TOR: The Onion Router v2

Daniel Otero CSE 584M, Winter 2009

The Problem...

- We “can” protect confidentiality/integrity of transmitted data:
  - Encryption
  - Authentication
- But can we protect message meta-data?
  - Sender/Recipient
  - Timing
- Routing meta-data allows for “traffic analysis”

Why Traffic Analysis Matters

- Sender and recipient information reveals a lot:
  - Political and religious views
  - National origin
  - Professional affiliations
  - Consumer behavior
  - ...and many other things.

Motivating Privacy

- So what? Why should we care?
  - “Only bad people doing bad things need Tor”
- Wrong!
  - Privacy is a fundamental human right
- Common test: “let me see your wallet”
- Better yet: “let me see your unencrypted laptop”

TOR: Onion Routing v2.0

- Tor (**T**he **O**nion **R**outer) is an open-source low-latency anonymizing overlay network implementation
- Designed to fix many shortcomings of early onion routers
- *Not* intended to overcome:
  - centralized design
  - vulnerability to end-to-end attacks
  - protocol-specific identifiers (e.g. cookies, machine info)
  - visibility of messages/senders (i.e. not steganographic)

Flaws in “old” onion routing

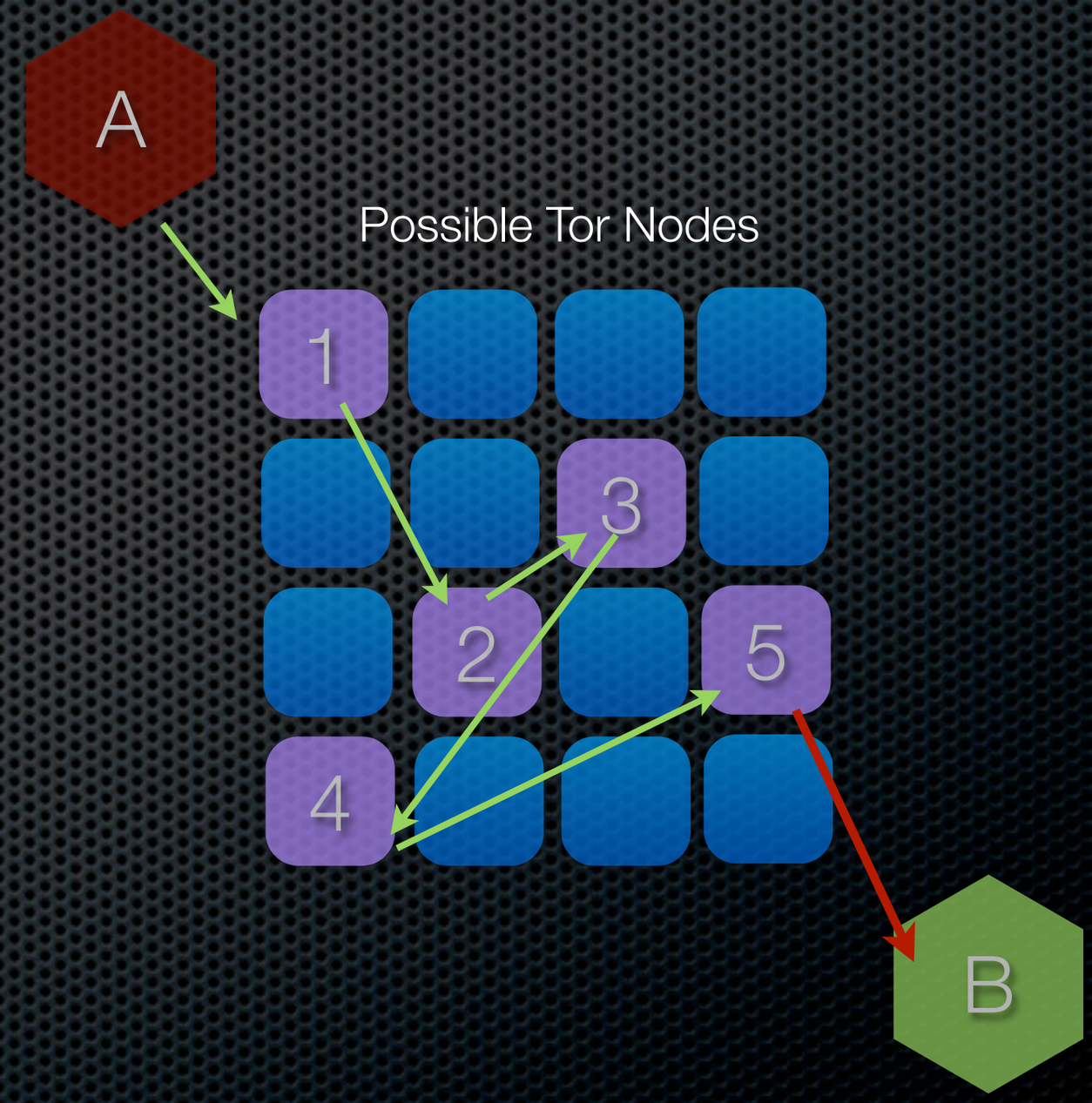
- Adversaries could record traffic, compromising downstream nodes and forcing decryption of old messages
- Each TCP stream required it’s own circuit
- Each application protocol required an application proxy
- Meta-data about network state was spread throughout
- Message integrity was never checked post-circuit
- No robust or secure way to connect to anonymous servers
- Implementations often did not run in user space

Tor's solutions

- "Perfect" forward security achieved
- TCP streams can share a circuit
- Use of SOCKS as a standard proxy interface
- "Centralized" knowledge of directory servers (trusted)
- End-to-end integrity checking
- "Hidden services": users can connect to anonymous services via agreed-upon rendezvous points
- Tor can be installed without kernel patches

Onion Routing, TOR style

- Retrieve Tor nodes from directory node
- Build a “circuit” of Tor relays
- Send message, tunneling through circuit
- Final Tor “exit” node sends original message to B (unencrypted).



Open problems (at the time)

- Denial-of-service
  - Rate limits and congestion control in place
  - Doesn’t prevent forced cryptographic computation
- Exit abuse
- End-to-end analysis
- Scaling
- Incentive to contribute
- Performance

Give Tor a try...

- http://www.torproject.org/
- Free download for Windows and Mac OS X

- You may also want:
  - Firefox

  - Torbutton (turns on/off FF's use of Tor)
    - https://addons.mozilla.org/en-US/firefox/addon/2275