Tor

An Overview of the Second-Generation Onion Router

James Lee

jlee23@umbc.edu

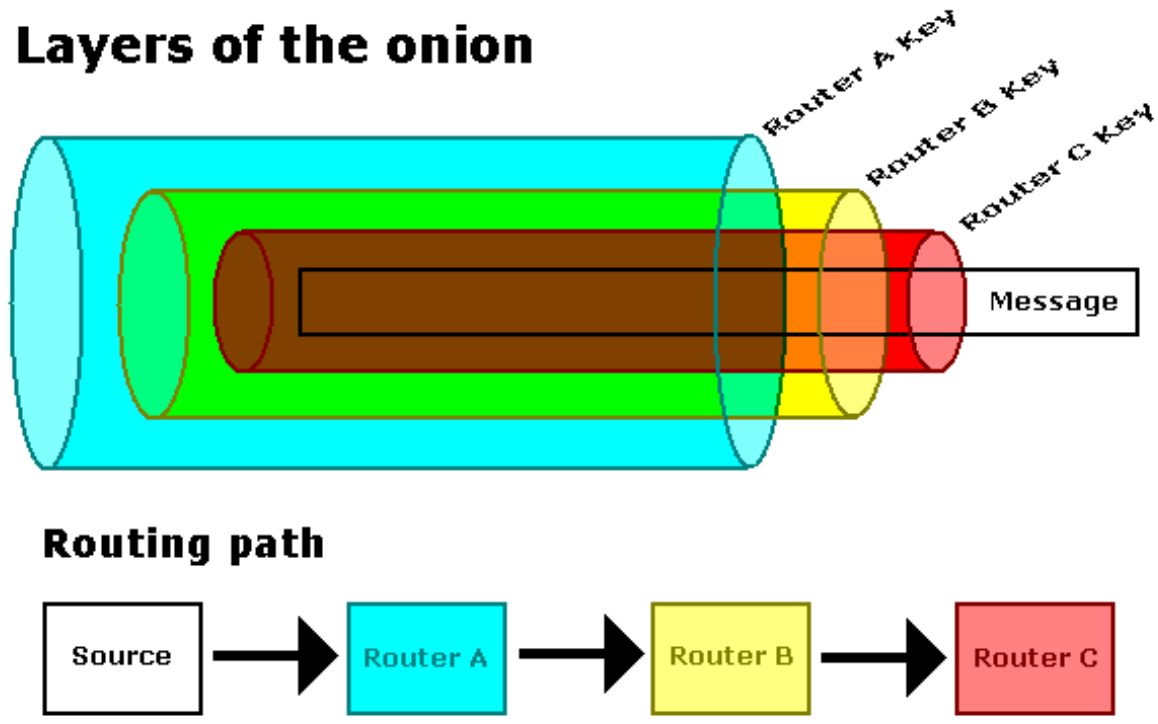
University of Maryland Baltimore County

References

[1] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

Overview of Onion Routing

- Clients choose a path through the network
- Each node only knows the predecessor and successor nodes



Improvements in Tor

- Perfect forward security

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity
- No mixing, padding, or traffic shaping

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity
- No mixing, padding, or traffic shaping
- Many TCP streams can share one circuit

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity
- No mixing, padding, or traffic shaping
- Many TCP streams can share one circuit
- Leaky-pipe circuit topology

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity
- No mixing, padding, or traffic shaping
- Many TCP streams can share one circuit
- Leaky-pipe circuit topology
- Congestion control

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity
- No mixing, padding, or traffic shaping
- Many TCP streams can share one circuit
- Leaky-pipe circuit topology
- Congestion control
- Directory servers

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity
- No mixing, padding, or traffic shaping
- Many TCP streams can share one circuit
- Leaky-pipe circuit topology
- Congestion control
- Directory servers
- Variable exit policies

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity
- No mixing, padding, or traffic shaping
- Many TCP streams can share one circuit
- Leaky-pipe circuit topology
- Congestion control
- Directory servers
- Variable exit policies
- End-to-end integrity checking

Improvements in Tor

- Perfect forward security
- Separation of “protocol cleaning” from anonymity
- No mixing, padding, or traffic shaping
- Many TCP streams can share one circuit
- Leaky-pipe circuit topology
- Congestion control
- Directory servers
- Variable exit policies
- End-to-end integrity checking
- Rendezvous points and hidden services

Design Goals

Above all, Tor seeks to frustrate attackers from linking communication partners.

Design Goals

Above all, Tor seeks to frustrate attackers from linking communication partners.

- Goals
  - Deployability
  - Usability
  - Flexibility
  - Simple Design

Design Goals

Above all, Tor seeks to frustrate attackers from linking communication partners.

- Goals
  - Deployability
  - Usability
  - Flexibility
  - Simple Design
- Non-goals
  - Not peer-to-peer
  - Not secure against end-to-end attacks
  - No protocol normalization
  - Not steganographic

Tor Design

Overview

- Each onion router (OR) runs as a normal user-level process
- Each OR maintains a TLS connection to every other OR
- Each user runs an onion proxy (OP)

Tor Design

Overview

- Each onion router (OR) runs as a normal user-level process
- Each OR maintains a TLS connection to every other OR
- Each user runs an onion proxy (OP)

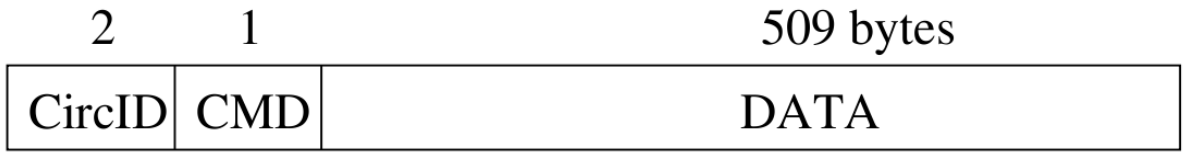
Keys

- Each OR has two keys: identity key and onion key
- Identity key signs TLS certificates and router descriptions
- Directory servers use identity keys to sign directories
- Onion keys used to decrypt circuit setup requests

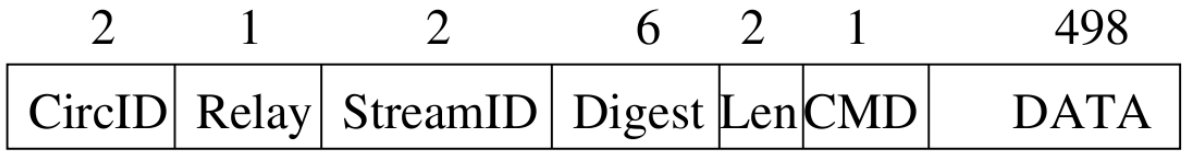
Cells

Cells are the basic unit of communication in Tor.

- “Control” cell:



- “Relay” cell:



Circuits and Streams