Tor:
Anonymous Communications for the Dept of Defense...and you.

Roger Dingledine
Free Haven Project
Electronic Frontier Foundation

http://tor.eff.org/

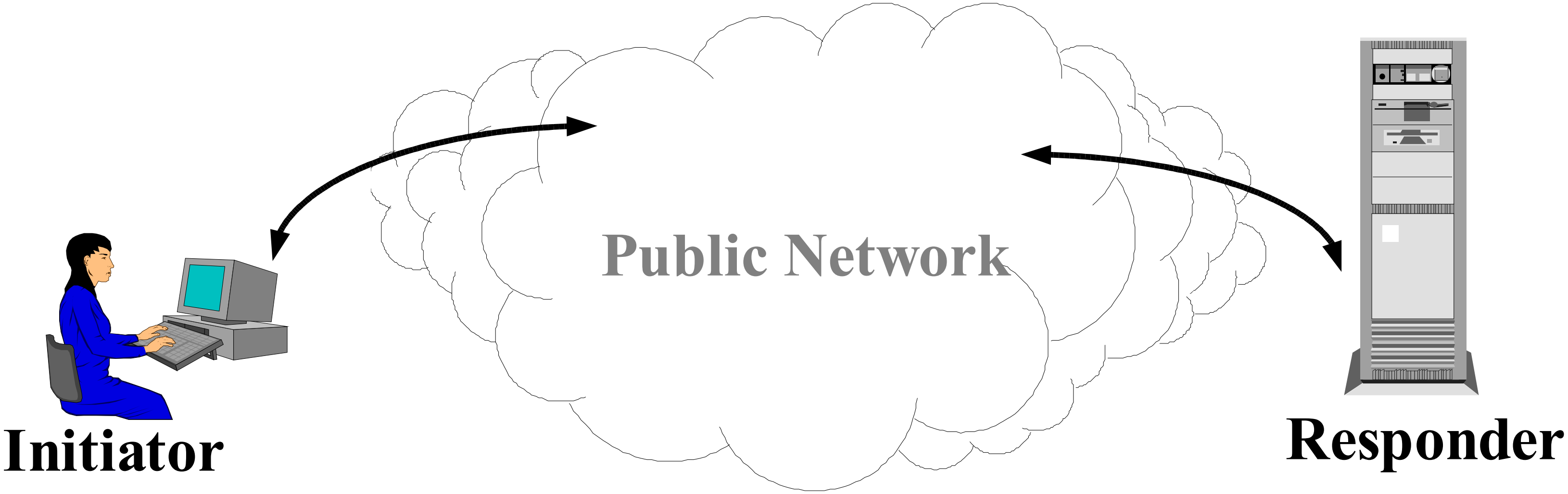
21 July 2005

Talk Outline

- Motivation: Why anonymous communication?
  - Myth 1: This is only for privacy nuts.
  - Myth 2: This stuff enables criminals.
- Tor design overview
- Hidden servers and rendezvous points
- Policy issues raised
- Open technical issues and hard problems

Public Networks are Vulnerable to Traffic Analysis

- In a Public Network (Internet):
- Packet (message) headers identify recipients
- Packet routes can be tracked



Encryption does *not* hide routing information.

Who Needs Anonymity?

- Journalists, Dissidents, Whistleblowers (Indymedia, bloggers, Iran, Tibet)
- Censorship resistant publishers/readers (libraries)
- Socially sensitive communicants:
  - Chat rooms and web forums for abuse survivors, people with illnesses
- Law Enforcement: (In-q-tel, Nye Kripos)
  - Anonymous tips or crime reporting
  - Surveillance and honeypots (sting operations)

Who Needs Anonymity?

- Corporations: (Google, Wal-Mart, ...)
  - Who's talking to the company lawyers? Are your employees looking at monster.com?
  - Hiding procurement suppliers or patterns
  - Competitive analysis

Who Needs Anonymity?

- You:
  - Where are you sending email (who is emailing you)
  - What web sites are you browsing
  - Where do you work, where are you from
  - What do you buy, what kind of physicians do you visit, what books do you read, ...

Who Needs Anonymity?

- Government

Government Needs Anonymity? Yes, for…

- Open source intelligence gathering
  - Hiding individual analysts is not enough
  - That a query was from a govt. source may be sensitive
- Defense in depth on open and *classified* networks
  - Networks with only cleared users (but a million of them)
- Dynamic and semitrusted international coalitions
  - Network can be shared without revealing existence or amount of communication between all parties
- Elections and voting

Anonymity Loves Company

- You can't be anonymous by yourself.
  - *Can* have confidentiality by yourself.
- A network that protects only DoD network users won't hide that connections from that network are from DoD.
- You must carry traffic for others to protect yourself.
- But those others don't want to trust their traffic to just one entity either. Network needs *distributed trust*.
- Security depends on diversity and dispersal of network.

Who Needs Anonymity?

- And yes criminals

Who Needs Anonymity?

- And yes criminals

But they already have it.
We need to protect everyone else.

Privacy and Criminals

- Criminals have privacy
  - Motivation to learn
  - Motivation to buy
  - Identity theft
- Normal People, Companies, Governments, Police don’t
- The worst of all possible worlds

Privacy and Hackers

- Hackers have privacy
  - Break into system
  - Destroy the logs
  - Repeat as needed
  - They don’t use or need our software
- Normal People, Companies, Governments, Police don’t
- The worst of all possible worlds

Anonymous From Whom?
Adversary Model

- Recipient of your message
- Sender of your message

=> Need Channel and Data Anonymity

- Observer of network from outside
- Network Infrastructure (Insider)

=> Need Channel Anonymity

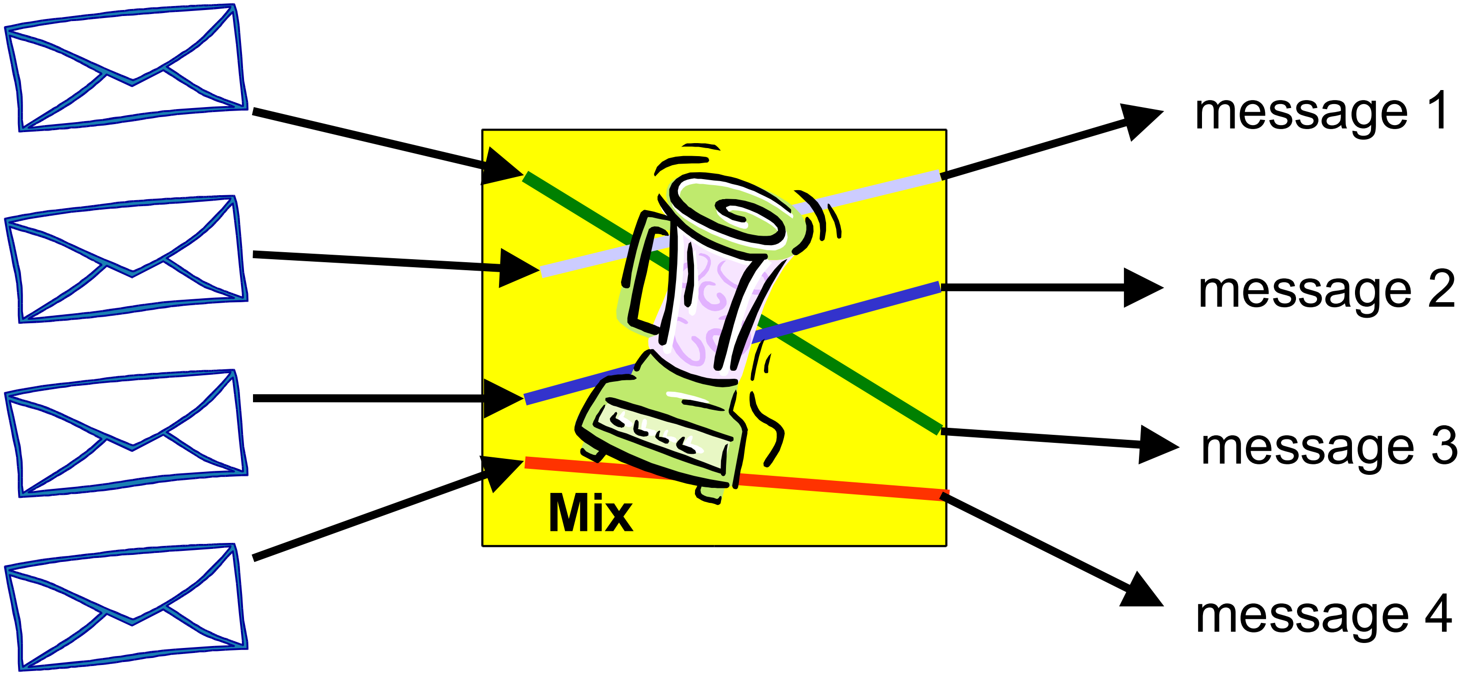
- Note: Anonymous authenticated communication makes perfect sense
- Communicant identification should be inside the basic channel, not a property of the channel

Focus of Tor is anonymity of the communication pipe, not what goes through it

How Do You Get Communication Anonymity?

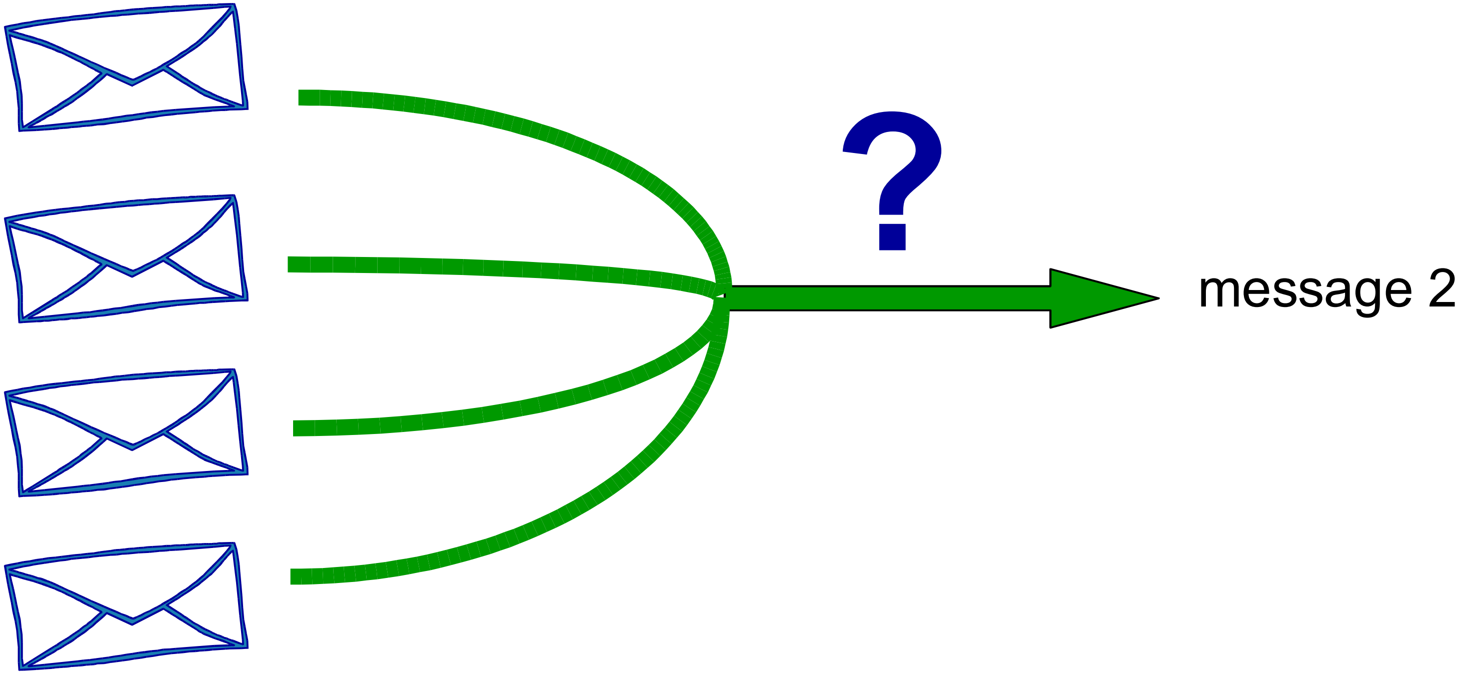
- Many technical approaches
- Overview of two extensively used approaches
  - Mixes
  - Proxies

What does a mix do?



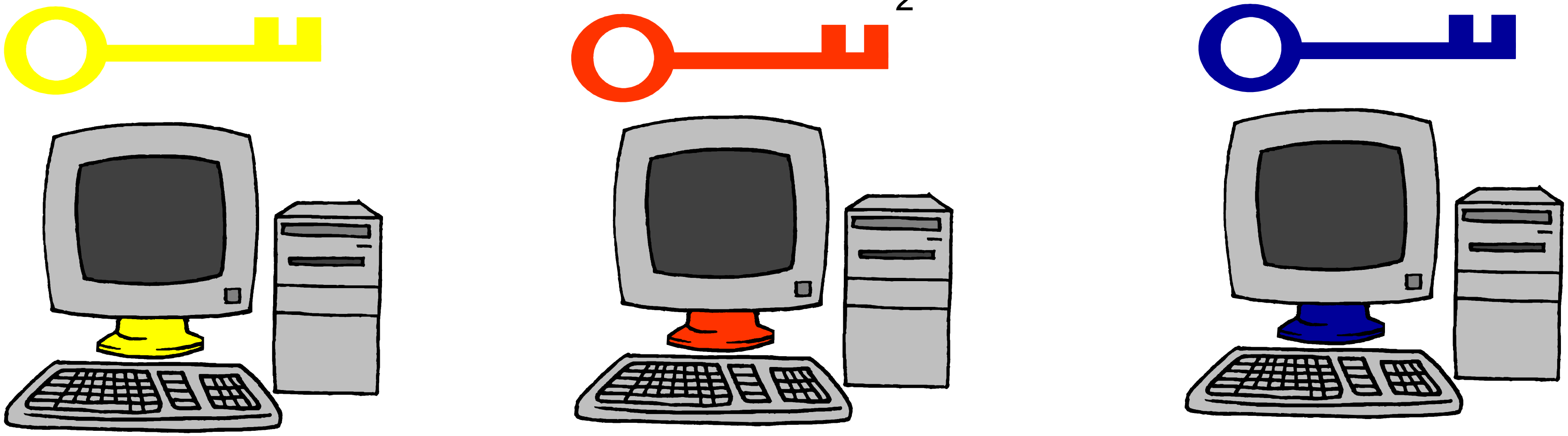
Randomly permutes and decrypts inputs

What does a mix do?



Key property: Adversary can't tell which ciphertext corresponds to a given message

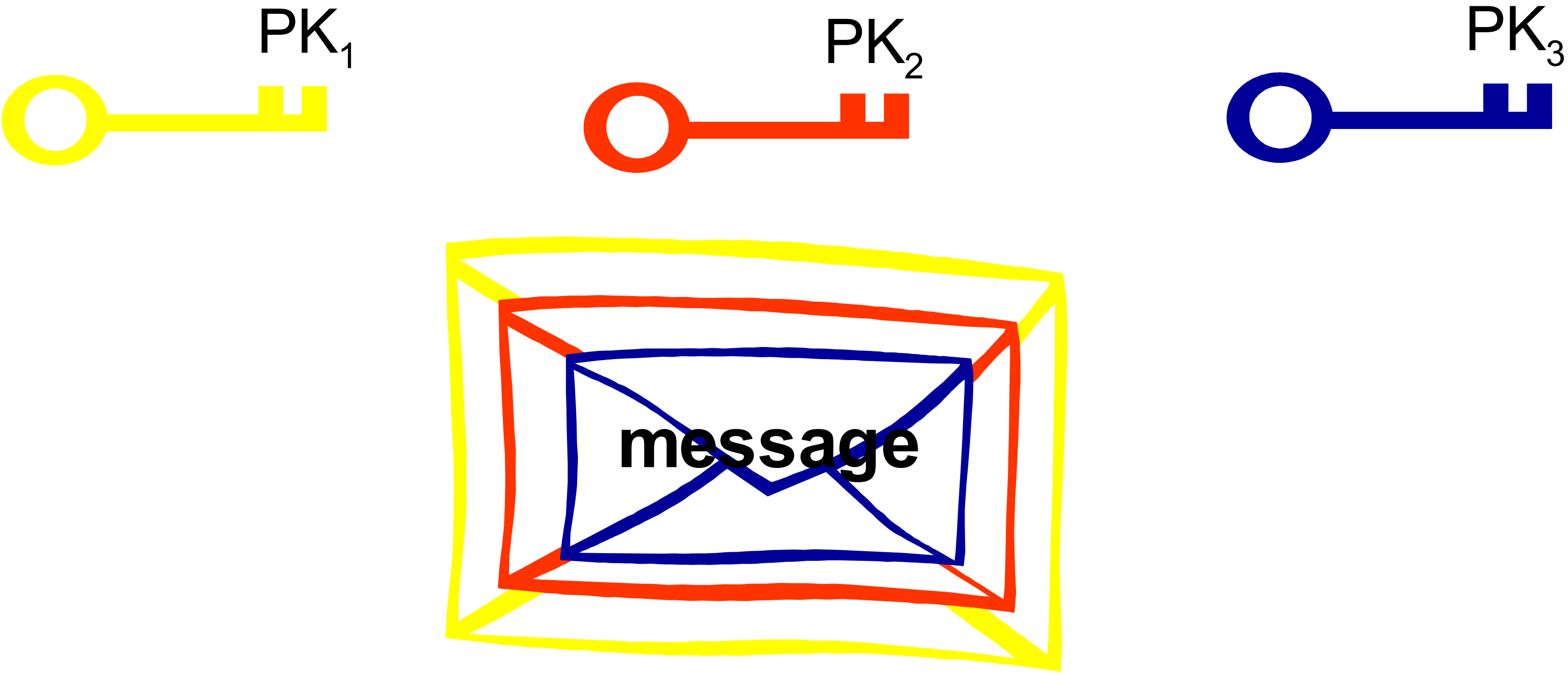
Basic Mix (Chaum ‘81)



PK_1 PK_2 PK_3

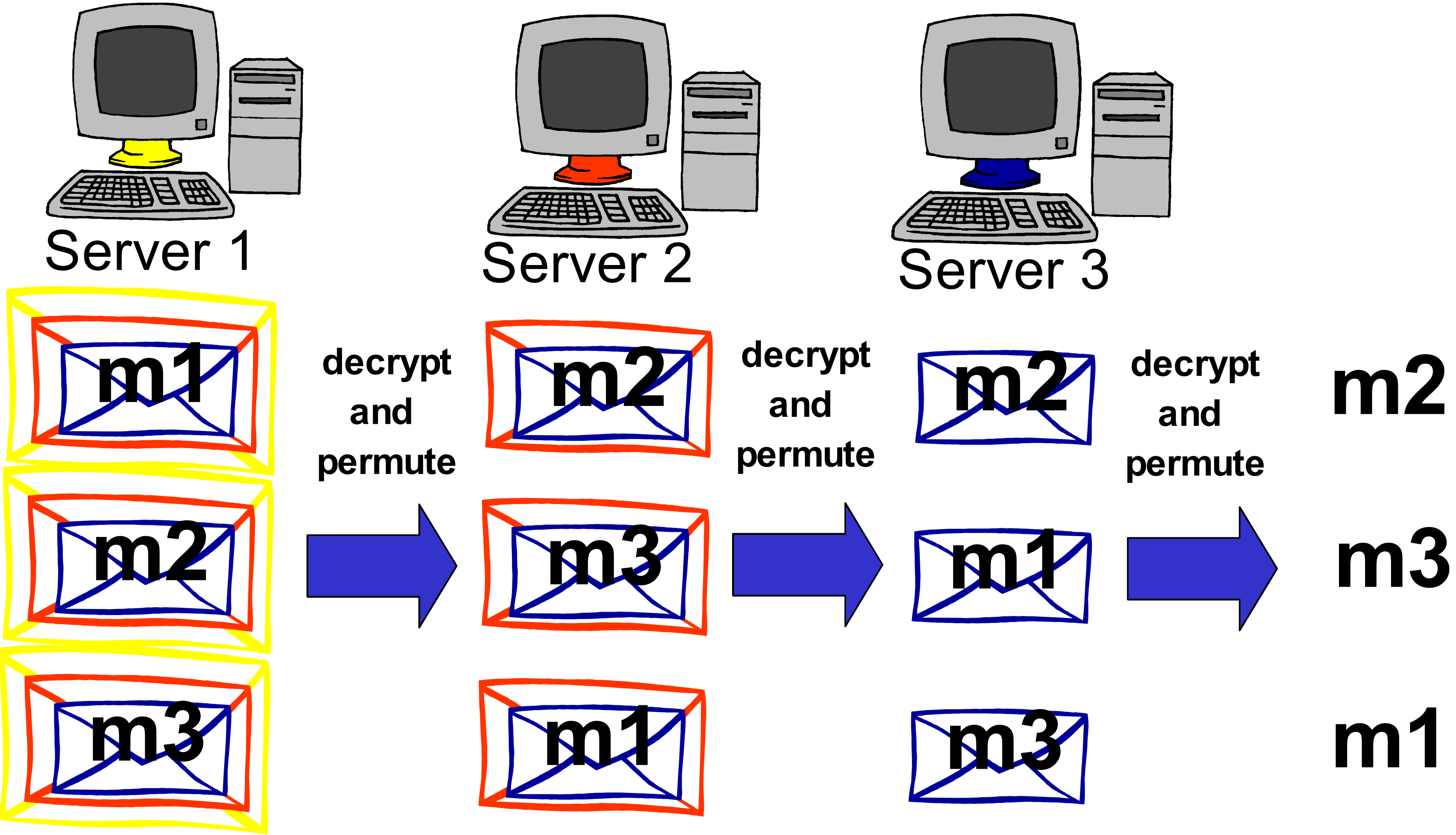
Server 1 Server 2 Server 3

Encryption of Message



Ciphertext = $E_{PK1}[E_{PK2}[E_{PK3}[\text{message}]]]$

Basic Chaum-type Mix



Server 1

Server 2

Server 3

m1

m2

m3

decrypt
and
permute

m2

m3

m1

decrypt
and
permute

m2

m1

m3

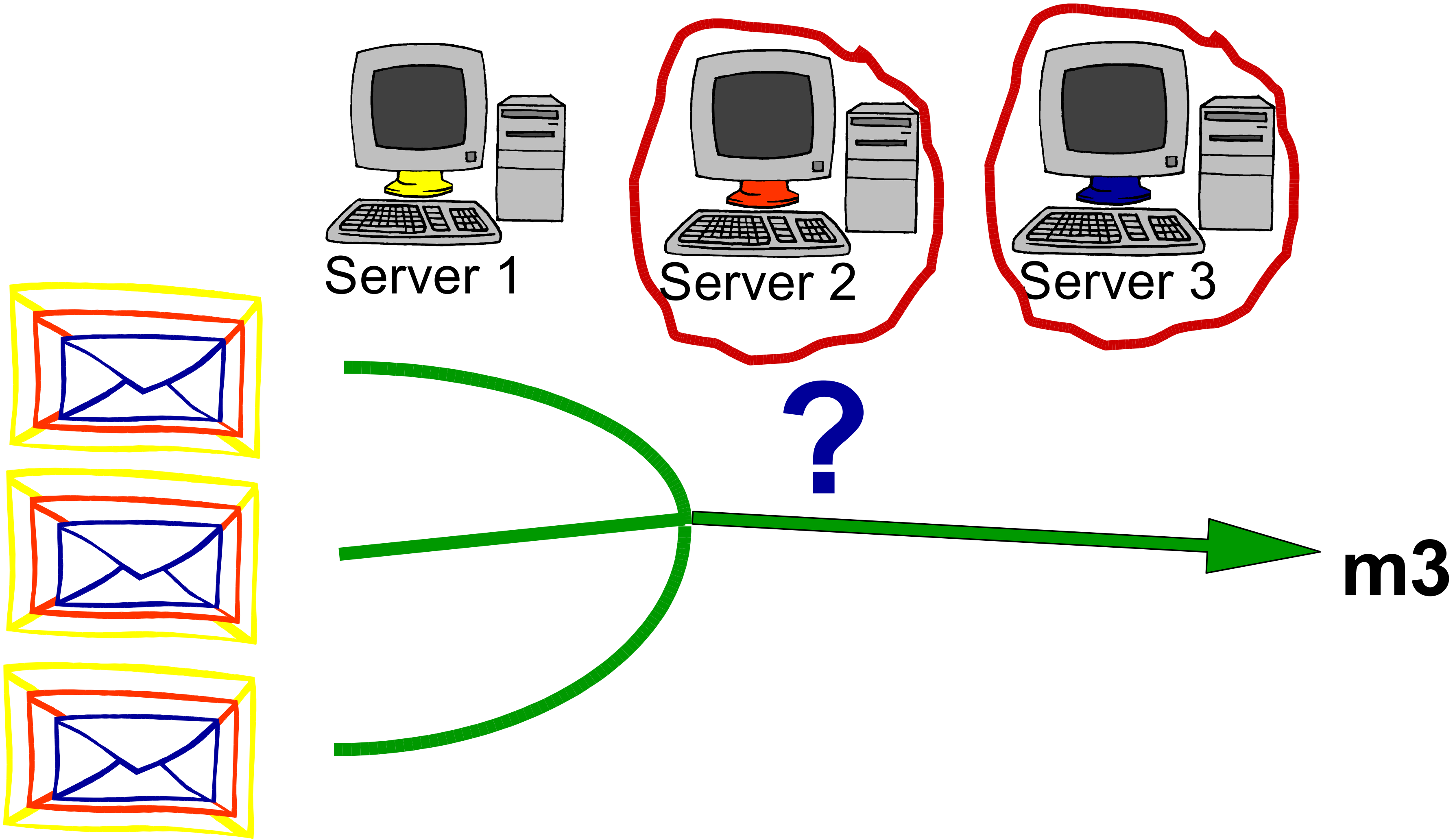
decrypt
and
permute

m2

m3

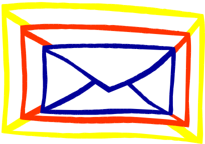
m1

One honest server preserves privacy

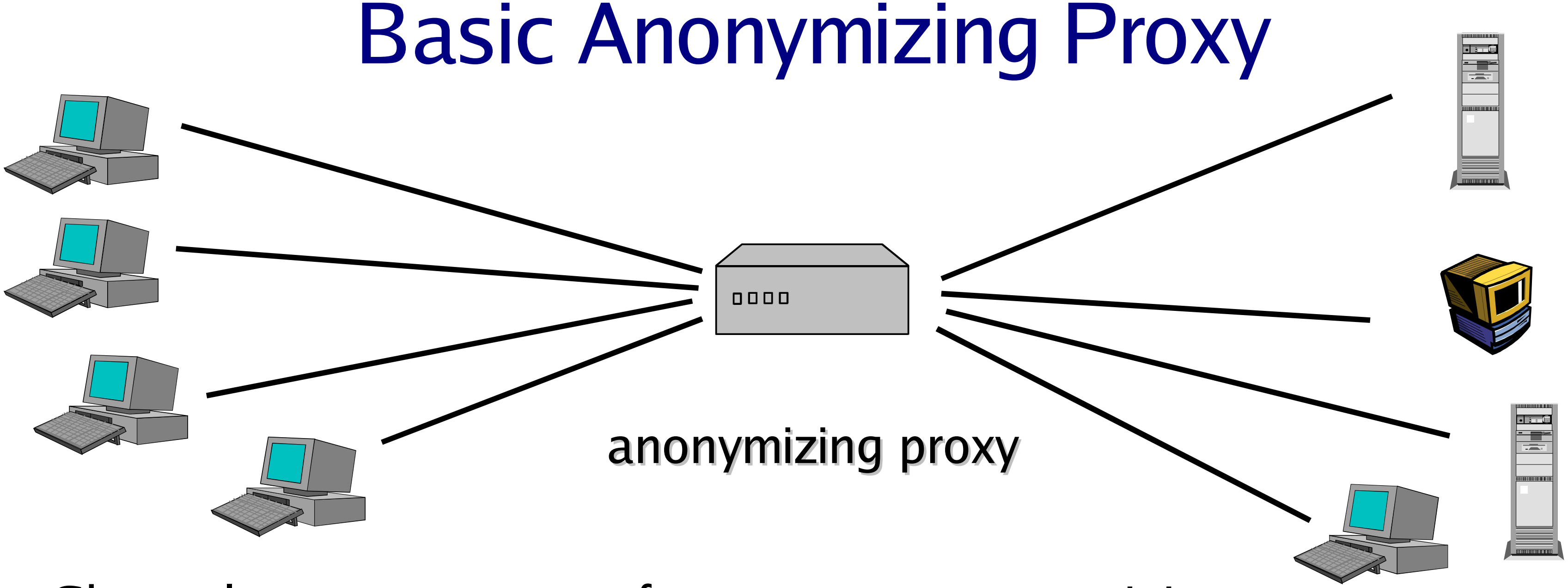


What if you need quick interaction?

- Web browsing, Remote login, Chat, etc.
- Mixnets introduced for email and other high latency apps
- Each layer of message requires expensive public-key crypto



Basic Anonymizing Proxy



- Channels appear to come from proxy, not true originator
- Appropriate for Web connections, etc.:
  SSL, TLS, SSH (lower cost symmetric encryption)
- Examples: The Anonymizer
- Advantages: Simple, Focuses lots of traffic for more anonymity
- Main Disadvantage: Single point of failure, compromise, attack

Onion Routing
Traffic Analysis Resistant Infrastructure

- Main Idea: Combine Advantages of mixes and proxies
- Use (expensive) public-key crypto to establish circuits
- Use (cheaper) symmetric-key crypto to move data
  - Like SSL/TLS based proxies
- Distributed trust like mixes
- Related Work (some implemented, some just designs):
  - ISDN Mixes
  - Crowds, JAP Webmixes, Freedom Network
  - Tarzan, Morphmix

Tor

Tor

The Onion Router

Tor

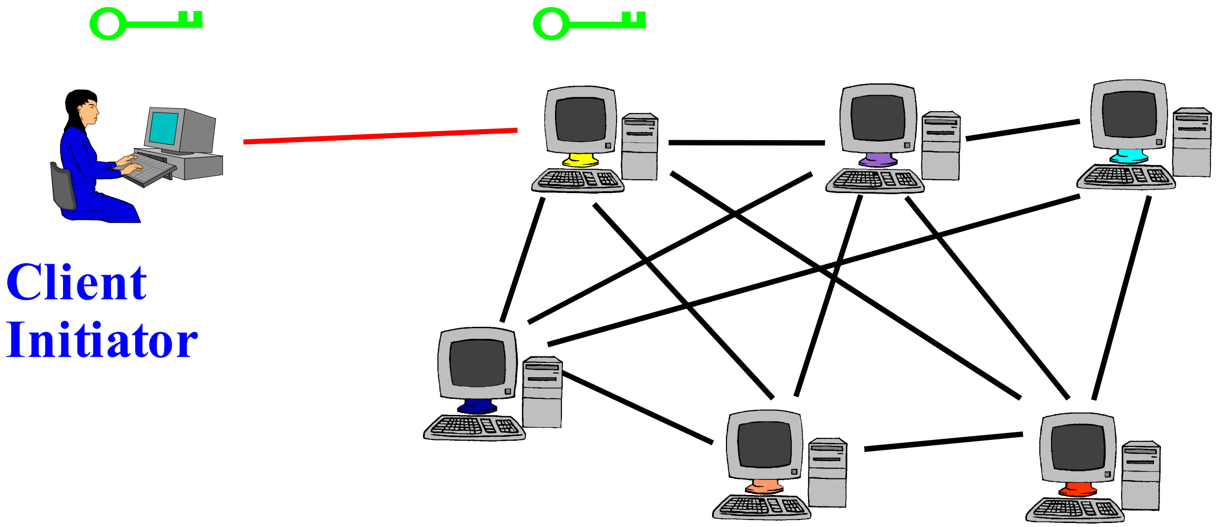
Tor's Onion Routing

Numbers and Performance

- Running since October 2003
- 250 nodes on five continents (North America, South America, Europe, Asia, Australia)
- Volunteer-based infrastructure
- Fifty thousand+ (?) users
- Nodes process 1-100 GB / day application cells
- Network has never been down

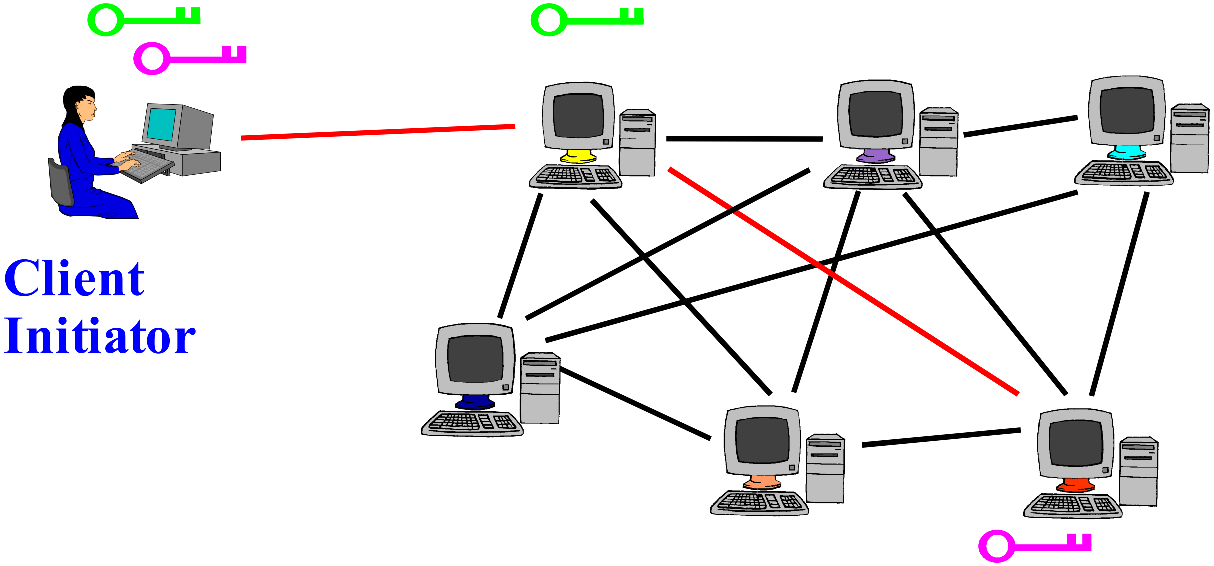
Tor Circuit Setup

- Client Proxy establishes session key + circuit w/ Onion Router 1



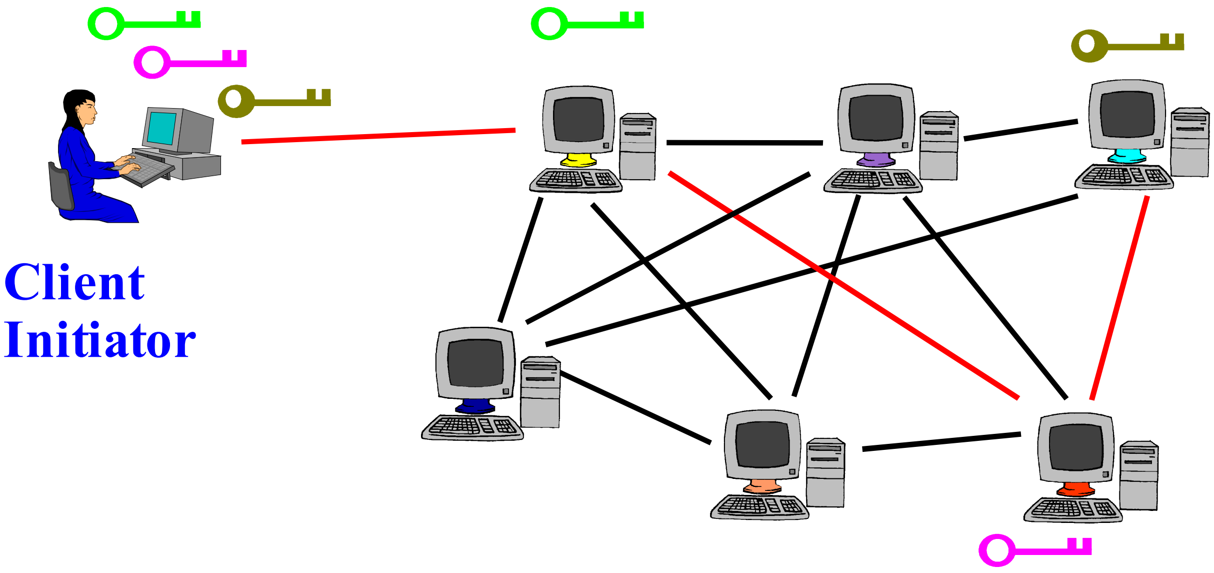
Tor Circuit Setup

- Client Proxy establishes session key + circuit w/ Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2



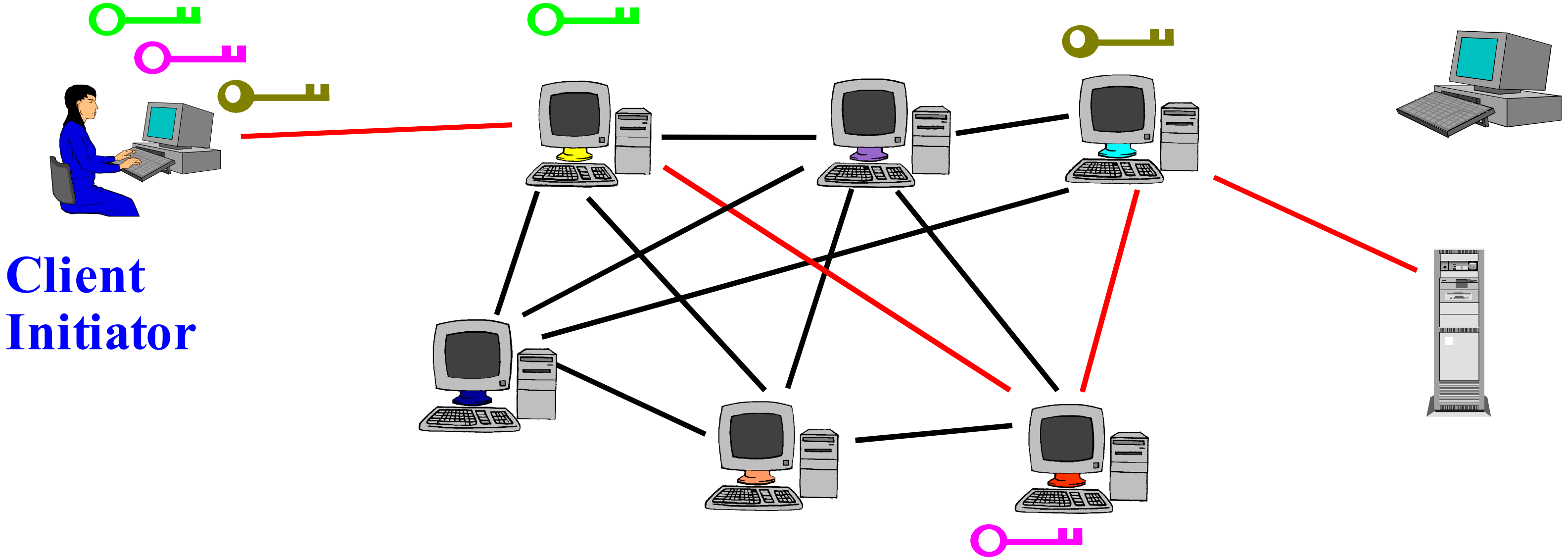
Tor Circuit Setup

- Client Proxy establishes session key + circuit w/ Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc



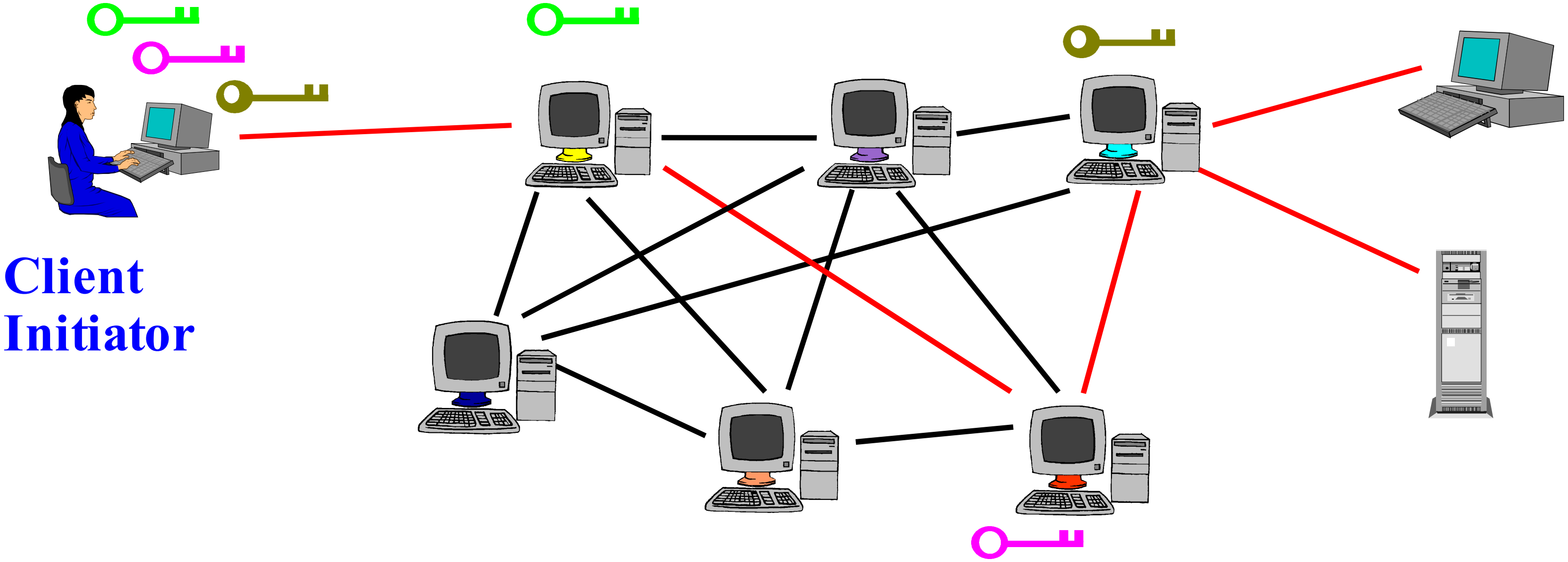
Tor Circuit Usage

- Client Proxy establishes session key + circuit w/ Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc
- Client applications connect and communicate over Tor circuit



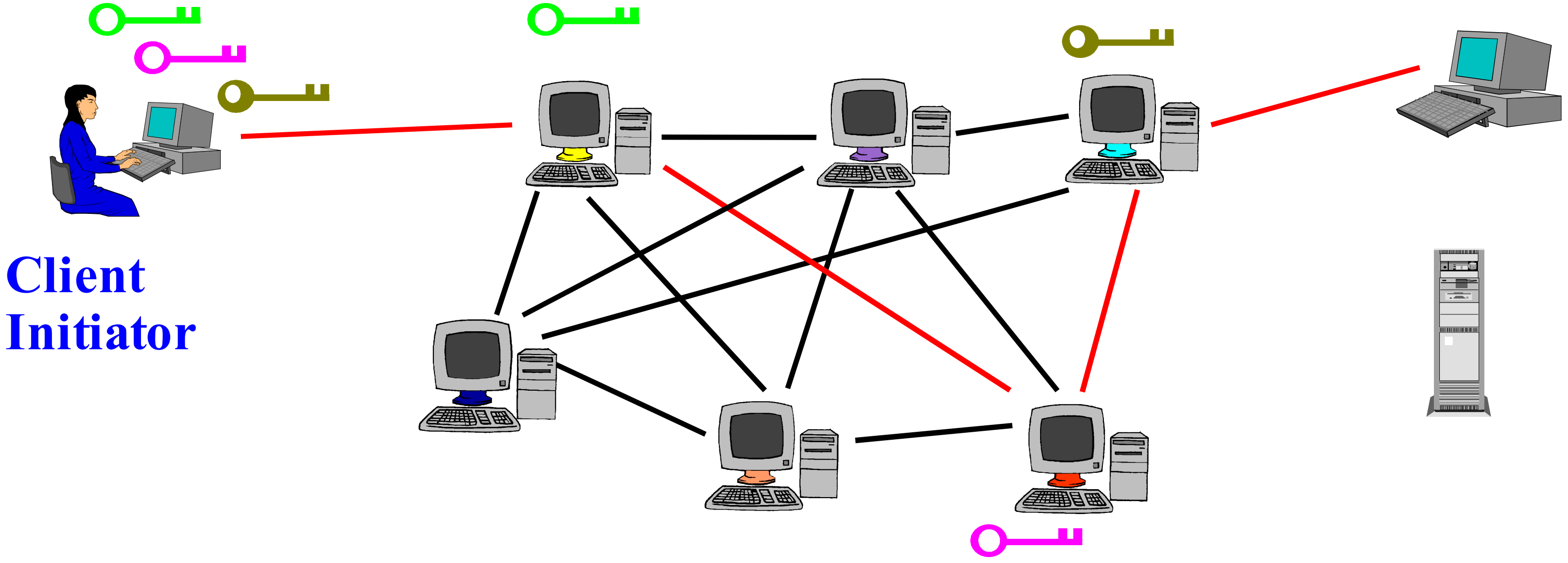
Tor Circuit Usage

- Client Proxy establishes session key + circuit w/ Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc
- Client applications connect and communicate over Tor circuit



Tor Circuit Usage

- Client Proxy establishes session key + circuit w/ Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc
- Client applications connect and communicate over Tor circuit



Where do I go to connect to the network?

- Directory Servers
  - Maintain list of which onion routers are up, their locations, current keys, exit policies, etc.
  - Directory server keys ship with the code
  - Control which nodes can join network
    - Important to guard against “Sybil attack” and related problems
  - These directories are cached and served by other servers, to reduce bottlenecks
  - Need to decentralize, get humans out of the loop, without letting attackers sign up 100,000 nodes.

Some Tor Properties

- Simple modular design, restricted ambitions.
  - ~40K lines of C code
  - Even servers run in user space, no need to be root
  - Flexible exit policies, each node chooses what applications/destinations can emerge from it
  - Server usability is key to adoption. Without a network, we are nothing.

Some Tor Properties

- Simple modular design, restricted ambitions.
  - Just anonymize the pipe
    - Can use, e.g., privoxy as front end if desired to anonymize data
  - SOCKS compliant TCP: includes Web, remote login, mail, chat, more
    - No need to build proxies for every application

Some Tor Properties

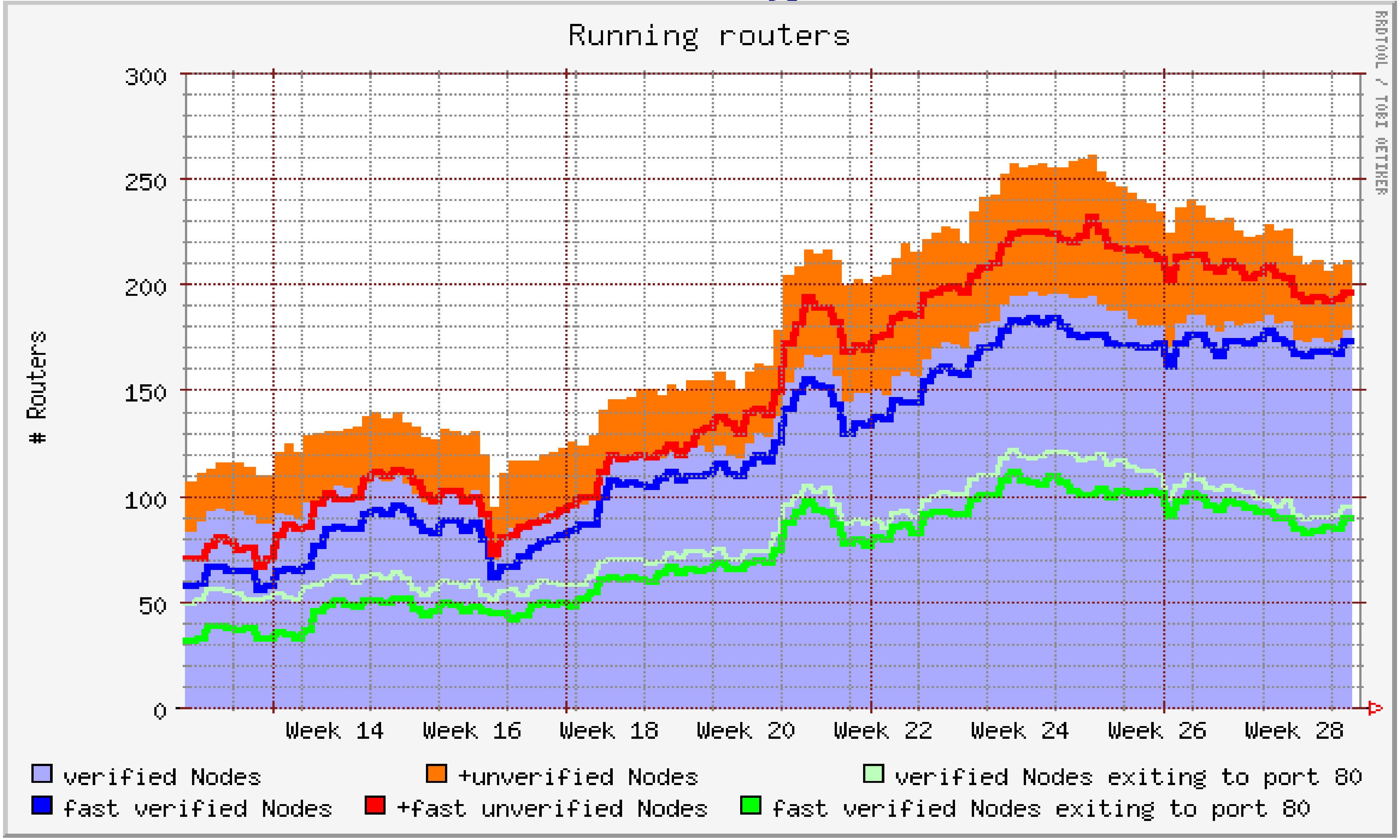
- Lots of supported platforms:

  Linux, BSD, MacOS X, Solaris, Windows, ...

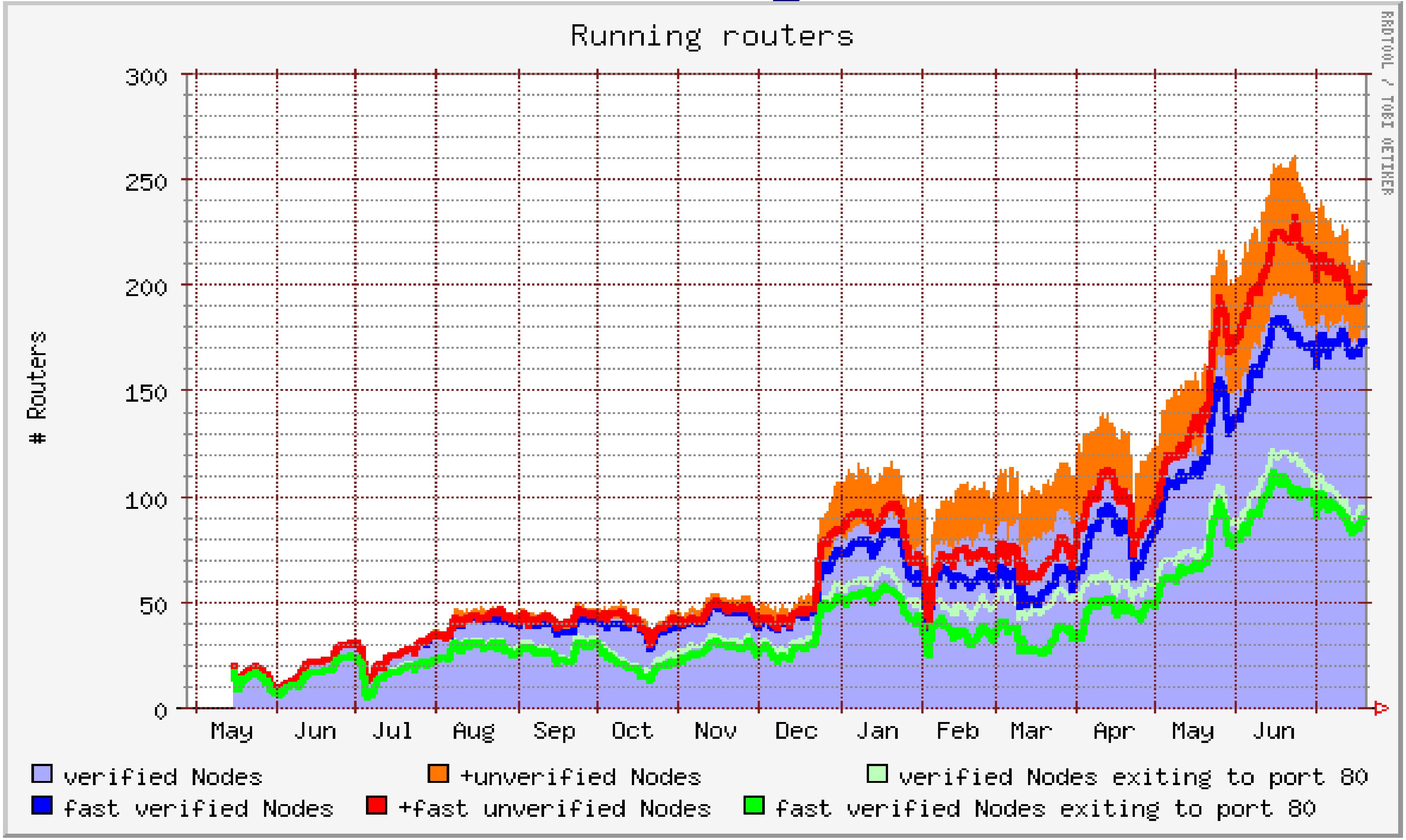
(Tor servers on xbox, linksys wireless routers.)

- Deployment paradigm:
  - Volunteer server operators
  - No payments, not proprietary
  - Moving to a P2P incentives model

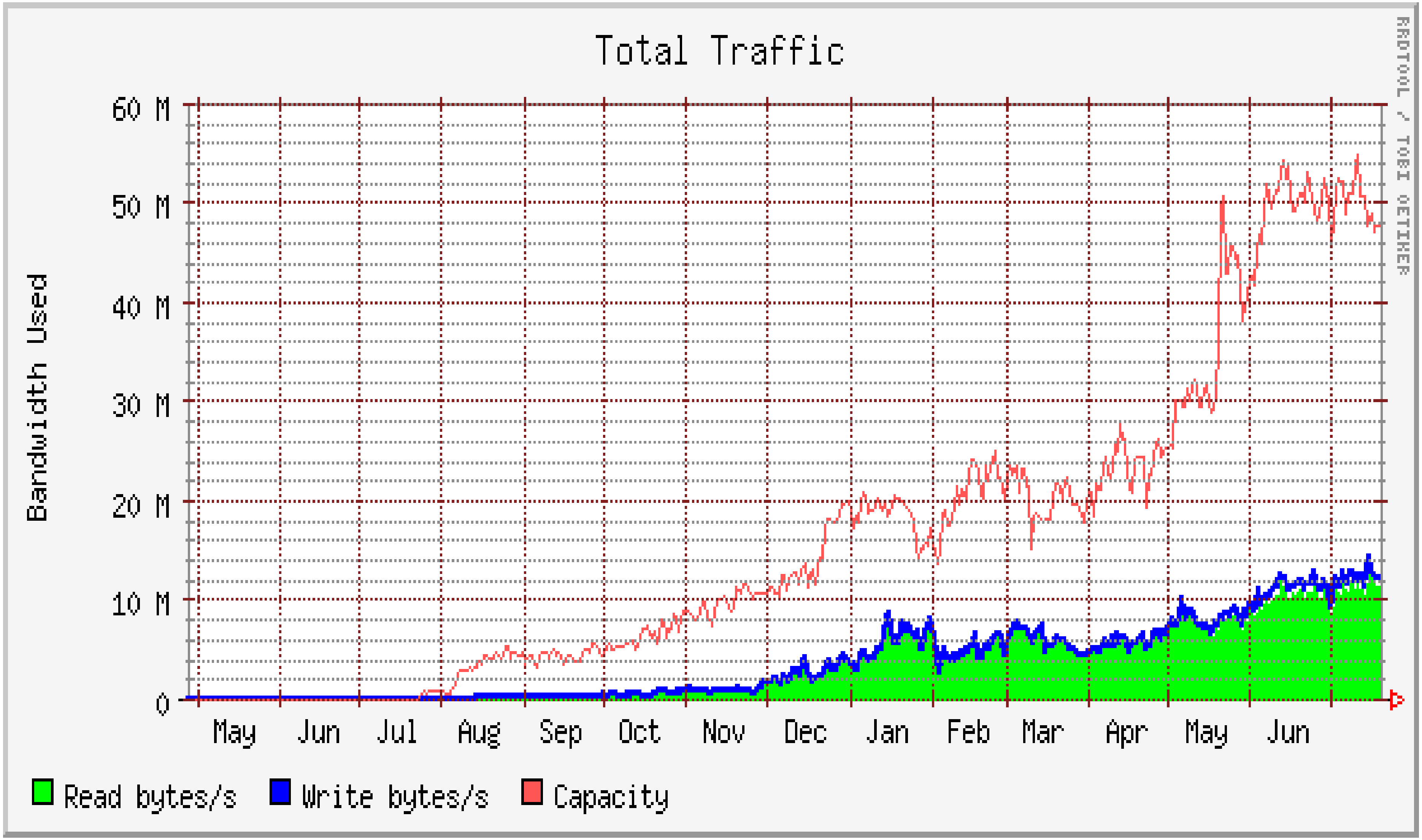
Number of running Tor servers



Number of running Tor servers



Total traffic through Tor network

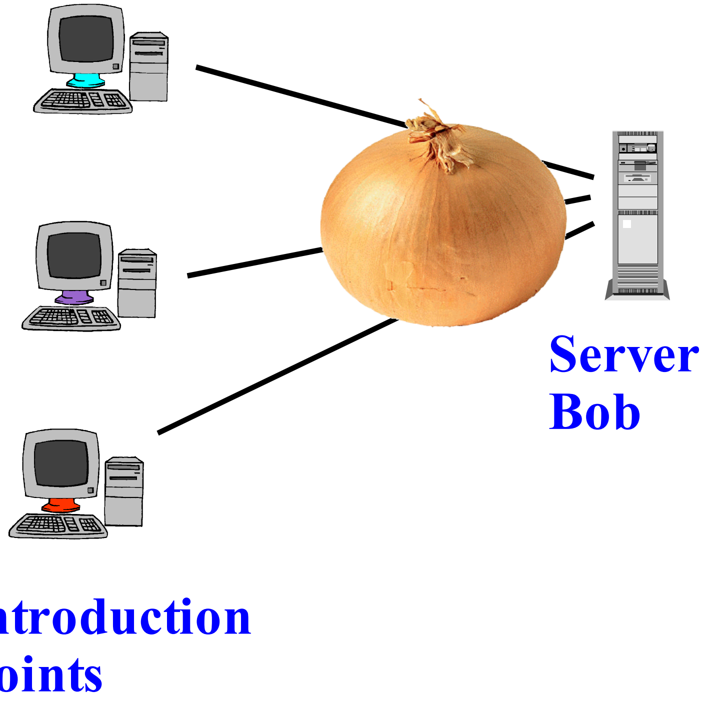


Location Hidden Servers

- Alice can connect to Bob's server without knowing where it is or possibly who he is
- Can provide servers that
  - Are accessible from anywhere
  - Resist censorship
  - Require minimal redundancy for resilience in denial of service (DoS) attack
  - Can survive to provide selected service even during full blown distributed DoS attack
  - Resistant to physical attack (you can't find them)
- How is this possible?

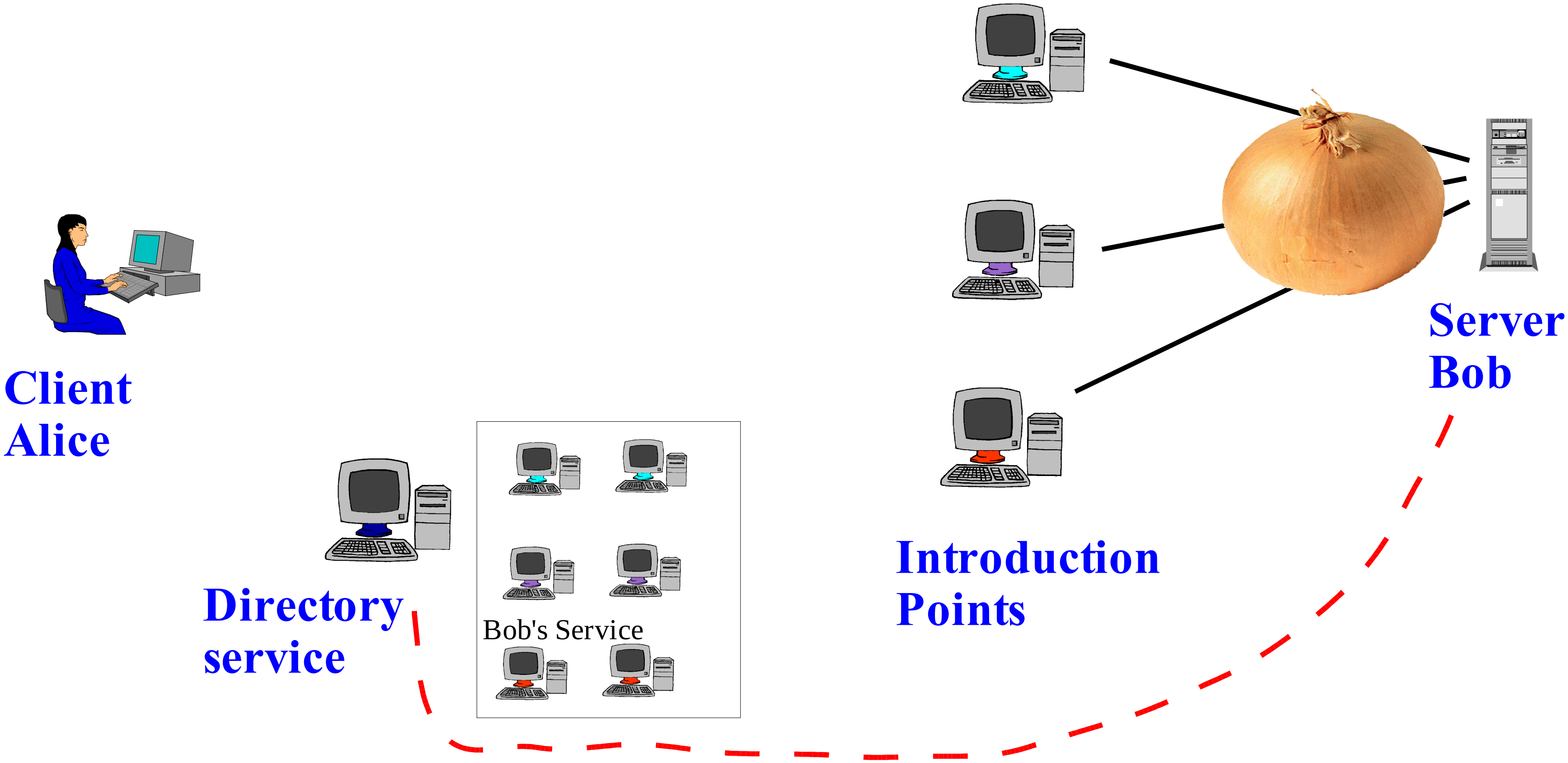
Location Hidden Servers

1. Server Bob creates onion routes to Introduction Points (IP)



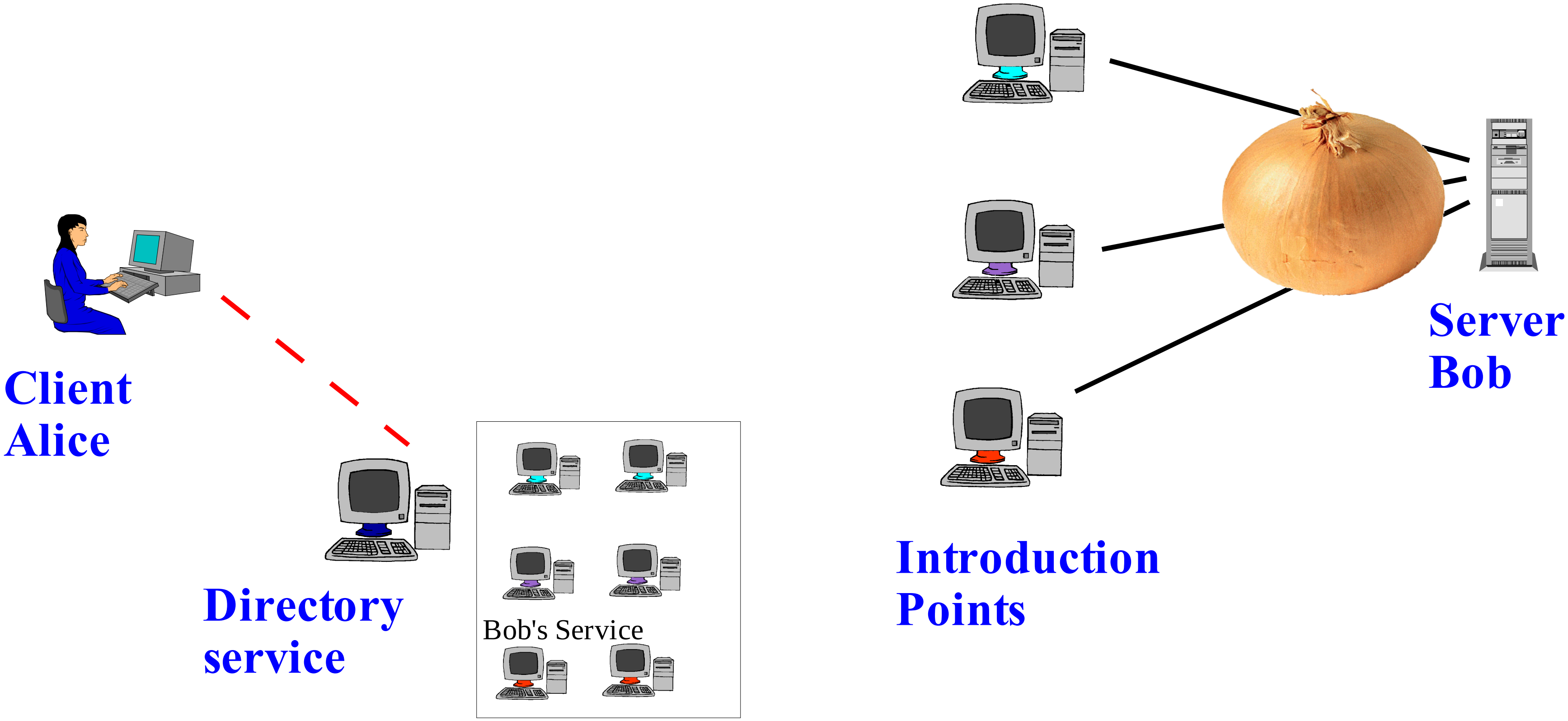
Location Hidden Servers

1. Server Bob creates onion routes to Introduction Points (IP)
2. Bob gets Service Descriptor incl. Intro Pt. addresses to Alice
   - In this example gives them to Service Lookup Server



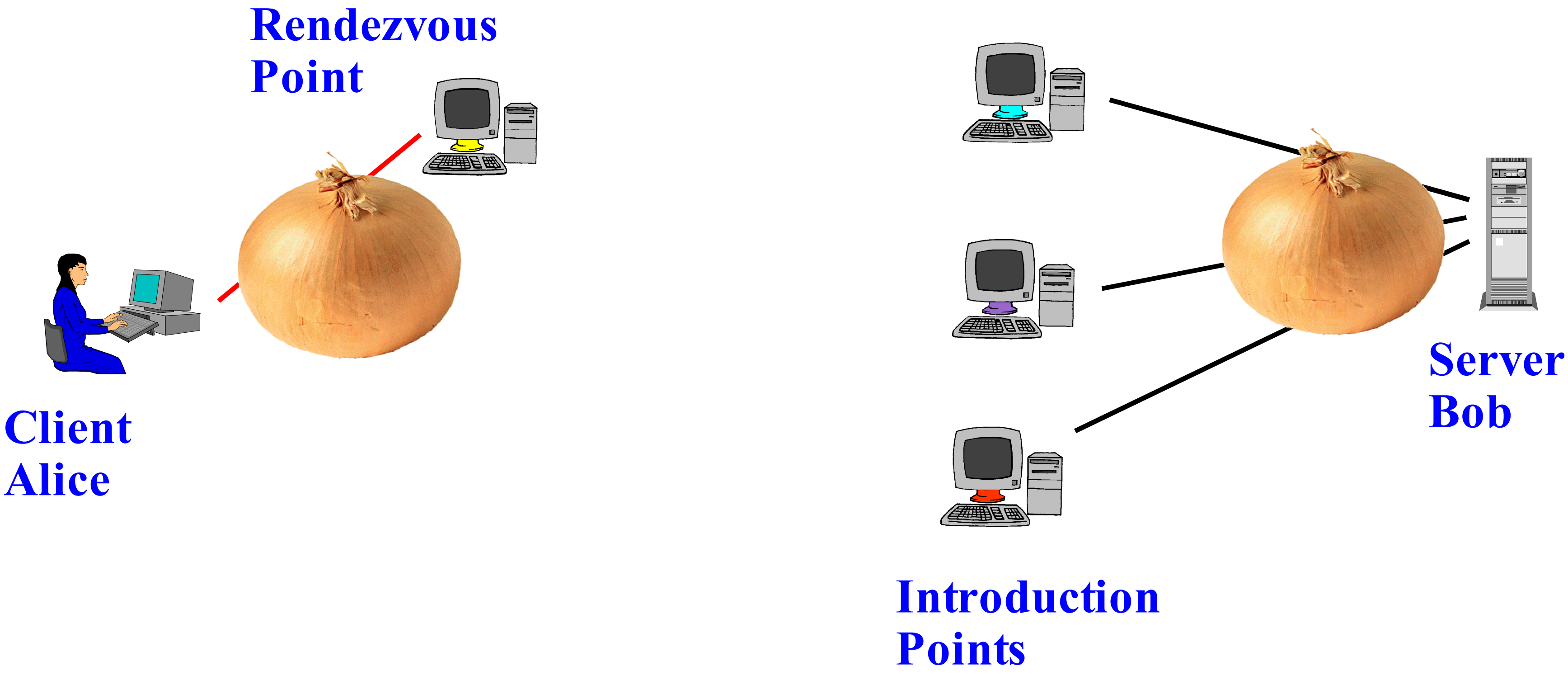
Location Hidden Servers

2'. Alice obtains Service Descriptor (including Intro Pt. address) at Lookup Server



Location Hidden Servers

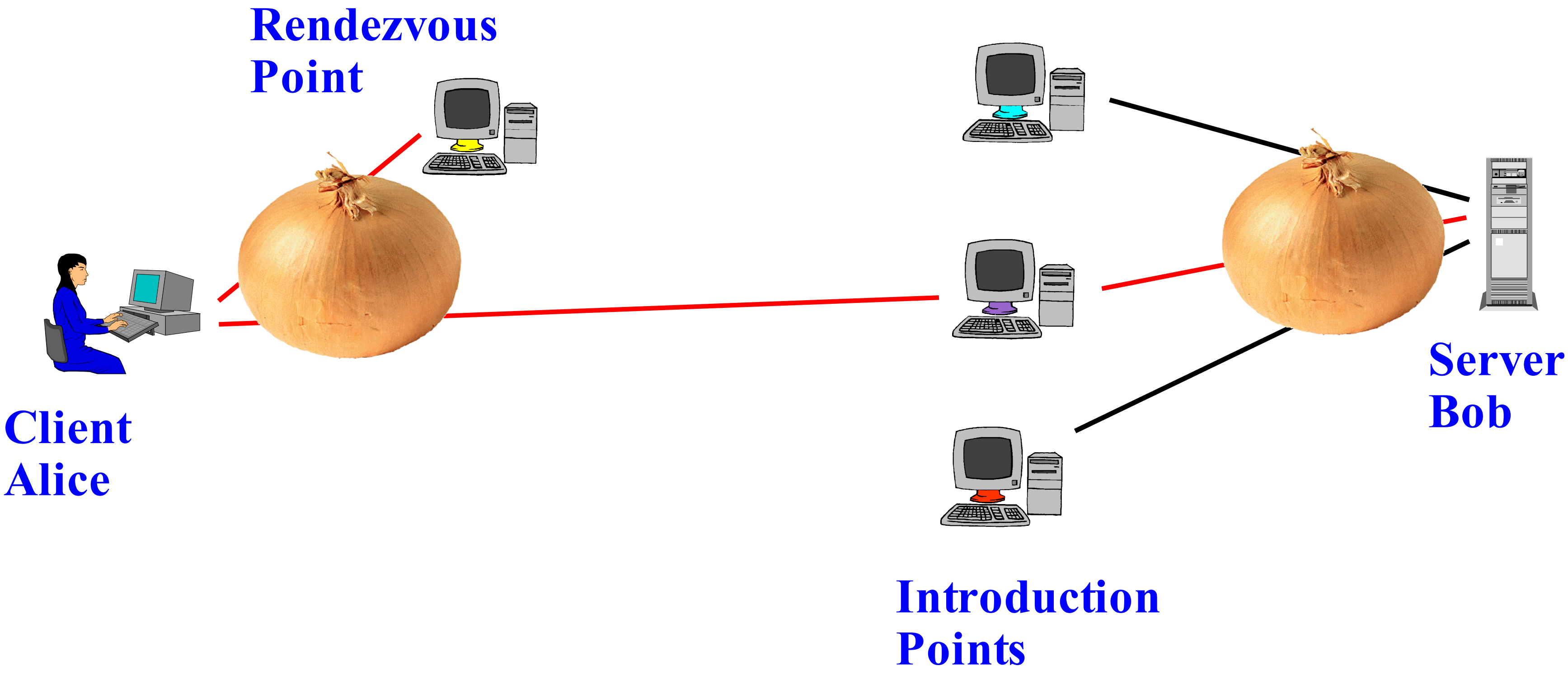
3. Client Alice creates onion route to Rendezvous Point (RP)



Location Hidden Servers

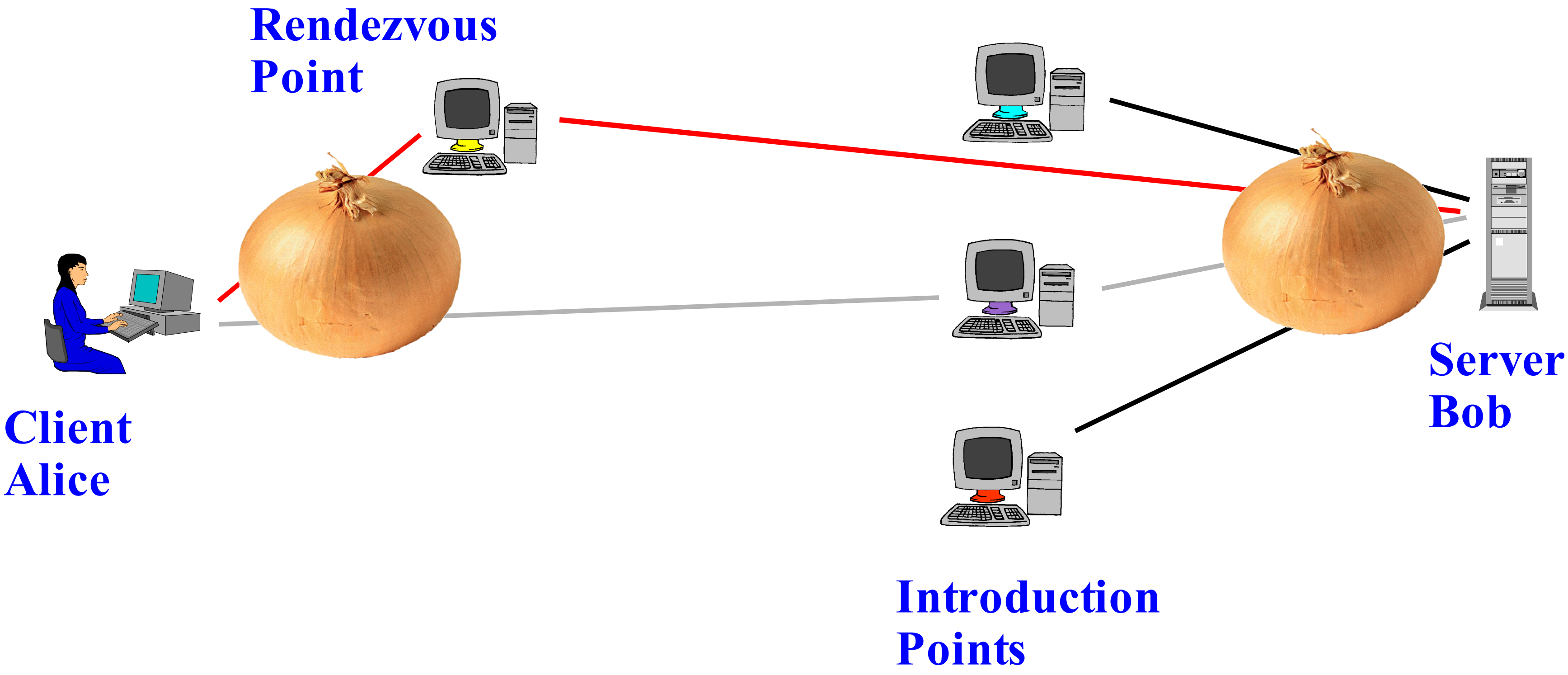
3. Client Alice creates onion route to Rendezvous Point (RP)

4. Alice sends RP addr. and any authorization through IP to Bob



Location Hidden Servers

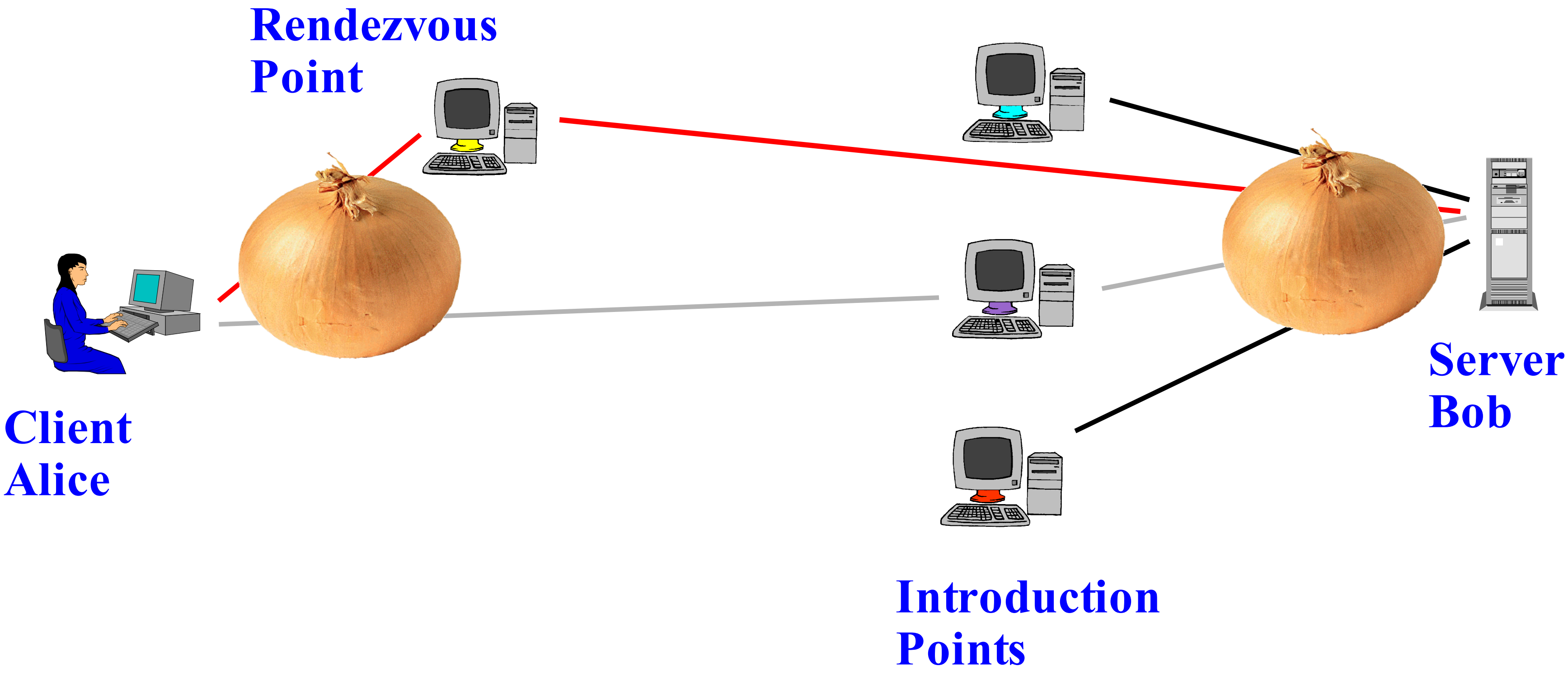
5. If Bob chooses to talk to Alice, connects to Rendezvous Point



Location Hidden Servers

5. If Bob chooses to talk to Alice, connects to Rendezvous Point

6. Rendezvous point mates the circuits from Alice and Bob



How do we compare Tor's security?

Assume the adversary owns c of the n nodes.

(he can choose which)

What's the chance for a random Alice talking to a random Bob that the adversary learns they are linked?

- Freedom, Tor: c^2/n^2 (10 of 100 => 1%)
- Peekabooty, six-four, freenet: c/n (10 of 100 => 10%)
- JAP: c^2/(n/2)^2 (10 of 100 => 4%)
- Anonymizer: 1 if c>0

Get the Code, Run a Node!
(or just surf the web anonymously)

- Current code freely available (free software license)
- Comes with a specification – the JAP team in Dresden implemented a compatible Tor client in Java
- Chosen as the anonymity layer for EU PRIME project
- Design paper, system spec, code, see the list of current nodes, etc.
- http://tor.eff.org/

Policy issues

- Attacks we've seen:
  - Ransom note via Hotmail
  - Spam via Google Groups
  - IRC jerks --> DDoS on Tor server
  - Vin Diesel movies
- Wikipedia, Slashdot
- SORBS / spam blacklists

Design Tradeoffs

- Low-latency (Tor) vs. high-latency (Mixminion)
- Packet-level vs stream-level capture
- Padding vs. no padding (mixing, traffic shaping)
- UI vs. no UI (Contest!)
- AS-level paths and proximity issues

Design Tradeoffs

- Enclave-level onion routers / proxies / helper nodes
- Path length? (3 hops, don't reuse nodes)
- China?
- P2P network vs. static network

Lessons?

- 1) Bad people don't need Tor. They're doing fine.
- 2) Honest people don't have Tor. They need it.
- 3) Law enforcement can benefit from it too.
- 4) Tor is not unbreakable.