed multiple times, the identity of the three Tor relays used in the circuit can be identified, and the identity of the client can be leaked [12], [13]. The burst server starts by sending very few packets for a period T_I, to obtain probe measurements during the initial period. The burst server then sends a burst of packets for a period of T_A, then sends very few packets for a period of time T_S, and so on. During the burst period, it is expected that the probe will measure an increase in network latency for the three Tor relays used in the circuit, but no increase in network latency in all other relays. We next describe in more detail how the probe works.

The probe client creates a circuit with each of the relays of the Tor network as the entry node, and chooses the probe exit relay as the exit node. There is no middle node. Since the exit node is controlled by the probe, essentially, a "one-hop" circuit is created. Figure 3 illustrates how the probe works. Once a circuit is created, at regular intervals, the probe client sends a timestamp t_1 to the probe server through the circuit. The server records the time t_2 that it received the timestamp. Since the probe client and probe server are located on the same physical machine, no time synchronization is required. Also, since the probe exit node is not used for other circuits, the latency through the exit node and the extra time for processing is minimal. That small extra time is also consistent for all the probe measurements and does not affect any of the timing information. The network latency of each circuit (each relay node p) is $t_p = t_2 - t_1$.

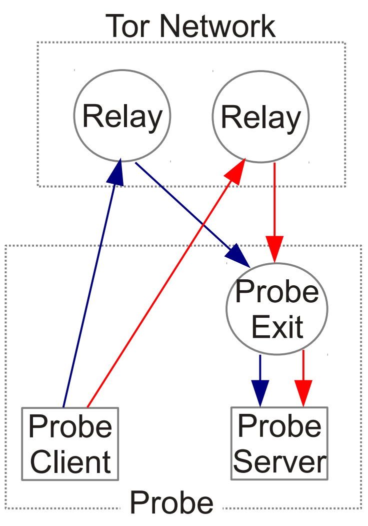


Fig. 3. The probe works by creating "one-hop" circuit to each Tor relay, with the exit node being the probe exit node. In this example, the Tor network is shown to have only two relays.

The adversary needs to control the probe's Tor exit node. It is possible for the probe's Tor exit node not to appear in the public consensus document; an adversary could be more stealthy that way. This requires setting up private directory servers, which mirror the real Tor directory servers. This does not affect the results of the experiments, just whether the probe's exit relay is public or not. Our modification to the original circuit clogging attack [17] also requires that the Tor exit node used in the circuit, shown in Figure 2, be controlled by the adversary. We plan to relax that requirement in future work. Although the Tor exit node advertises a low bandwidth to the directory servers (10 KB/s), it is allowed to relay as much traffic as the network bandwidth allows; this is to prevent our two Tor exit nodes (probe and client circuit) to be used in other circuits.

IV. Proposed Detection Scheme

We propose a scheme that can be used by all clients using Tor to detect when a circuit clogging attack is happening. Recall that a circuit clogging attack, like the one described in the previous section, can reduce the anonymity of all Tor users. When a possible circuit clogging attack is detected, the client can disconnect from the server and destroy the affected circuit. This detection scheme can also be used by the Tor operators, such as the directory servers, to monitor the Tor network for circuit clogging attacks.

The idea behind the proposed scheme is to use a probe to measure the network latency of each circuit. This is similar to the adversary using a probe to build one-hop circuits to each Tor relay. When the client creates a circuit, it also starts a client probe through the same circuit. The client probe regularly sends a timestamp ts_1 through the circuit to the victim probe server. The probe server replies with the same message ts_1. Once the client probe receives the reply from the probe server, it calculates the current time ts_2, and the RTT or network latency for that circuit is $ts = ts_2 - ts_1$. Both the victim

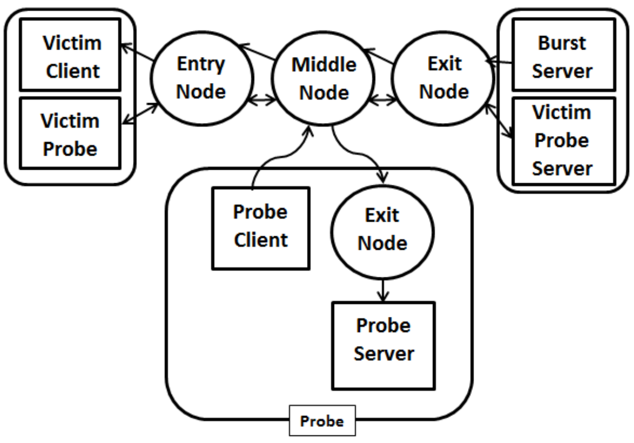


Fig. 4. The layout of the proposed detection scheme for circuit clogging attacks. The two new entities are the victim probe and the victim probe server. The probe works similar to the attack probe, except the whole circuit is monitored instead of individual relays.

client and the victim probe are hosted on the same physical machine. Both the burst server and the victim probe server are also hosted on the same server. Figure 4 depicts the setup for the proposed detection scheme.

In our experiments, we set up both a victim probe and a victim probe server. In a real attack, the victim probe server is not needed. The network latency of the whole circuit can be obtained through the lower layers. For example, the TCP sequence numbers and corresponding timestamps can be examined to determine the network latency for each circuit. Also, a public trusted server, similar to the Tor directory servers, can be hosted to reply to victim probes. As the next section shows, the costs required to run such a server are not very big. The challenges in setting up a public probe server is outside the scope of this paper. All the clients need to run is an extra process or thread for the victim probe, which sends a probe at regular intervals through all the created circuits.

It can be argued that clients can also set-up one-hop circuit for each of the relays used. However, since Tor, by default, blocks one-hop exits, the client will have to also host a Tor exit node, which might not be feasible nor practical.

V. Evaluation

A. Experimental Setup

All our experiments were performed during February 2013 using Tor version 0.2.3.25. The client, burst server, and probe were hosted on different machines, and were time synchronized. In a real attack, only the burst server and the probe are controlled by an adversary. Before each experiment, the latest consensus document was downloaded from the Tor directory servers. Only the Tor relays with the "fast" and "stable" flags were chosen. The Tor node selection uses these criteria as well, so our experiment is close to what would happen in a real attack. Two Tor relays were randomly chosen to be the entry and middle relay in the circuit used by the client to connect to the burst server. The exit relay is hosted on our server; we plan to relax this constraint later. Two other Tor relays were also chosen, as a control case. In our experiments, we are only

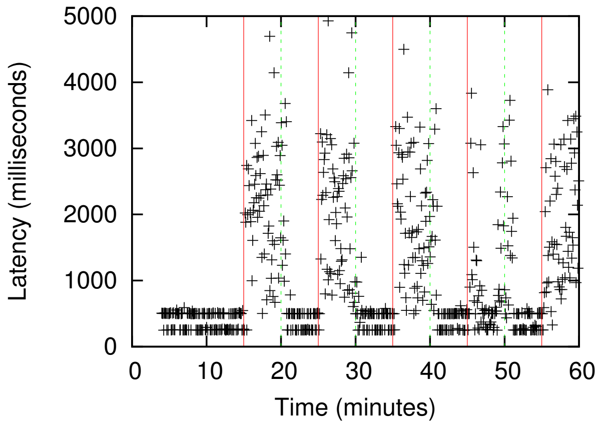


Fig. 5. Probe network measurement for a Tor relay used in the client circuit. The burst attack period starts at the red (solid) vertical line and stops at the green (dashed) vertical line. The pattern of burst/sleep periods can be clearly seen.

probing four Tor relays, not the whole Tor network – an actual attack would have to probe the whole Tor network. The entry and middle relays are referred as Tor_C, while the two other Tor relays are referred as Tor_R. The probe creates a "one"-hop circuit to each of the four relays and sends a network measurement every 5 seconds. The initial time period T_I was set to 15 minutes, while the burst attack period T_A was set to 5 minutes, and the sleep period T_S was set to 5 minutes. Each experiment was run for 60 minutes. Instead of creating one single connection, the client is multi-threaded and sets up 5 threads to connect to the server. All the 5 threads use the same circuit. In a real setting, a server can redirect a client to 5 different content servers; many web browsers download multiple parts of a server in parallel. For each experiment, the four Tor relays are randomly selected from the list of Tor nodes.

Figure 5 shows a probe's network measurement of a Tor relay used in the circuit. Each probe was sent at intervals of 5 seconds. It can be seen in the figure that during the burst period (the beginning of which is indicated by the red darker solid vertical line), the latency measured increases, and during the sleep period (the beginning of which is indicated by the green lighter dashed vertical line), the latency decreases to that of the initial period. The initial period is used to set the baseline for the average network latency for each relay.

The victim probe also is set to measure the latency of the circuit every 5 seconds.

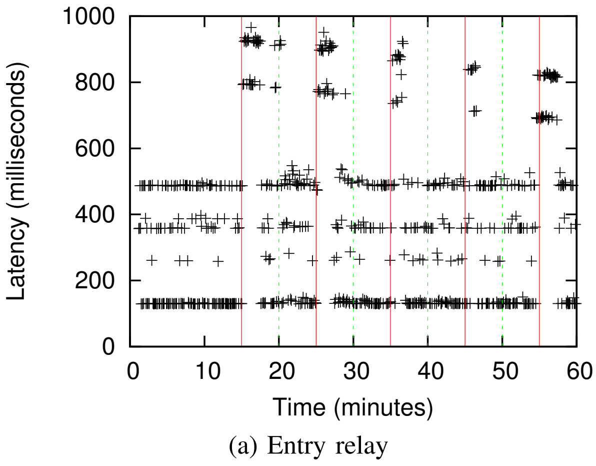
B. Results of Attack

Figure 6 shows an example of an experiment run. The figure shows the probe's network latency measurement for the four Tor relays used in that experiment: two of them were part of the circuit, and the other two were not part of the circuit. Figure 6(a) and (b) refer to the entry and middle node in the circuit, respectively, while Figure 6(c) and (d) refer to the other two Tor relays. These two Tor relays ((c) and (d)) act as a control case, representing all the relays in the Tor network. The probings of these two relays should show no difference during the burst attack period and during the sleep period. However, the entry and middle node should show a distinct pattern, which is seen in Figure 6.

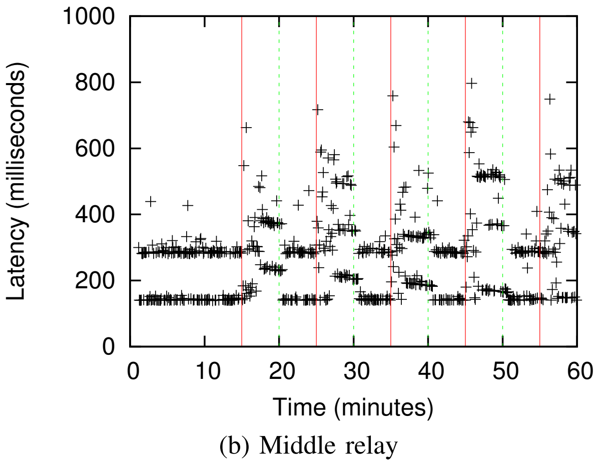
The probe machine is time-synchronized with the burst server and knows when the different periods (initial/burst/sleep) happen. The initial time period T_I is used as a baseline. The baseline time t_b is calculated as the average of all the network latency times during the initial period T_I. The average time for each period is also calculated. For the entry and middle relays' measurements, it is expected that the average time during the burst periods will be higher than t_b, and the average time during the sleep periods will be about the same as t_b. For the two other Tor relays Tor_R, it is expected that the average time for all the periods will be similar to the average initial time t_b. Due to noise and variations in probe measurements, such as a Tor relay being used in a bandwidth-intensive circuit, the average time for the sleep periods for Tor_C and for all periods for Tor_R might be higher than the average initial time t_b. If this happens, this is called a *false positive*. These regular variations in the network latencies measured lead to using a threshold value α to determine whether the average time t_a for a period indicates a burst period or a sleep period. If $t_a \geq \alpha \times t_b$, then this indicates that the network latency measured is high enough that it indicates a burst period. Relays experiencing such high network latencies during actual burst periods are marked as possible entry or middle relays used in the circuit. The value of the threshold α is varied from $1.0 - 5.0$.

The varying threshold produces different numbers of false positives (a random Tor relay accidentally marked as being part of the circuit used to connect to the burst server) and different numbers of correct predictions or numbers of true positives (correct Tor relay identified as being either entry or middle relay in the circuit used to connect to the burst server). The Receiver Operator Characteristic (ROC) [26] curve shows the trade-off between the false positive rate and the true positive rate. Figure 7 illustrates the ROC curve for our experiments, when varying the threshold α. The line $y = x$ (green line in the figure) shows the true positive rate and the false positive rate when randomly determining whether a Tor relay is part of the circuit or not (50% probability of being correct). The vertical line from $(0, 0)$ to $(1, 1)$ shows a perfect classifier where all the Tor relays part of the circuit are correctly identified and there are no false positives. Figure 7 shows that our experiments fall between the perfect classifier and random classifier. This means that our attack is better than random but not perfect. The area under the curve (AUC) shows the trade-off between the true and false positive rate. The perfect classifier has an area under the curve of 1.0 and the completely random classifier has an AUC of 0.5. The area under the curve for our attack is 0.78. The area under the curve when using one thread for the attack, instead of five threads, is 0.68 (ROC curve not shown). This means that using more threads is more effective at performing the circuit clogging attacks than using just one thread. The equal error rate indicates when the false positive rate is equal to the true positive rate. The lower the equal error rate, the more accurate is the system. Our attack achieved an equal error rate of 28%, which means that our attack is accurate.

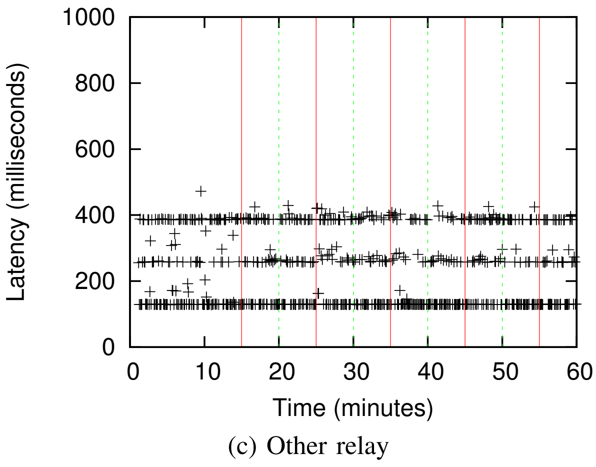
The results indicate that the circuit clogging attack is still possible in the current Tor network. The probings of Tor relays



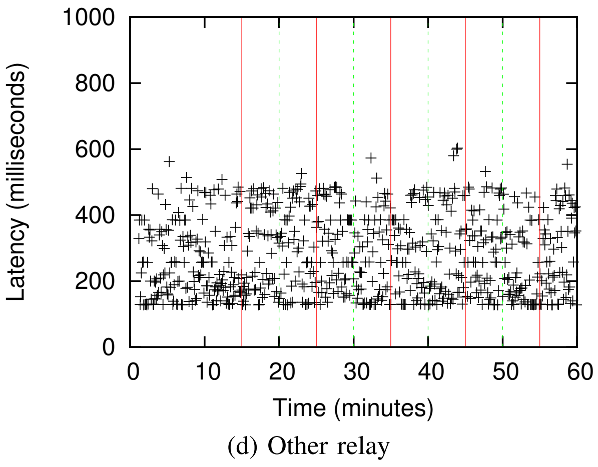
(a) Entry relay



(b) Middle relay



(c) Other relay



(d) Other relay

Fig. 6. The probe's network latency measurement over time for (a) the entry relay, (b) the middle relay, (c) and (d) the other two Tor relays not part of the circuit.

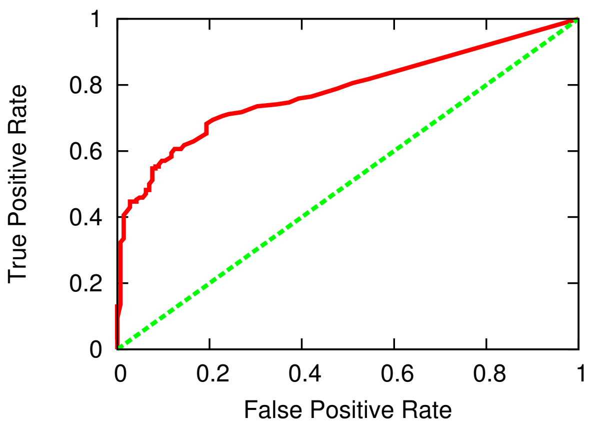


Fig. 7. The Receiver Operator Characteristic (ROC) curve for the probe measurements. The area under the curve (AUC) is 0.78 and the equal error rate is 28%. The $y = x$ line indicates the random classifier.

for measuring the network latency is effective at determining when a burst attack period occurs and at identifying whether a relay is used in the circuit by the client to connect to the malicious burst server. We note that our attack is conservative: more server threads, a higher burst of messages, and a longer running time with more burst/sleep periods will make the attack more accurate. However, since our experiments were performed on the real Tor network, we did not want to affect the load on the Tor relays unnecessarily.

C. Detection Scheme

The proposed detection scheme was run at the same time as the attack. Each probe measurement is per circuit created by the client. In a real setting, the client does not know when the burst periods are, which is different for the attacker's probe. For experimental simplicity, the victim machine and the burst server are also time-synchronized. This allows us to determine the baseline and calculate the number of correct predictions of circuit clogging attack happening. Figure 8 shows a time versus network latency measured graph for the victim probe in a circuit that is used to communicate with the busrt server. It can be clearly seen in the figure that during the burst period, the network latency measured increases and gradually goes back to normal when the burst period ends. The increase in network latency is also more pronounced than for the attack probes, as seen in Figure 6; the y-axis scales are different. For the one-hop attacker probe circuits, only one relay is affected, whereas

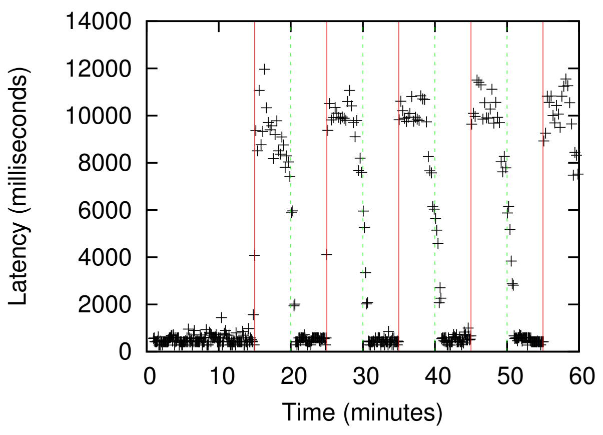


Fig. 8. One experiment showing the client's probe of a circuit used for connecting to the burst server. The periods of burst/sleep can be clearly seen in the figure.

for the client's three-hop circuits, three relays are affected.

Similar to the attack probe measurement, an initial time period T_I is needed to set the baseline average time t_c. The baseline time can be obtained by the client probe connecting to the client probe server some time before connecting to the burst server, or through the average of all other circuit times, or through a network-wide latency average in the consensus document. When the measured circuit network latency exceeds the baseline initial time t_c, this circuit is marked as suspicious. If the burst period was active at that time, then this counts as a correct prediction. Our proposed detection scheme is able to accurately detect 87.5% of all burst periods; that is, it can predict when a circuit clogging attack is happening 87.5% of the time.

The proposed detection scheme is thus accurate. Although the scheme is not perfect, once a possible attack is detected, all the connections streamed over that circuit can be reported to the user, with a list of all the servers. After the detection scheme is run for some time, the repeated servers on the reported list are suspicious and can be further investigated. This whole detection and reporting mechanism can be performed transparently to the user. The challenges in authenticating this list and preventing "bad-mouthing" of honest servers are outside the scope of this paper. The proposed detection scheme allows the client to destroy a suspicious circuit, to preserve the user's anonymity. The proposed detection mechanism has some limitations, such as false positives and the overhead/cost in creating a new circuit. The user has control when to destroy a suspicious circuit. The user can disconnect as soon as the circuit is flagged as suspicious, or it can wait until k "bursts" are detected.

D. Costs

The total costs and overhead for performing the burst attack (hosting the malicious content) and the probe machine are small, as will be shown in this section.

The bandwidth used in our burst server during the burst attack period is less than 30 KB/second, on average. If the malicious content is hosted on Amazon EC2 [27], the cost to run one server for one circuit is $\$11.38$ per month. The storage and processing needed to maintain the burst server are minimal, and the costs for storage and processing are assumed to be minimal.

The bandwidth required to probe one Tor relay is 130.4 bytes/second. Probings are performed every 5 seconds. If a more fine-grained measurement is required, the interval between measurement can be decreased, which will increase the costs to run the probe machine. With $3,000$ relays, this is $3000 * 130.40 = 391,200$ bytes/second, which is a bandwidth of about 391 KB/second. The cost to run the probe on Amazon EC2 is $\$121.91$ per month. The processing needed for each network latency measurement is minimal. The storage required is to record all the network latency measurements, which consist of timestamps. Each timestamp is only 13 bytes. For a 60 minute experiment, this is 9.36KB for probing one relay, or 28MB storage for all the relays.

The bandwidth required for a client to run the proposed detection mechanism is 130.4 bytes/second, for each circuit the client creates. With possibly 20 circuits created, this is an extra bandwidth of $2,608$ bytes/second or 2.6 KB/second which is minimal. The interval for each probe is 5 seconds; this interval can be decreased for an increase in extra bandwidth needed but faster detection. If the victim probe server is a public server, the bandwidth required to reply to 5-minutes probe from $500,000$ clients using 20 circuits is 21.7MB/second or 1.9TB/day, which is $\$5,870$ on Amazon EC2. In general, the overall costs of running a victim probe, including processing, storage, and network, are minimal. The benefit of running a probe, however, is real, and allows a client to detect the occurrence of a circuit clogging attack. This helps to preserve the client's anonymity.

VI. Discussion and Conclusion

We showed that a circuit clogging attack is still possible in the current Tor network, contrary to previous claims. The goal of a circuit clogging attack is to identify all the Tor relays used in a circuit. A client uses the circuit to connect to a malicious server (or malicious content hosted on a honest server). The malicious server periodically switches between burst mode and sleep mode. During burst modes, the server sends a burst of messages, while during the sleep modes, the server sends a few messages only. Since all the relays in the Tor network are monitored through probe network latency measurements, a spike in network latency is observed for the three Tor relays used in the circuit during the burst periods. The costs to perform a circuit clogging attack are also very low, making it a practical attack. We showed that the Tor relays used in a circuit can be accurately identified. Moreover, the false positive rate is low as only some other Tor relays not used in the circuit are accidentally identified as being part of the circuit.

A circuit clogging attack detection mechanism is also proposed. The scheme uses a probe to monitor all the circuits created by the client, instead of each Tor relay. Once an increase in network latency from a previously recorded baseline time is measured, the server is flagged as suspicious. The client can then disconnect from the server and destroy the affected circuit. Through experiments on the real Tor network, the proposed detection scheme has an accuracy of over 85%.

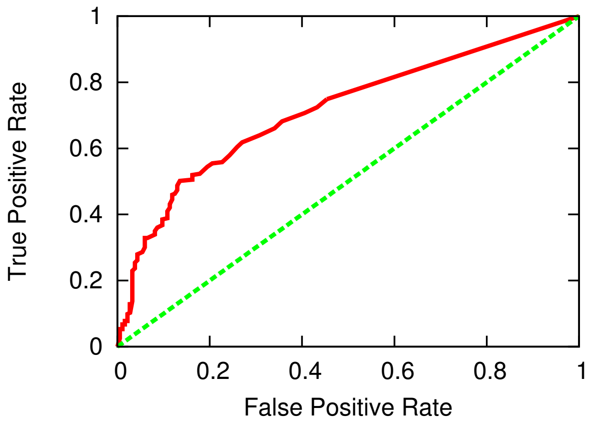


Fig. 9. The Receiver Operator Characteristic (ROC) curve for the probe measurements, where all the relays in the client circuit are public Tor relays. The area under the curve (AUC) is 0.72. The $y = x$ line indicates the random classifier.

This paper showed that the anonymity of a person using Tor is reduced since the Tor relays used can be identified. This can be a stepping stone towards narrowing down the possible users behind these relays. The detection scheme proposed allows a user to detect possible occurrences of circuit clogging attacks. With over $500,000$ users daily, the attack has huge potential consequences. The proposed detection scheme can help hundreds of thousands of people stay anonymous on the Internet.

Although a detection mechanism is better than nothing, a prevention algorithm would be best. We leave as future work designing a scheduling algorithm that can prevent circuit clogging attacks, such as [25]. The current experimental set-up requires that the Tor exit node in the circuit be under the control of the attacker. We performed 15 experiments where all the nodes in the circuit are public Tor relays and with one server thread. Figure 9 shows the ROC curve. The area under the curve (AUC) was 0.72, which is close to an AUC of 0.78 when the exit relay belonged to the adversary. Based on this promising preliminary result, we plan on extending our attack such that the exit node used in the circuit is a regular Tor exit node. We will explore ways to improve the accuracy of the attack, such as using multiple server threads, varying the time of the burst/sleep periods, and modifying the amount of data sent during the burst periods.

The current attack identifies the Tor relays in the circuit; future work will identify which of the relays are the entry, middle, and exit relays. This would require more fine-grained probe measurements. One of the anonymity improvements in Tor is to use entry guards, a fixed set of three relays used as entry relay in any circuit. We will also analyze whether using entry guards leaks any information, as the user could be more easily identified. If the same entry relay is found in circuits, this can leak information about the user. The possible impact of the attack on bridges and hidden servers is left as future work.

References

[1] J. Cowie, "Egypt leaves the internet," http://www.renesys.com/blog/2011/01/egypt-leaves-the-internet.shtml.

[2] J. Vargas, "How an egyptian revolution began on facebook," http://www.nytimes.com/2012/02/19/books/review/how-an-egyptian-revolution-began-on-facebook.html.

[3] J. Brodkin, "Iran reportedly blocking encrypted internet traffic," http://arstechnica.com/tech-policy/2012/02/iran-reportedly-blocking-encrypted-internet-traffic/.

[4] "I2P anonymous network," http://www.i2p2.de/.

[5] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003, pp. 2–15.

[6] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster Protocol — Version 2," IETF Internet Draft, July 2003.

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

[8] ——, "Practical anonymity for the masses with morphmix," in *Proceedings of Financial Cryptography (FC '04)*, A. Juels, Ed. Springer-Verlag, LNCS 3110, February 2004, pp. 233–250.

[9] "Tor," https://www.torproject.org/.

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

[11] "Tor metrics portal," https://metrics.torproject.org/.

[12] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How much anonymity does network latency leak?" *ACM Transactions on Information and System Security*, vol. 13, no. 2, February 2010.

[13] ——, "How much anonymity does network latency leak?" in *Proceedings of the 14th ACM conference on Computer and communications security (CCS)*, 2007, pp. 82–91.

[14] A. Back, U. Möller, and A. Stiglic, "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, April 2001, pp. 245–257.

[15] G. Danezis and A. Serjantov, "Statistical disclosure or intersection attacks on anonymity systems," in *Proceedings of 6th Information Hiding Workshop (IH 2004)*, ser. LNCS, Toronto, May 2004.

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

[17] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[18] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against Tor," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)*, Washington, DC, USA, October 2007.

[19] N. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on tor using long paths," in *Proceedings of the 18th USENIX Security Symposium*, August 2009.

[20] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *IEEE Symposium on Security and Privacy*, may 2007, pp. 116 –130.

[21] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting," in *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS 2011)*, October 2011.

[22] Y. Gilad and A. Herzberg, "Spying in the Dark: TCP and Tor Traffic Analysis," in *Proceedings of the 12th Privacy Enhancing Technologies Symposium (PETS 2012)*. Springer, July 2012.

[23] "BitTorrent," http://www.bittorrent.com/.

[24] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding Routing Information," in *Proceedings of Information Hiding: First International*

Workshop, R. Anderson, Ed. Springer-Verlag, LNCS 1174, May 1996, pp. 137–150.

[25] J. McLachlan and N. Hopper, "Don't clog the queue: Circuit clogging and mitigation in P2P anonymity schemes," in *Proceedings of Financial Cryptography (FC '08)*, January 2008.

[26] T. Fawcett, "An introduction to roc analysis," *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[27] "Amazon EC2," http://aws.amazon.com/ec2/.