Detecting Denial of Service Attacks in Tor

Norman Danner Danny Krizanc Marc Liberatore

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

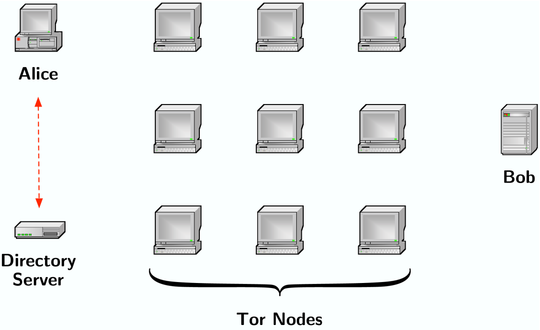
Financial Cryptography and Data Security 2009

Outline

1. Motivation
   - Background
   - Selective Denial of Service Attack

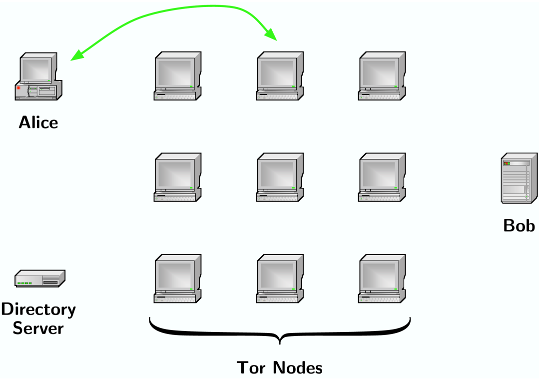
2. Contribution
   - Main Results
   - Attack Detection Algorithm
   - Handling Error
   - Detection in Practice

How Tor Works
Retrieving the list of Tor nodes



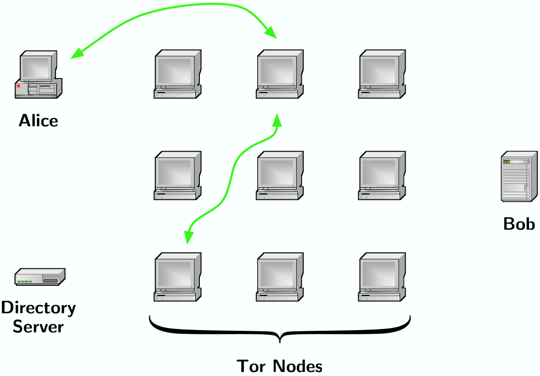
Alice retrieves a list of Tor nodes from a trusted directory server.

How Tor Works
Creating the circuit



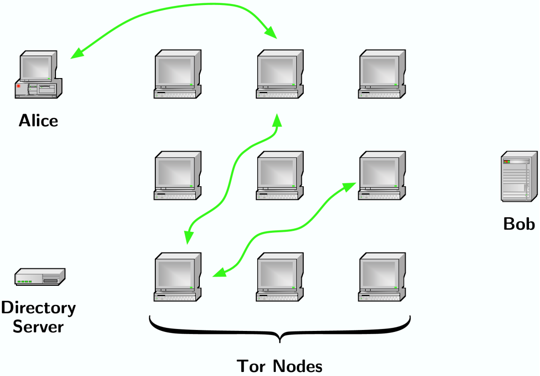
Alice chooses a node and creates a circuit.

How Tor Works
Extending the circuit



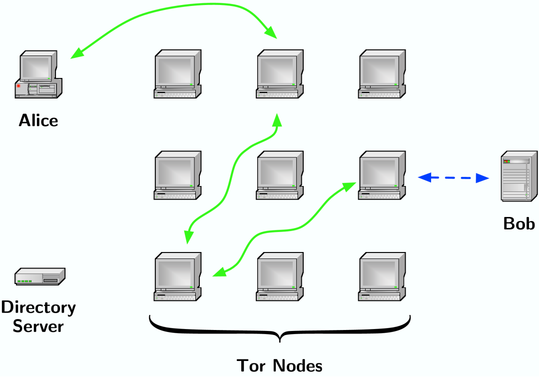
Alice instructs the current endpoint to extend the circuit.

How Tor Works
Extending the circuit again



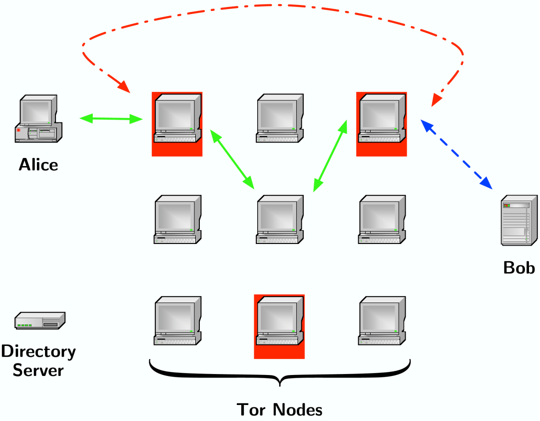
Alice instructs the new endpoint to extend the circuit.

How Tor Works
Using the circuit



Alice tunnels traffic through the circuit. Traffic is only readable between exit node and Bob.

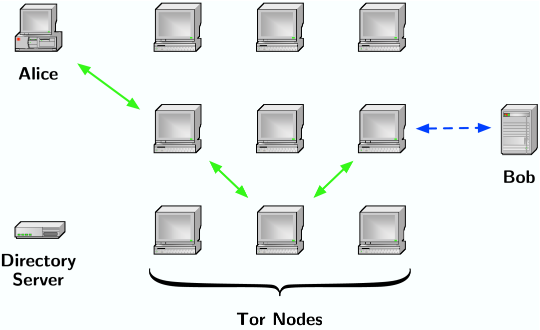
The Correlation Attack



Malicious nodes can passively collude to link Alice and Bob. $\alpha = \frac{c}{n}$ malicious implies α^2 probability of compromise.

Path Reformation

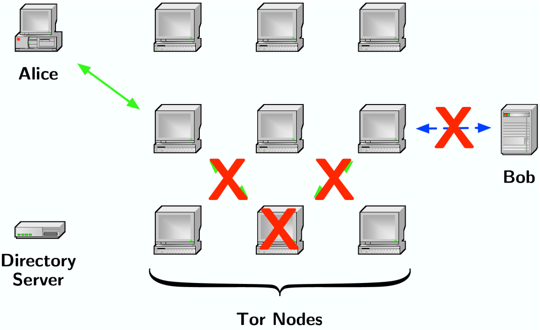
Alice forms a path



Alice is happily using Tor.

Path Reformation

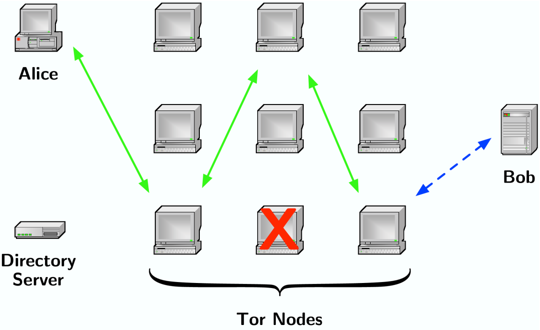
Death of a Tor node



When a node dies, Alice loses use of the circuit.

Path Reformation

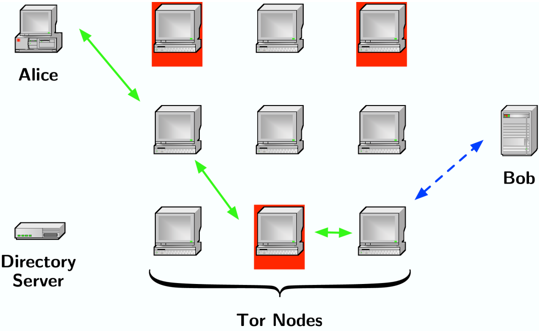
Re-forming the path



Alice will re-form a path with new Tor nodes.

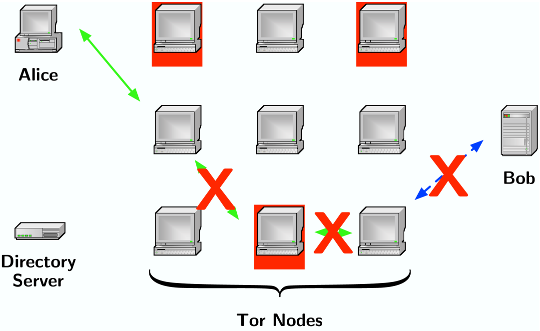
The Adaptive Adversary

The setup



Alice is using Tor, and some nodes are under attacker control.

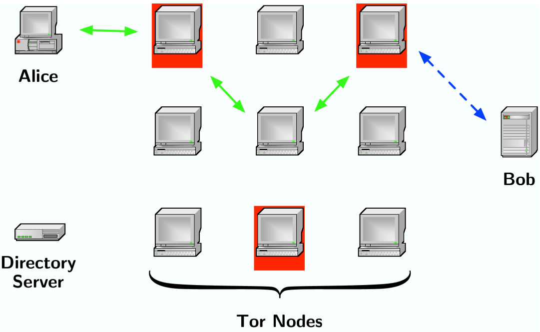
The Adaptive Adversary
The attack



Attacker kills any path where the endpoints are not under his control.

The Adaptive Adversary

The attack

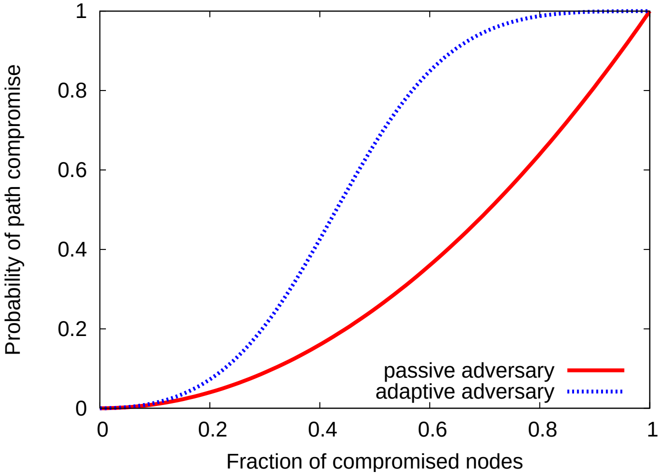


Alice is forced to make a circuit where either:

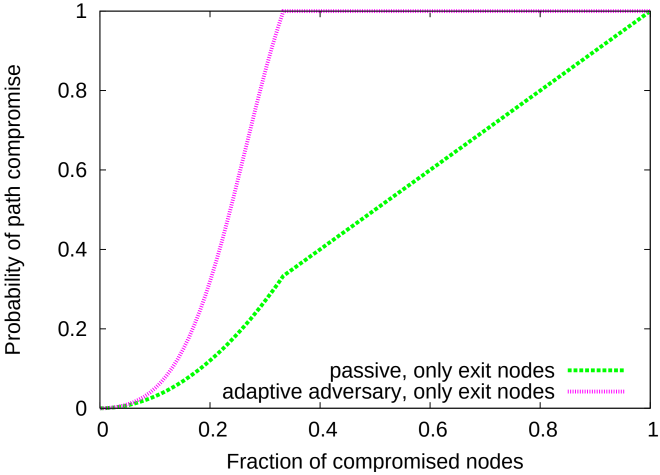
- Attacker controls endpoints, or
- No nodes are attacker controlled

The Adaptive Adversary

Power of the attack



The Adaptive Adversary
Smart adversaries control exit nodes



Main Results

Our contributions:

- An $O(n)$ algorithm for finding attackers among n participants
- An examination of a smarter attacker and ensuing arms race
- Results of examining the actual Tor network

Assumptions

- Naive attacker follows previous description.
- Deliberate circuit kills happen quickly.
- n nodes total, of which c are compromised (attacker-controlled). $2 \leq c < n$
- Circuit length k is fixed. $k < n$

Sketch of the Detection Algorithm

Choose a set of two nodes $X = \{x_1, x_2\}$ arbitrarily.
For each node y where $y \notin X$, probe the circuit (x_1, y, x_2). One of three things will happen.

Sketch of the Detection Algorithm

Case 1

All probes of circuits of the form (x_1, y, x_2) succeed.

x_1 and x_2 are compromised. For any other node y, test with the probe (x_1, x_2, y).

Sketch of the Detection Algorithm

Case 2

While probing all circuits of the form (x_1, y, x_2), at least one probe succeeds and at least one probe fails.

x_1 and x_2 are uncompromised. Any y for which the probe failed is compromised; any y where it succeeded is uncompromised.

Sketch of the Detection Algorithm

Case 3

All probes of circuits of the form (x_1, y, x_2) fail.

One of x_1, x_2 is compromised, or both are honest and all others are compromised.

Probe all circuits of the form (x_1, x_2, y) and (x_2, x_1, y). One of two things will happen.

Sketch of the Detection Algorithm

Case 3a

While probing all circuits of the form (x_1, x_2, y), at least one probe succeeds and at least one probe fails.

x_2 is uncompromised, x_1 is compromised. Any y for which the probe succeeded is compromised.
Same result holds for circuits of the form(x_2, x_1, y).

Case 3b

While probing all circuits of the form (x_1, x_2, y) and (x_1, x_2, y), all probes fail.

x_1, x_2 are honest, and all other nodes are compromised.

Proof of Algorithm's Correctness

The detection algorithm can be generalized to any fixed k.

Theorem

Under our assumptions, using $O(n)$ probes we can detect all of the compromised nodes in a network. For $k = 3$, the number of probes required is at most $3n$.

Proof.

See paper. □

What About Error?

Circuits fail for various reasons all the time:

- Network errors
- Onion shutdowns
- Attackers?

Multiple Measurements

Probability of correctness

Assume circuits have a natural failure probability of f.

Assume a probe is repeated independently l times, then $p_{\text{probe\_correct}} \geq (1 - f^l)$

If the algorithm performs m probes, $p_{\text{alg\_correct}} \geq (1 - f^l)^m$.

Multiple Measurements

Limits on error

Require correct identification (honest or compromised) $p_{\mathrm{id\_correct}} \geq (1 - \epsilon)$. Then:

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

For reasonable values in the Tor network, $l = 10$ is sufficient.

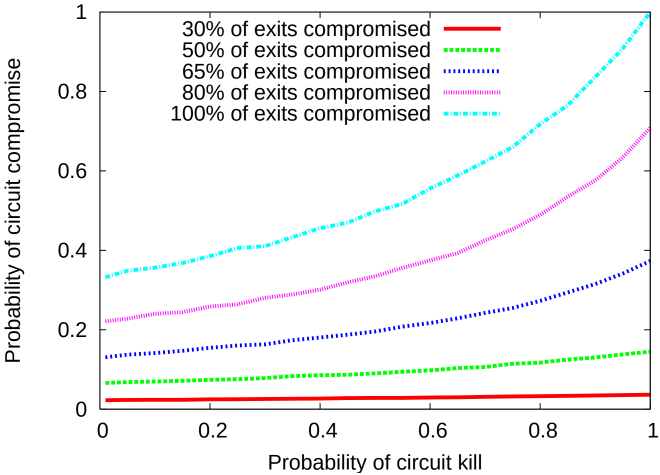
Does Selective Circuit Killing Help the Attacker?

Less frequent circuit kills help hide the attacker

A smart attacker can do the previous analysis.
Killing circuits less often:

- requires the observer to perform more probes to find the adversary (but they'll always be found, in the limit), but
- negatively impacts the attacker's performance. As $p_{\mathrm{circuit\_kill}} \to 0$, the attacker becomes the passive adversary.

Probabilistic Circuit Killing is Counterproductive



Gains come when circuit kill (and detection) is very likely!

What is a Circuit Failure?

Circuits can fail at many points:

- At any point during creation
- At the start of application-layer traffic
- During application-layer traffic

Observed Circuit Failures

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)

Simplifying Assumptions

- Assume a trustworthy guard g node not known to the adversary.
- Assume attacker only compromises exit nodes.
- Assume circuits of length 2 can be created.

A Simplified, Practical Detection Algorithm

Finding suspects

- For each Tor node y, create a circuit of the form (g, y) and attempt a file retrieval over this circuit. (Repeat l times.)
- If the file retrieval fails, add that y to the list of suspects, $s_1, s_2, \ldots$

A Simplified, Practical Detection Algorithm

Finding guilt

- For each pair of suspects, create a circuit of the form (s_i, s_j), and attempt a file retrieval over this circuit. (Repeat l' times.)
- If the file retrieval succeeds, add those suspects to the list of likely guilty nodes.
- Guilt is more likely if the guilty nodes form a clique — that is, they can communicate among one another but not with other nodes.

Results

We searched for suspects among active Tor nodes in October 2008. $l = 20$, $l' = 10$, suspicion threshold (failure rate) of 50%

- About 20 suspects per test, though 50 unique nodes were identified as suspects.
- Two to five of the suspects seemed guilty, but...
- the list of guilty suspects were typically disjoint from test-to-test! (Guilty only of transient failures?)

Weaknesses

Several problems in the study prevent us from having high confidence in the guilt of nodes:

- How independent were our trials? (We interleaved, but inter-trial delay was on the order of minutes.)
- How are failures in the network distributed? (We assumed transient failures were independent and memoryless — probably unrealistic. We also assumed the error rate we observed was natural.)
- Would a smarter attacker be watching for and attempting to foil this algorithm? (We assumed not.)

Summary

- Selective denial of service among Tor nodes can be detected in $3n$ probes.
- No strong evidence of this attack was found (last October).