THE SECOND GENERATION ONION ROUTER

Roger Dingledine
Nick Mathewson
Paul Syverson

-Presented by Arindam Paul
Motivation: Why do we need Onion Routing?
Introduction: What is TOR?
Basic TOR Design
Fairness, Congestion Control
Hidden Services
Attacks and Defense
Design Decisions
Limitations and Future Work
Conclusion
What is “wrong” with ordinary routing?

- Packet headers identify recipients
- Packet routes can be tracked
- Moreover, encryption does not hide routing information
- So, we need “anonymity” of the communication channel
Introduction

- TOR is a distributed overlay network designed to anonymize TCP-based applications.
- But, why is it named after a...

- Probably to make us “cry” 😃
What is Onion Routing?

- Routing Onion is a data structure where the data is wrapped by successive layers of encryption and then each layer is decrypted like layer of an onion (there’s the analogy).

- This makes the original text being only to sender, exit node and receiver.
How Onion Routing works?
How TOR improves on Onion Routing?

- Perfect forward secrecy: It uses a telescoping path-building design rather than a single encrypted onion
- Congestion control
- Directory Servers
- Integrity checking
- Rendezvous points and Hidden Services
TOR DESIGN
TOR DESIGN

- Cells

<table>
<thead>
<tr>
<th>CircID</th>
<th>CMD</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

509 bytes

<table>
<thead>
<tr>
<th>CircID</th>
<th>Relay</th>
<th>StreamID</th>
<th>Digest</th>
<th>Len</th>
<th>CMD</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>498</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CircID</th>
<th>Relay</th>
<th>StreamID</th>
<th>Digest</th>
<th>Len</th>
<th>CMD</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
E.g. of Cell Types

- Create: When a circuit is created from OP/OR to next OR
- Created: Acknowledgement for Create
- Relay Extend: When OP wants to extend circuit to 2\textsuperscript{nd} OR or farther
- Relay Extended: Acknowledgement for Relay Extend
- Relay Begin: To begin a relay
- Relay Connected: Ack for previous
- Relay Data: To send data down the channel
Setting up connection(s)

Create $c_1$, $E(g^{x_1})$, $g^{y_1}$, $H(K_1)$

Created $c_1$, $E(g^{x_1})$

Relay $c_1$ (Extended, $g^{y_2}$, $H(K_2)$)

Create $c_2$, $E(g^{x_2})$

Created $c_2$, $g^{y_2}$, $H(K_2)$

Relay $c_1$ (Extend, OR$_2$, $E(g^{x_1})$)
Web page Fetching

TCP Handshake

Relay $c_1$ (Begin)

Relay $c_1$ (Connected)

Relay $c_2$ (Begin)

Relay $c_2$ (Connected)
Directory Servers

- In traditional Onion Routing, each router floods its information to neighbors like typical link-state protocols.
- Delays can cause inconsistent views at different parts of the network.
- This allows attackers to exploit this lack of knowledge.
Directory Servers

- TOR provides a small group of redundant well-known onion routers to track changes in topology information
- Each such directory server acts as an HTTP server
- Clients can fetch current network state
- ORs can periodically publish state to each server
TOR IN ACTION
How Tor Works: 1

Step 1: Alice’s Tor client obtains a list of Tor nodes from a directory server.
How Tor Works: 2

Step 2: Alice’s Tor client picks a random path to destination server. **Green links** are encrypted, **red links** are in the clear.
How Tor Works: 3

Step 3: If the user wants access to another site, Alice’s Tor client selects a second random path. Again, green links are encrypted, red links are in the clear.
Integrity, Fairness and Congestion Control
Integrity Checks

- Integrity checks are only done at the edges of each stream
- SHA-1 digest of data sent and received
- First 4 bytes of digest are sent with each message for verification
- Prevents an adversary from modifying data
Fairness

- Tor uses a token bucket approach to enforce a long-term average of incoming bytes
- Volunteers don’t have to give up all their bandwidth to run Tor
- This is important as Usability is a necessity for TOR to work
- Interactive streams get preferential treatment
Congestion Control

- Circuit-level throttling
  - Packaging window: how many cells the OR can package back to OP
  - Delivery window: how many data cells can be forwarded to outside of network

- Stream-level throttling:
  - Similar to above, end-to-end
  - Also, checks for successful flushed TCP stream, and only sends “send me” relays when flushed under a set threshold by the user.
Rendezvous Points

- Bob (The Server) creates onion routes to Introduction Points (IP)
- Bob gets Service Descriptor (which includes IP) to Alice via Lookup Server
- Alice obtains Service Description (IP) via Lookup
- Alice creates Onion route to Rendezvous Point (RP)
- Alice sends RP address any authorization through IP to Bob
- If Bob chooses to talk to Alice, connects to RP
- RP makes the circuit from Alice and Bob
Hidden Services via RPs
DESIGN DECISIONS
Choices:

- Deployability and usability:
  - TOR has a simple design
  - Good usability and deployability brings more users and more anonymity
- TOR aims at preventing traffic analysis attacks and not, traffic confirmation attacks
  - TOR is a low-latency system
  - The argument is that other low-latency systems also cannot prevent traffic confirmation
- Security only in the intermediate channel and not against end-to-end attacks
• Denial of Service
  • Attackers can force an OR to do expensive cryptographic operations
  • Currently no defense
  • Suggestions: Ask clients to solve puzzles during handshaking

• Exit abuse
  • ORs can be mistaken to be originators
  • Spectrum of exit policies at nodes
  • Open exit nodes, middleman and private
ATTACKS AND DEFENCES

• Active Attacks
  • Compromising Keys
  • Run a hostile OR

• Passive Attacks
  • Observing user content
  • End-to-end timing correlation
ATTACKS AND DEFENCES

- Directory Attacks
  - Destroy directory servers
  - Subvert 1 or more directory servers

- Attacks against rendezvous points
  - Make many introduction requests
  - Compromise a rendezvous point
RESULTS, FUTURE DIRECTION & CONCLUSION
INITIAL RESULTS

- 32 nodes
  - Small, can still experiment and change
- Lookup of cnn.com (normally 0.3 sec)
  - Median: 2.8 sec, 90% finished in 5.3 sec
- Over time, TOR has increasingly been used and have been found to scale exceedingly well
LIMITATIONS & FUTURE WORK

• How often should user rotate fresh circuits?
• Fixed path lengths?
• Further specification review: more external review of Tor
• Wider-scale deployment: more users for better anonymity
CONCLUSION

- Tor is great with providing anonymity, deterring replay attacks and ensuring integrity checks. For a “low-latency” design, TOR does quite well.

- Vulnerabilities to several passive and active attacks within its network leave it in need of vast improvements and research in the future.

- Overall, TOR succeeded in achieving its objective
  - Onion Routing based designs can work and work really well !!!
QUESTIONS