Detecting Denial of Service Attacks in Tor

Norman Danner, Danny Krizanc, and Marc Liberatore

Department of Mathematics and Computer Science
Wesleyan University
Middletown, CT 06459 USA

Abstract. Tor is currently one of the more popular systems for anonymizing near real-time communications on the Internet. Recently, Borisov et al. proposed a denial of service based attack on Tor (and related systems) that significantly increases the probability of compromising the anonymity provided. In this paper, we propose an algorithm for detecting such attacks and examine the effectiveness of the obvious approach to evading such detection. We implement a simplified version of the detection algorithm and study whether the attack may be in progress on the current Tor network. Our preliminary measurements indicate that the attack was probably not implemented during the period we observed the network.

Keywords: Anonymity, reliability, denial of service, attack, detection

1 Introduction

A low-latency anonymous communication system attempts to allow near-real-time communication between hosts while hiding the identity of these hosts from various types of observers. Such a system is useful whenever communication privacy is desirable — personal, medical, legal, governmental, or financial applications all may require some degree of privacy. Financial applications that might benefit from such privacy include e-cash or credit systems, contract proposal and acceptance, or retrieval of financial data.

Dingledine et al. developed the Tor [1] system for such communication. Tor (and other related systems) anonymizes communication by sending it along paths of anonymizing proxies. Syverson et al. [2] showed that such systems are vulnerable to a passive adversary who controls the first and last proxies along such a path. More recently, Borisov et al. [3] showed that an adversary willing to engage in denial of service (DoS) could increase their probability of compromising anonymity. When a path is reconstructed after a denial of service, new proxies are chosen, and thus the adversary has another chance to be on the endpoints of the path.

Our contributions are as follows. We prove that an adversary engaging in the DoS attack in an idealized Tor-like system can be detected by probing at most $3n$ paths in the system, where n is the number of proxies in the system. Through simulation, we show that an adversary attempting to avoid detection by engaging in DoS probabilistically can still be detected, and that the attempt to avoid detection radically degrades the effectiveness of the attack. Finally, using measurements of connection drop rates across Tor nodes, we implement a version of the detection algorithm and conclude it is unlikely that such an attack was in progress during the time period the network was observed.

We introduce related work and present the attack in more detail in Section 2. Section 3 describes the algorithm to detect attacker-controlled nodes, and Section 4 describes one possible attacker strategy to avoid detection, along with an evaluation of its effectiveness. This is followed by our measurements of Tor node drop rates (Section 5) and the results of a practical implementation of our detection algorithm (Section 6). We conclude in Section 7.

2 The Denial of Service Attack

We model the Tor network with a fully connected undirected graph.[1] The set N of vertices of the graph represent the Tor nodes (or routers), and the edges represent network connections between nodes. We define n to be $|N|$.

Tor sets up circuits (also referred to as tunnels) consisting of three nodes; in our model, this equates to a path containing three vertices (in order) and the corresponding edges between them. To simplify the analysis, we allow the same node to appear on the path more than once – that is, nodes are chosen uniformly at random with replacement. Application level communications between an initiator and a responder is then passed through the circuit. We assume that a timing cross-correlation attack works perfectly, i.e., the adversary can break the system's anonymity properties if it controls the first and last node along the path by observing the timing of communications between an initiator and responder.

In any attack we assume some subset C of N are compromised, that is, they are collaborators under the control of an adversary that will attempt to break the anonymity of users in the system. As with N, we define c to be $|C|$. Here we limit our attention to the nodes within the Tor network, under the assumption that an adversary will compromise some of these nodes in an attempt to link initiators and responders.

In actual deployments of Tor, not all nodes can appear in all locations on the path. In particular, only certain nodes can be the final node on the path; in all other ways, they are identical to other nodes. These nodes are referred to as exit nodes. Let $E \subseteq N$ be the set of exit nodes where $e = |E|$.

Syverson et al. [2] observed that a passive adversary controls both the first and last node of a path with probability $\frac{c^2}{n^2}$ if all nodes may act as exit nodes. In the case where exit nodes are selectively compromised this may be improved to $\frac{c^2}{ne}$. (Currently approximately one third of Tor nodes act as exit nodes at any one time.)

Levine et al. [4] observe that if long-lived connections between an initiator and responder are reset at a reasonable rate then such an attack will be

[1] Some individual Tor nodes may disable connections on specific ports or to specific IP addresses. We have not determined if these significantly limit the graph.

able to compromise anonymity with high probability within $O(\frac{n^2}{c^2} \ln n)$ resets.

In order to further improve the chances of compromise of communications over a Tor circuit a number of researchers [3, 5, 6] have suggested that compromised nodes that occur on paths in which they are not the first or last node artificially create a reset event by dropping the connection. Borisov et al. [3] analyze the following version of this attack on Tor:

> If the adversary acts as a first or last router on a tunnel, the tunnel is observed for a brief period of time and matched against all other tunnels where a colluding router is the last or first router, respectively. If there is a match, the tunnel is compromised; otherwise, the adversary kills the tunnel by no longer forwarding traffic on it. The adversary also kills all tunnels where it is the middle node, unless both the previous and next nodes are also colluding.

In this case, the adversary controls the endpoints of a randomly generated path with probability

$$\frac{c^3 + c^2(n-c)}{c^3 + c^2(n-c) + (n-c)^3} = \frac{\alpha^3 + \alpha^2(1-\alpha)}{\alpha^3 + \alpha^2(1-\alpha) + (1-\alpha)^3}$$

where $\alpha = c/n$ is the fraction of compromised nodes,

Assuming that not all nodes are exit nodes and $C \subseteq E$ the endpoints of the path are compromised with probability:

$$\frac{c^3 + c^2(n-c)}{c^3 + c^2(n-c) + (n-c)^2(e-c)} = \frac{\alpha^3 + \alpha^2(1-\alpha)}{\alpha^3 + \alpha^2(1-\alpha) + (1-\alpha)^2(\frac{e}{n} - \alpha)}$$

The conclusion by Borisov et al. is that their optimization brings significant gains to the attacker. As shown in Figure 1, it is strongly in the attacker's interest to kill circuits that can't be compromised. The gain from this attack is even more pronounced when exit nodes are selected for compromise before other nodes.

3 Detecting the Attack

In this section we show how to detect such a DoS attack using $O(n)$ probes of the network where a probe consists of setting up a circuit using a given path through the network and passing data through it. We assume a naive attacker that follows the procedure precisely as formulated above. Further we assume that the time taken for an attacker to detect a match is negligible when compared to the expected time between circuit kills due to unreliable but uncompromised nodes. I.e., we assume that if a probe results in a circuit being killed inside of a short period of time after being created, this is due to the fact that there is at least one compromised node on the circuit and that it is not the case that both endpoints of the corresponding path are compromised. Note that by sending traffic with a predictable pattern through the circuit we can make the time taken for detection very low. As we discuss below, by repeating the experiment we can make the probability of confusing

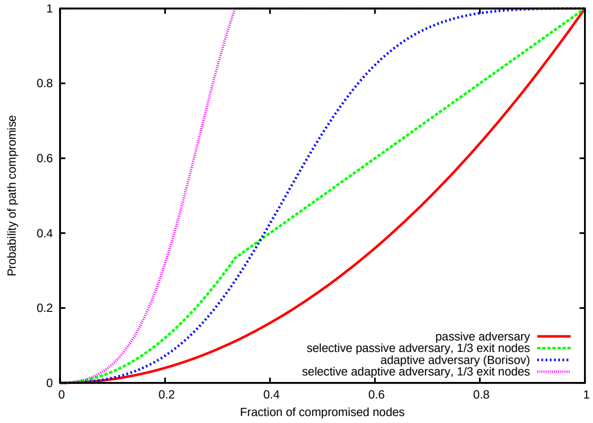

Fig. 1. A comparison of the probability of an circuit being compromised, given either the passive or adaptive (Borisov) adversary. The selective adversary, in either case, compromises only exit nodes.

a kill due to unreliable but uncompromised nodes and a kill due to a compromised node as small as we wish.

First observe that it is impossible by using only the above probes of the network to distinguish between the case where all nodes are compromised and no nodes are compromised. In both cases, all probes will result in circuits that are not killed. Therefore we assume the number of compromised nodes is greater than 0 but less than n. Further we assume that the length of the paths used by the Tor implementation under attack is fixed independent of (and strictly less than) n and that paths consist of distinct nodes. We can prove:

Theorem 1. *Under the above assumptions, using $O(n)$ probes we can detect all of the compromised nodes of the Tor network. For the case of paths of length 3 the number of probes required is at most $3n$.*

Proof. Let k be the length of the paths used by the Tor implementation under consideration. We denote the probe consisting of the path of length k starting with u_1 and ending with u_k with edges between u_i and u_{i+1} for $i = 1, \ldots, u_{k-1}$ by $(u_1, \ldots, u_k)$. We say a probe *succeeds* if the circuit is not killed, otherwise it *fails*.

Choose a set $X = \{x_1, \ldots, x_{k-1}\}$ of $k-1$ (distinct) nodes, arbitrarily. Perform the following set of probes: $(x_1, y, x_2, \ldots, x_{k-1})$ for each y not in X. One of three cases results.

Case 1: All $n - k + 1$ probes succeed. In this case both x_1 and x_{k-1} are compromised. For any other node y, we can determine if it is compromised by using the probe $(x_1, \ldots, x_{k-1}, y)$. If it succeeds then y is compromised, if not, y is uncompromised. (To test nodes in X, replace them in the above probe set with an arbitrary node not in X and try a probe with the given node in the last position.)

Case 2: Among the $n-k+1$ probes, at least one succeeds and at least one fails. In this case all of the nodes in X are uncompromised, any y for which the probe failed is compromised, and any y for which the probe succeeded is uncompromised.

Case 3: All $n-k+1$ probes fail. In this case we can conclude that at least one of the nodes in X is compromised. For each pair of nodes in X create a probe set where this pair comprises the first and last node of the path, positions 3 through $k-1$ consist of the remaining nodes of X (in an arbitrary fixed order), and the second position is taken by each y not in X. If for any of these probe sets the result is all the probes succeed then both endpoints are compromised and we proceed as in Case 1 to determine the status of the remaining nodes. If for all pairs the result is all probes fail, then there is precisely one compromised node in X. For each node in X create a probe set where that node is in the first position, the remaining nodes are positions 2 through $k-1$ and the last node is taken by each y not in X. If for any of these probe sets the result is at least one circuit that succeeds and at least one circuit that fails then the node in the first position is compromised, the remaining nodes in X are uncompromised, any y for which the probe was a success is compromised and any y for which the probe was a failure is uncompromised. If all of the above probe sets result in all probes failing then there is exactly one compromised node and it is in X. For each node in X create a probe that does not use that node but uses the remaining nodes in X (plus two other arbitrary nodes not in X). The unique such probe that succeeds points to the single compromised node.

The worst case number of probes occurs in Case 3 in which at most $(\binom{k}{2}+k+1)(n-k+1)+k$ probes beyond the initial $n-k+1$ probes that define the cases. As k is assumed to be fixed independent of n this is clearly $O(n)$. For the case $k=3$, Case 3 requires that we determine which the endpoints is compromised which can be done using at most two probe sets of size $n-k+1$ one of which will determine the identity of the remaining compromised nodes. In this case at most $3(n-k+1)$ probes are required. □

Above we state that (under the perhaps unrealistic assumption that the results of probes are independent) repeated probes can be used to distinguish the cases of an attacker killing a circuit and that of a circuit consisting of honest nodes failing. To do this, for any given probe of the above algorithm we repeat the probe l times where l (determined below) depends upon on the probability of error in the algorithm we find acceptable. If all l of the trials fail we report that the path contains at least one compromised node and that at least one end point that is honest. Otherwise we conclude the path contains all honest nodes or both end points are compromised.

Assume that an attacker always successfully kills a circuit it is on that it does not control. Then a probe consisting of l independent trials can be wrong only if (a) an honest circuit fails l times in a row or (b) a circuit with both end points compromised fails l times in a row. Assume that any given circuit fails due to unreliable nodes or edges with probability f. Then, under the independence assumption, (a) or (b) occur with

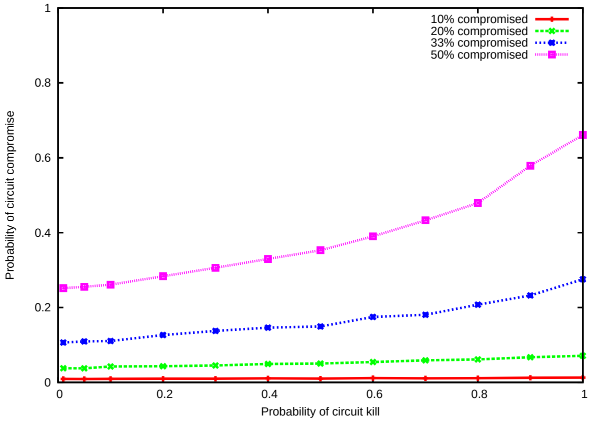

Fig. 2. Probability of an circuit formation being compromised, given the adaptive adversary (performing a DoS), varying the fraction of killed paths.

probability at most f^l, i.e., the probability that a probe consisting of l independent trials is correct is at least $1-f^l$. If the algorithm performs m such probes the probability they are all correct is greater than $(1-f^l)^m$. Assume we require that our algorithm correctly identifies all nodes as either honest or compromised with probability at least $1-\epsilon$. Then it is easy to see (using standard approximations) that choosing

$$l > \frac{\ln\ln(\frac{1}{1-\epsilon}) - \ln m}{\ln f}$$

is sufficient. If we take $m = 3000$ (the worst case number of probes for a 1000 node Tor network), $f = .2$ (an approximate bound the observed probability of path failures on Tor — see Section 5) and $\epsilon = .001$ (so that we expect less than one misidentification) we see that $l = 10$ is more than sufficient.

Of course, we require that the above repeated probes be independent which is highly unlikely to be the case. But by spreading the repetitions out over time we can increase our confidence that observed failures are not random.

4 Attacker Strategy

An intelligent attacker will be aware that killing circuits at a rate higher than the background rate can, in theory, be detected. Here, we consider the case of an attacker that kills some fraction of the circuits through nodes under its control. In particular, circuits that contain compromised nodes, but where both endpoints are not compromised, are killed. The attacker can choose to kill any fraction of such circuits, from all (which is Borisov's description of the DoS attack) to none (equivalent to the passive adversary).

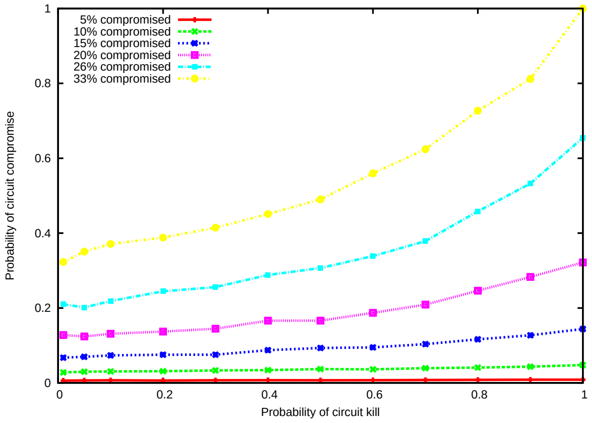

Fig. 3. Probability of an circuit formation being compromised, given the selective adaptive adversary (performing a DoS, all compromised nodes are exit nodes), varying the fraction of paths killed. While we assume that the background kill rate is zero and we fix $e = \frac{n}{3}$, the general shape of the graph remains when changing these assumptions.

Figure 2 shows the effect of varying the kill probability for various fractions of compromised nodes, obtained through simulations conforming to the assumptions in Section 2. Clearly, the adaptive attacker is most effective when in control of many nodes, and when killing as many circuits as possible.

In the deployed Tor system, only certain nodes are eligible to be last in the circuit. The intelligent selective adversary compromises these exit nodes. Figure 3 shows the effect of varying the kill probability for various fractions of compromised exit nodes.

These results strongly suggest that the selective adversary is in a difficult position. In failing to kill circuits, the adversary does no better than a passive adversary. An adaptive adversary killing a fraction of circuits results in small gains over the passive adversary for fractions that are not close to one. As discussed in Section 3, an attacker that always kills paths can be detected in a small number of probes; the same reasoning leads to the conclusion that an adversary that kills any significant fraction of paths can still be detected in a small number of probes.

5 Measuring Failure Rates in Tor

As already mentioned, implementation of our detection algorithm is dependent upon knowing the background drop-rate for circuits in Tor. We can consider a range of sophistication for the attacker, corresponding to how much data the attacker must see before killing the circuit:

1. Minimal data. The strongest attacker knows to kill the circuit based solely on the Tor circuit creation cells. Bauer et al. [6] describe such an attacker.

2. Low data. A weaker attacker must wait until the circuit has been successfully built, but can kill the circuit before any data is sent along a TCP stream. The attacker might be able to send some data through the circuit itself, or might observe the Tor cells sent from the initiator to the exit node instructing the latter to open a TCP connection.
3. High data. The weakest attacker observes the actual TCP data and bases its decision to kill the circuit on that.

Accordingly, we conducted an experiment in which we repeatedly downloaded a file through Tor and measured the rates of the different failure modes. The controlling process launches a `curl` process to download the file. The Tor proxy is responsible for circuit creation; the controlling process ignores circuits with exit policies that do not allow the download, but otherwise attaches one `curl` process to one circuit. We then monitor the various failure rates. Kills by the minimal-data attacker correspond to circuit creation failures. Kills by the low-data attacker correspond to attempting to attach a `curl` process to a successfully-built circuit, but the process then receiving either no reply from the server or timing out.[2] Kills by the high-data attacker correspond to a `curl` process failing to completely download the file.

We present our findings in Figure 4.[3] We can treat these failure rates as upper bounds on the corresponding background rates in Tor. An immediate conclusion that we can draw from these measurements is that detecting a minimal-, low-, or high-data attacker using l-times repeated probe version of our detection algorithm requires taking $l = 10$, 9, and 3, respectively. Since it is possible that some version of the DoS attack was in progress when we performed our measurements, the failure rates among honest nodes may in fact be lower; additional measurements along the lines of those described in Section 6 are in order to identity potentially suspicious nodes and determine whether the failure rates decrease significantly when those nodes are removed from the experiment.

6 Detection in Practice

The detection algorithm described in Section 3 along with the measurements made above provide a reasonably practical method for detecting the DoS attack in progress, and serves as a theoretical upper bound on the amount of work necessary to discover such an attack. A number of simplifications are possible if we assume the existence of a single, presumably honest, onion router under our control. In essence, this single honest router is a trustworthy guard node [7]. This trust is important: Borisov et al. note that the use of guard nodes in general may make

[2] We also used the Tor Control protocol to measure the failure rate of the instruction to the exit node to open a TCP connection to the recipient; the results are comparable to those reported here.

[3] These rather high failure rates are consistent with those reported by Mike Perry at BlackHat USA 2007, available at http://www.blackhat.com/html/bh-media-archives/bh-archives-2007.html.

Circuits launched	4995		
Circuit failure at hop 1	106	(2.1%)	
Circuit failure at hop 2	258	(5.2%)	
Circuit failure at hop 3	640	(12.8%)	
Total circuit construction failures	1004	(20.1%)	(minimal-data)
`curl` processes launched	3010		
No reply or timeout	537	(17.8%)	(low-data)
Partial file	6	(0.2%)	(high-data)

Fig. 4. Observed Tor drop-rates corresponding to different-strength attackers.

the selective adversary more powerful when performing the predecessor attack [8]. The assumption of a trusted guard node avoids this problem entirely. We note that this assumption is not strong—by "trusted" here we mean that the node itself is not under the control of an attacker. This can be arranged by installing one's own onion router and using it as the guard node. We further note that this assumption hinges upon the trusted node being indistinguishable from other nodes and that it be unknown to the adversary. If these conditions do not hold, then the adversary can choose to not attack connections from the trusted node and remain hidden. In this sense, the simplified detection algorithm is easier for the attacker to game.

Regardless, what are the advantages of this approach? First, observe that we need only probe nodes that are advertised as exit nodes, as there is a clear reason for an attacker to control exit nodes over non-exit nodes. Second, circuits in Tor are configurable by the initiator. In particular, paths of length two can be created, where the first node is known to be honest and the second, exit node's behavior can be observed. These simplifications allow for a practical, simplified algorithm for detecting the attack. We describe this algorithm in the following text. We then describe our implementation of the algorithm as well as the limitations and assumptions of the algorithm. Finally, we present the results of several runs of the algorithm on the Tor network.

First, query the Tor directory servers for a list of all public nodes. Filter this list based upon the nodes that are flagged as *valid, running, stable, exit* nodes, as these should be most advantageous for the adversary to compromise. Call this list of nodes the *candidates.* Then, repeat the following steps l times, where larger values of l increase certainty as described in Section 3: For each candidate node, create a circuit where the first node is known to be honest, and the second is a candidate. Retrieve a file through this circuit, and log the results. Each such test either succeeds completely, or fails at some point, either during circuit creation or other initialization, or during the retrieval itself. Either failure mode could be the result of a natural failure (e.g., network outages, overloaded nodes), or an attacker implementing the DoS attack. A candidate node with a high failure rate is a *suspect*; this failure rate can be tuned with the usual tradeoff between false positives and negatives.

Once the list of suspect nodes is generated, the following steps are repeated l' times for an appropriately chosen l'. Each possible pairing of suspect nodes is used to create a circuit of length two. As above, the circuits thus created are used to perform a retrieval, and the successes and failures are logged. In this set of trials, we are looking for paths with low failure rates over the l' trials. Nodes on such paths could be under control of an adaptive adversary, and are termed *guilty.*

Also, consider a graph, where vertices are nodes, and edges exist between guilty nodes. If such paths form a clique, there are a limited number of explanations: One is that the guilty nodes are actually malicious; another is that all have good connectivity with one another, but bad connectivity with the honest node used to form the list of suspects. We suspect that larger cliques are less likely to be due to the latter, particularly if the nodes appear to be running on disparate networks. We have not verified this conjecture.

We implemented this algorithm on the Tor network, using a Tor node we had been running for several months prior as our honest node. We formed suspect lists with $l = 20$, looking for a failure rate of greater than 0.5 when retrieving a 100 kB file. We then attempted to find cliques of guilty nodes, using $l' = 10$ and a failure rate of less than 0.2.

Our results were mixed. Over the course of five days, we created and tested candidate lists once per day, tested suspects, and identified cliques of guilty nodes. Our observations are as follows. The set of candidates for any particular trial contained around two hundred nodes, although the union of all such sets contained nearly two hundred fifty nodes. Any given trial typically discovered about twenty suspect nodes, though the union of all such sets contained about fifty nodes. Two of the trials revealed cliques of size five, though the membership sets of these two cliques were disjoint.

Do these results indicate the presence of an attacker? Naively, if we were to assume all tests were independent, the likelihood of such cliques arising by chance alone is infinitesimal. However, the tests are not independent. For example, a node that is overloaded will refuse connections, or perhaps will be so congested that TCP timeouts will be reached, resulting in a failure of the stream being carried by the circuit. Such an overload will persist over time, reducing the independence between each of the l probes sent to a given node. We attempted to control for this effect by temporally spacing the probes by at least half an hour, but there is no way for a given Tor node to be sure of the reason for a failure elsewhere in a circuit.

We performed a modified version of this test. After creating the suspect list, we interleaved probes of the suspect, again as exits, through our honest node, as in the candidate probes. The purpose of these probes were to determine if the suspect nodes' performance had changed in the time since the suspect list was generated. Again, results were mixed: some suspects remained suspicious, while others had improved performance. Without exception, those whose performance improved did not appear to be guilty when examining the probes of suspect-paths.

We also performed a third version of test, where pairs of suspects were the entry and exit nodes on a path of length three. The third node was our trusted node. In this test, we saw the overall average failure rate rise back to the level observed when searching for suspect nodes. No suspicious cliques emerged.

More measurements should be performed to make more definitive statements about the presence of the denial of service attack in the Tor network. We plan to validate the simplified and general detection algorithms in a private Tor network. We further plan to perform more systematic measurements on the Tor network, depending upon the results of our validation and feedback from the Tor community.

7 Conclusion

The denial of service attack on Tor-like networks is potentially quite powerful, allowing an adversary attempting to break the anonymity of users at a rate much higher than when passively listening. Fortunately, this power comes at a price: We have shown that an attacker performing the denial of service is easily detected. We have presented an algorithm that deterministically detects such attackers with a number of probes into the network linear in the number of nodes in the network. Further, we have shown that while an attacker may choose to deny service probabilistically in an attempt to avoid detection, such an attempt is self-defeating: Most of the attacker's gain occurs as the probability of denial approaches one — lower values do not gain much over a passive approach, but are still detectable in a linear number of probes. Finally, we have presented preliminary evidence that no such attack is currently being executed within the deployed Tor network, on the basis of the background connection drop rate within Tor and on an practical version of our detection algorithm.

References

1. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium. (August 2004) 303–320
2. Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an analysis of onion routing security. In: Proceedings of ACM Workshop on Design Issues in Anonymity and Unobservability, Springer-Verlag, LNCS 2009 (July 2000) 96–114
3. Borisov, N., Danezis, G., Mittal, P., Tabriz, P.: Denial of service or denial of security? How attacks on reliability can compromise anonymity. In: Proceedings of CCS 2007. (October 2007) 92–102
4. Levine, B.N., Reiter, M.K., Wang, C., Wright, M.K.: Timing attacks in low-latency mix-based systems. In Juels, A., ed.: Proceedings of Financial Cryptography (FC '04), Springer-Verlag, LNCS 3110 (February 2004) 251–265

5. Overlier, L., Syverson, P.: Locating hidden servers. In: Proceedings of IEEE Symposium on Security and Privacy, IEEE Computer Society (2006) 100–114
6. Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against tor. In: Proceedings of ACM Workshop on Privacy in the Electronic Society, ACM (October 2007) 11–20
7. Wright, M., Adler, M., Levine, B.N., Shields, C.: Defending Anonymous Communication Against Passive Logging Attacks. In: Proc. IEEE Symposium on Security and Privacy (Oakland). (May 2003) 28–41
8. Wright, M., Adler, M., Levine, B.N., Shields, C.: An Analysis of the Degradation of Anonymous Protocols. In: Proc. ISOC Symposium Network and Distributed System Security (NDSS). (February 2002) 38–50