Theory and Design of Low-latency Anonymity Systems (Lecture 2)

Paul Syverson
U.S. Naval Research Laboratory
syverson@itd.nrl.navy.mil
http://www.syverson.org

Course Outline

Lecture 1:

- Usage examples, basic notions of anonymity, types of anonymous comms systems
- Crowds: Probabilistic anonymity, predecessor attacks

Lecture 2:

- Onion routing basics: simple demo of using Tor, network discovery, circuit construction, crypto, node types and exit policies
- Economics, incentives, usability, network effects

Course Outline

Lecture 3:

- Formalization and analysis, possibilistic and probabilistic definitions of anonymity
- Hidden services: responder anonymity, predecessor attacks revisited, guard nodes

Lecture 4:

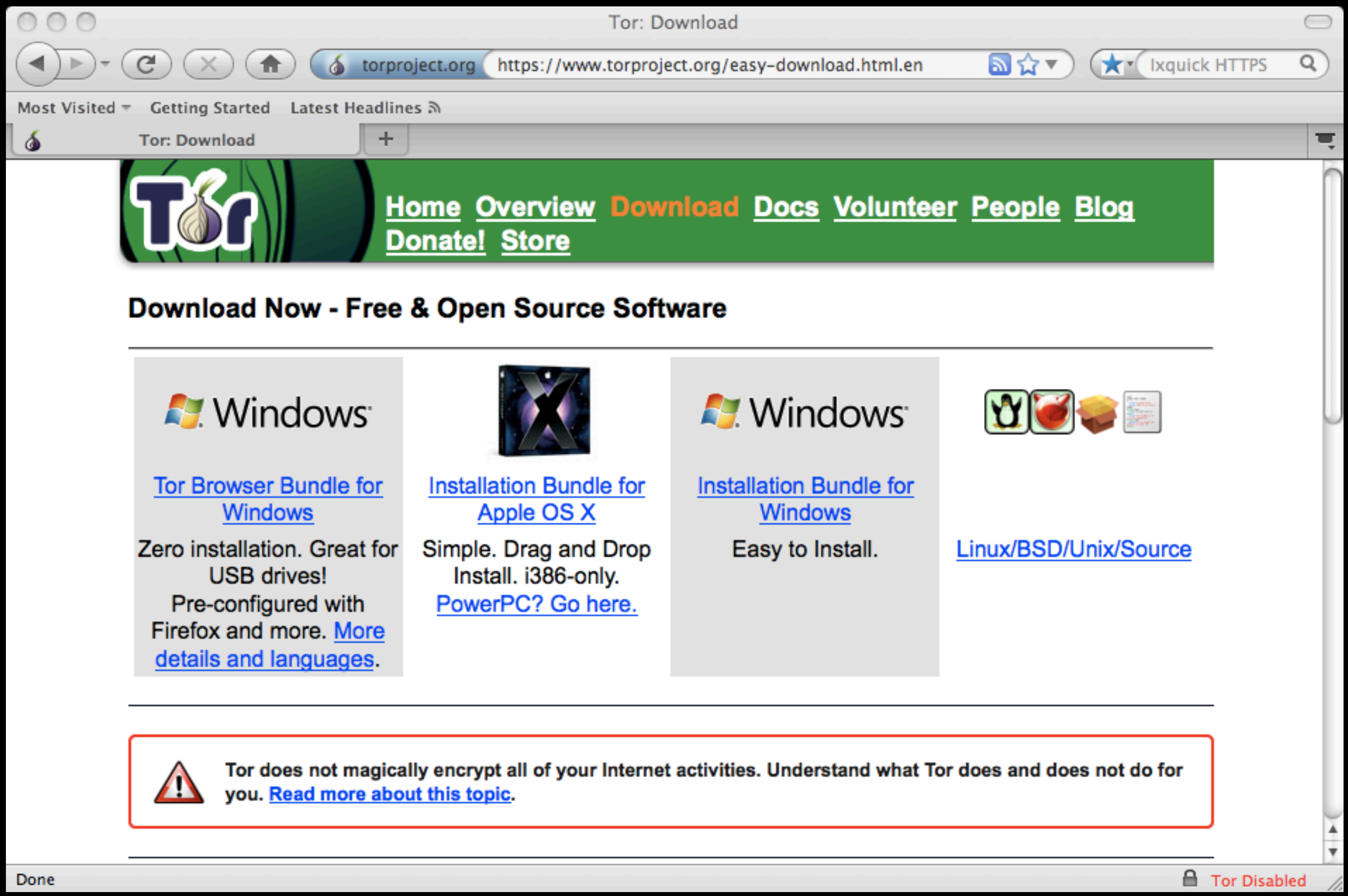
- Link attacks
- Trust

Tor Demo Background

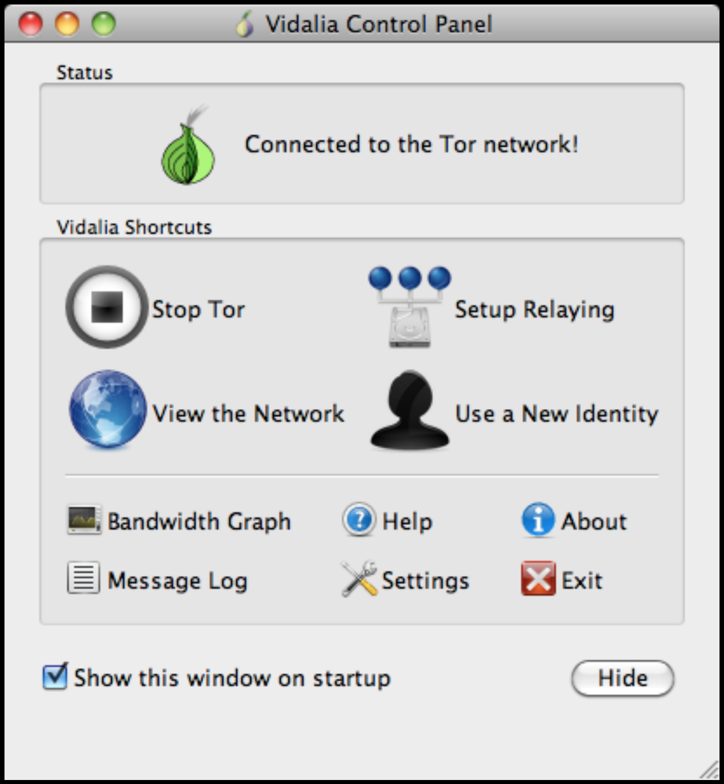
Tor is an onion routing system for anonymous communication

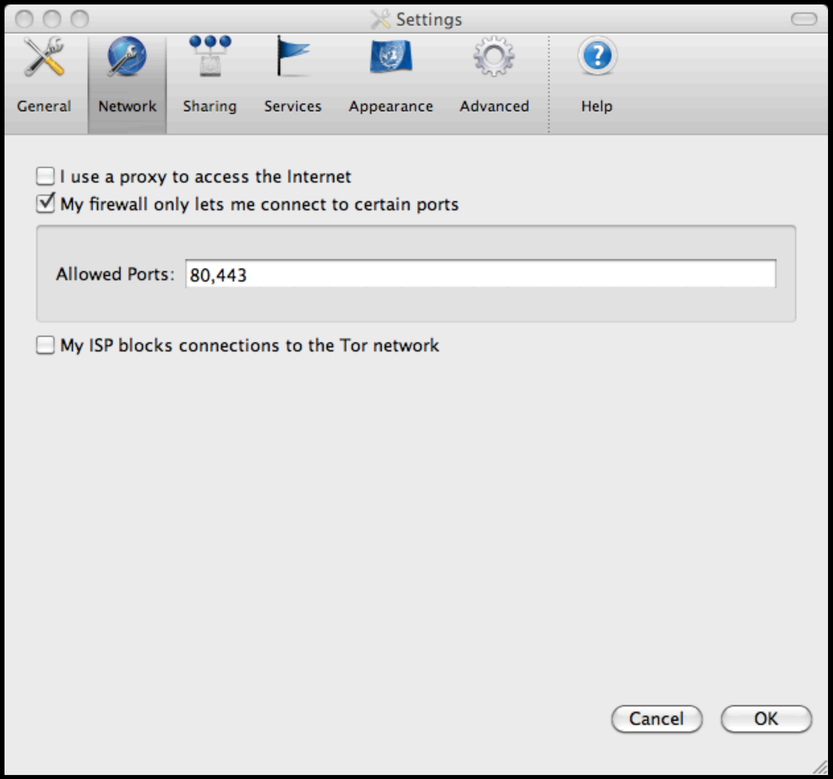
- Initially a project at the U.S. Naval Research Laboratory
- The Tor Project Inc. is now a U.S. nonprofit 501 (c) (3)
- Network comprised of thousands of volunteer nodes from around the world
- Free and open software maintained by the Tor Project, used by hundreds of thousands

Getting Tor

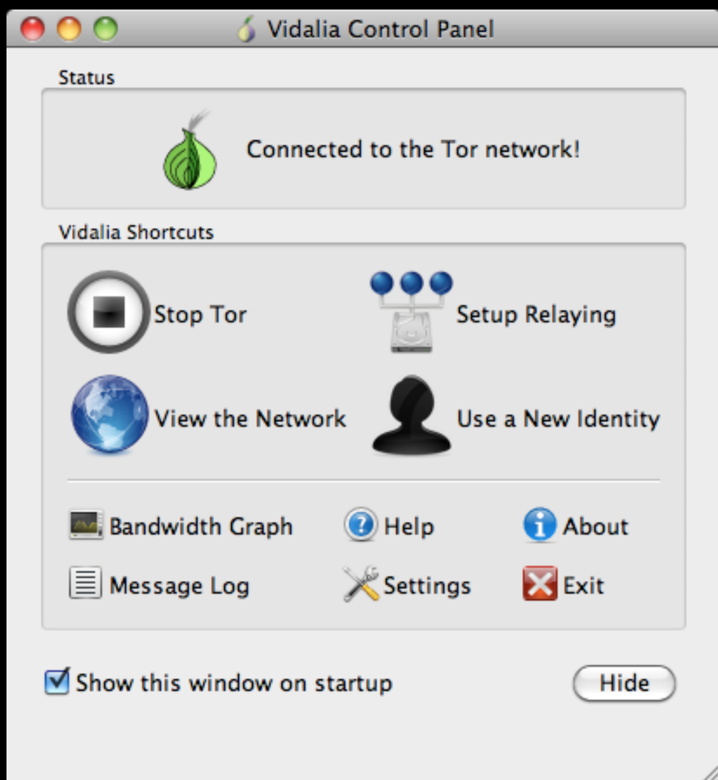

Tor: Download
torproject.org
https://www.torproject.org/easy-download.html.en
Ixquick HTTPS
Most Visited
Getting Started
Latest Headlines
Tor: Download
Home Overview Download Docs Volunteer People Blog
Donate! Store
Download Now - Free & Open Source Software
Windows
Tor Browser Bundle for Windows
Zero installation. Great for USB drives! Pre-configured with Firefox and more. More details and languages.
Installation Bundle for Apple OS X
Simple. Drag and Drop Install. i386-only. PowerPC? Go here.
Windows
Installation Bundle for Windows
Easy to Install.
Linux/BSD/Unix/Source
Tor does not magically encrypt all of your Internet activities. Understand what Tor does and does not do for you. Read more about this topic.
Done
Tor Disabled

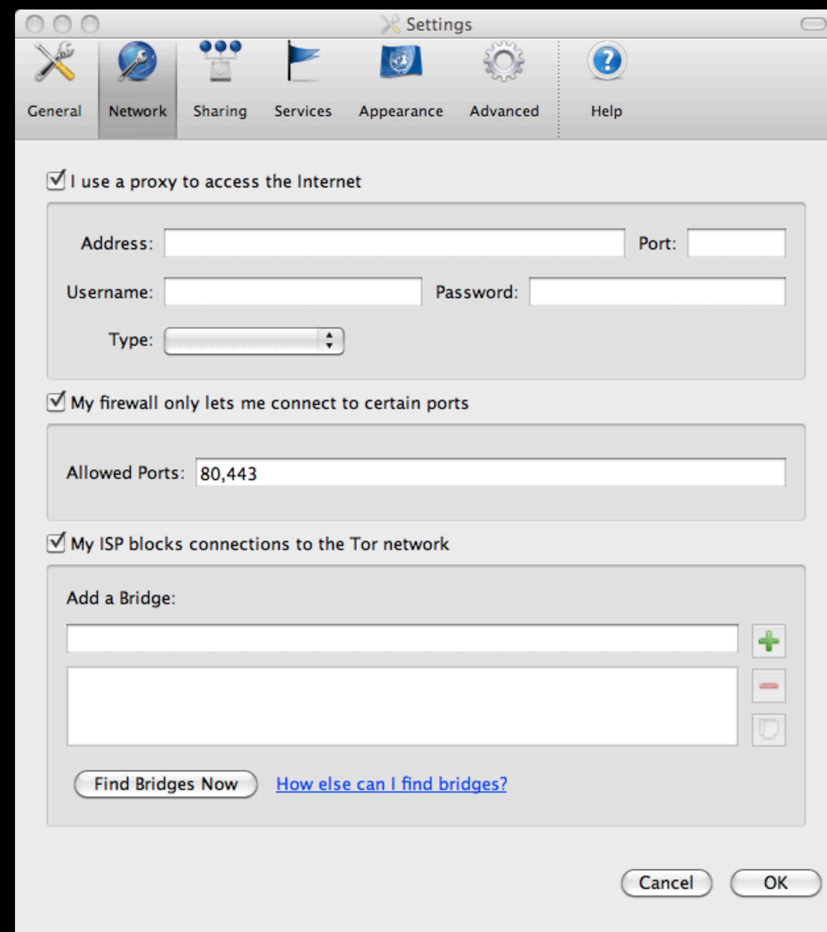
Vidalia: Tor's GUI

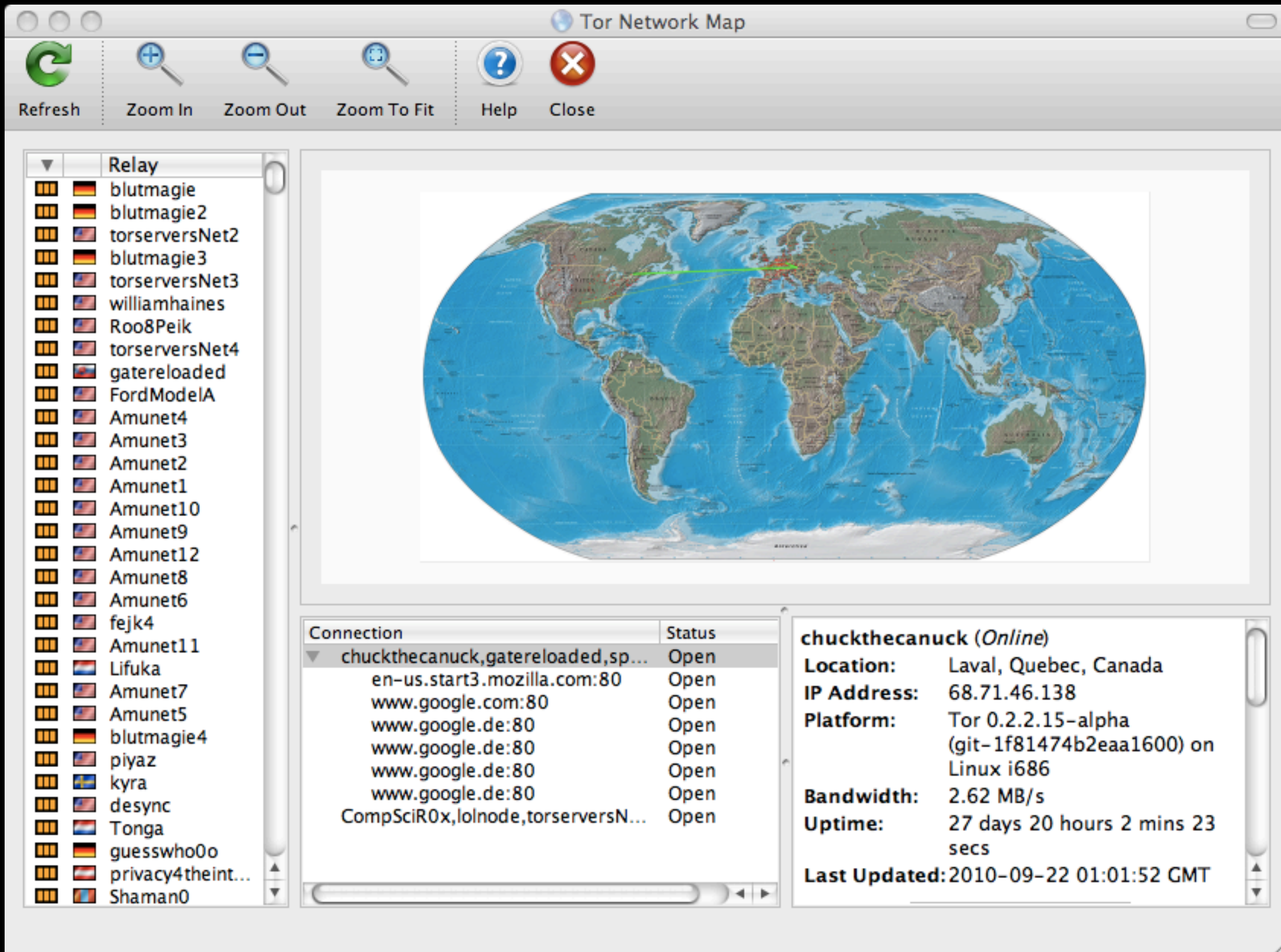




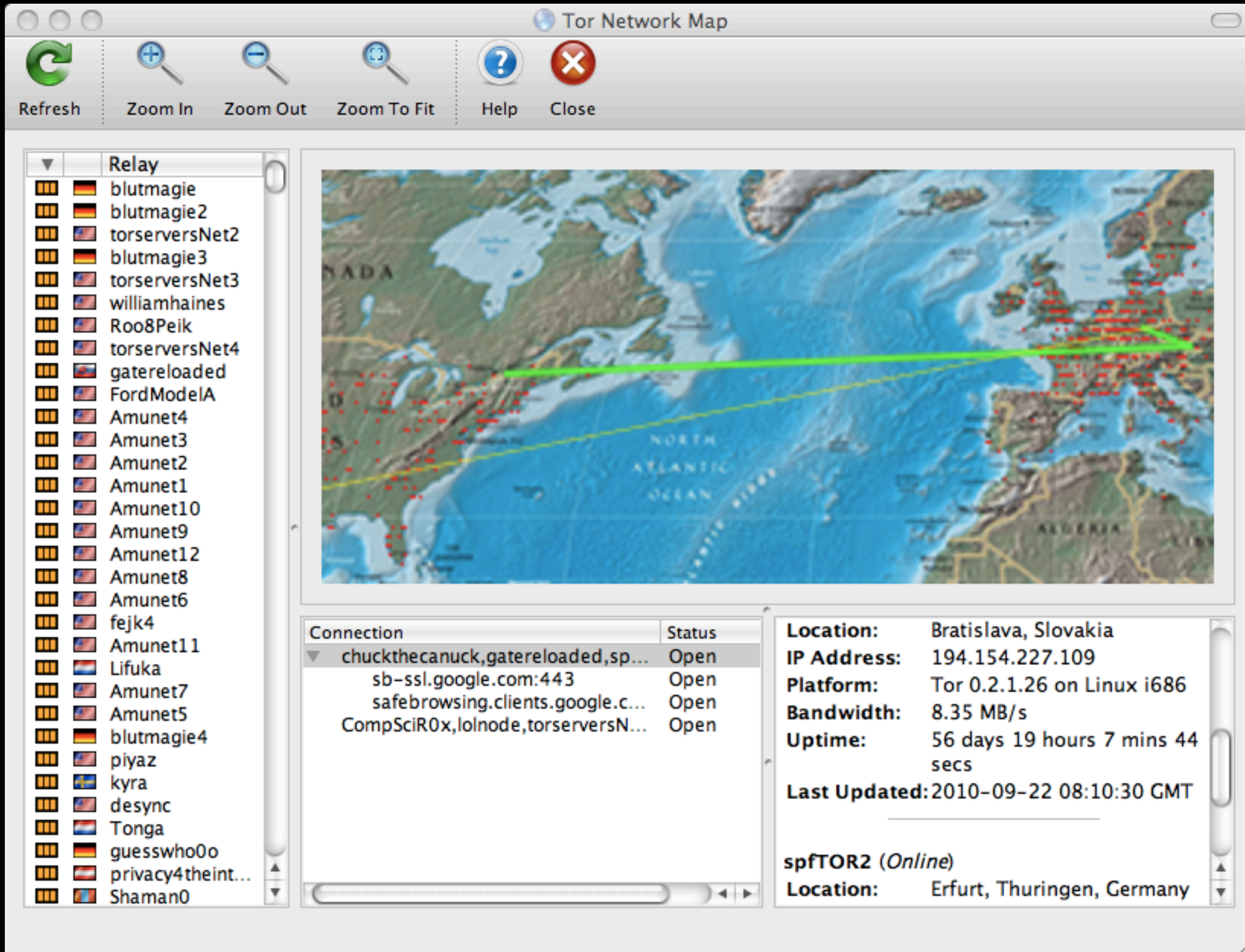
Vidalia: Tor's GUI



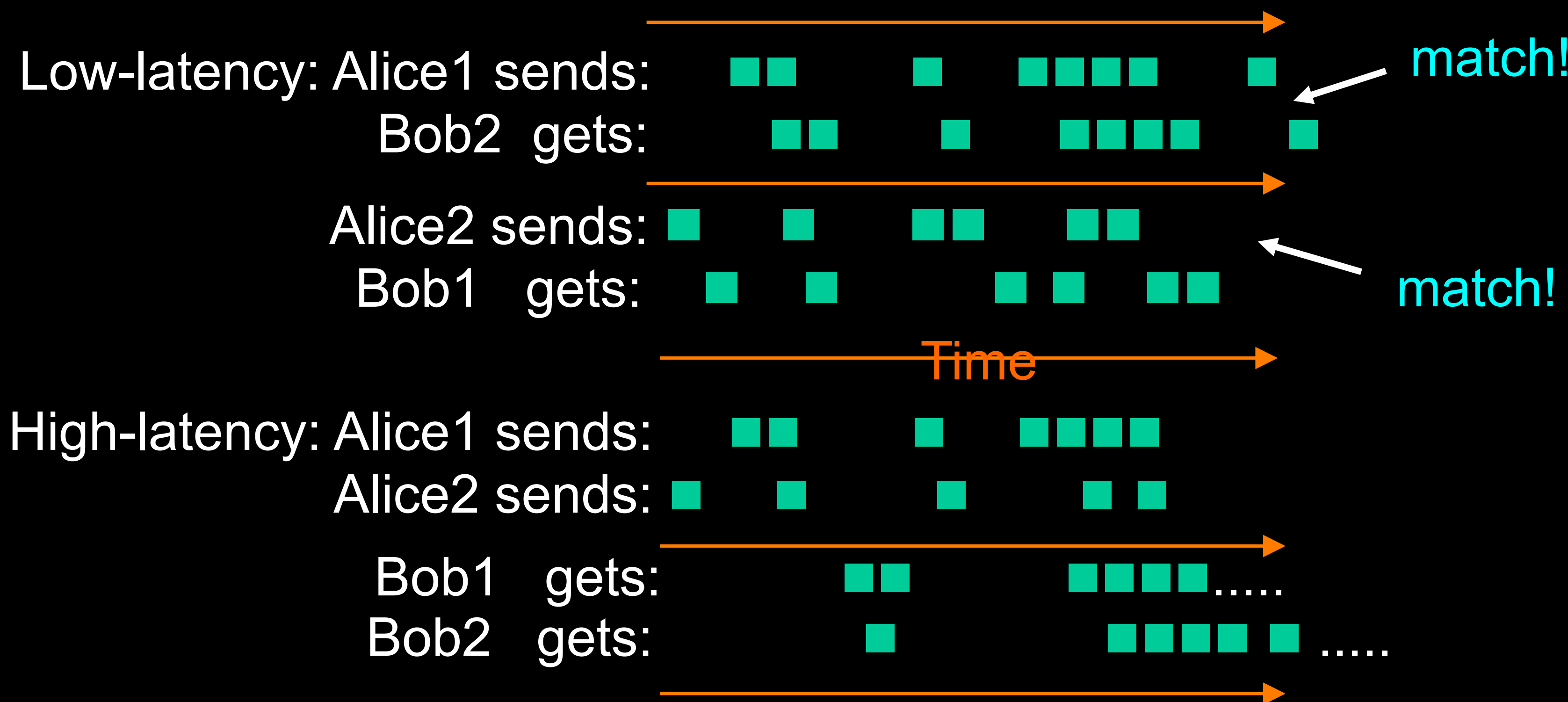



Tor Network Map
Refresh
Zoom In
Zoom Out
Zoom To Fit
Help
Close
Relay
blutmagie
blutmagie2
torserversNet2
blutmagie3
torserversNet3
williamhaines
Roo8Peik
torserversNet4
gatereloaded
FordModelA
Amunet4
Amunet3
Amunet2
Amunet1
Amunet10
Amunet9
Amunet12
Amunet8
Amunet6
fejk4
Amunet11
Lifuka
Amunet7
Amunet5
blutmagie4
piyaz
kyra
desync
Tonga
guesswho0o
privacy4theint...
Shaman0
Connection
Status
chuckthecanuck,gatereloaded,sp...
Open
en-us.start3.mozilla.com:80
Open
www.google.com:80
Open
www.google.de:80
Open
www.google.de:80
Open
www.google.de:80
Open
www.google.de:80
Open
CompSciR0x,lolnode,torserversN...
Open
chuckthecanuck (Online)
Location: Laval, Quebec, Canada
IP Address: 68.71.46.138
Platform: Tor 0.2.2.15-alpha (git-1f81474b2eaa1600) on Linux i686
Bandwidth: 2.62 MB/s
Uptime: 27 days 20 hours 2 mins 23 secs
Last Updated: 2010-09-22 01:01:52 GMT

The Web through Tor and TorButton

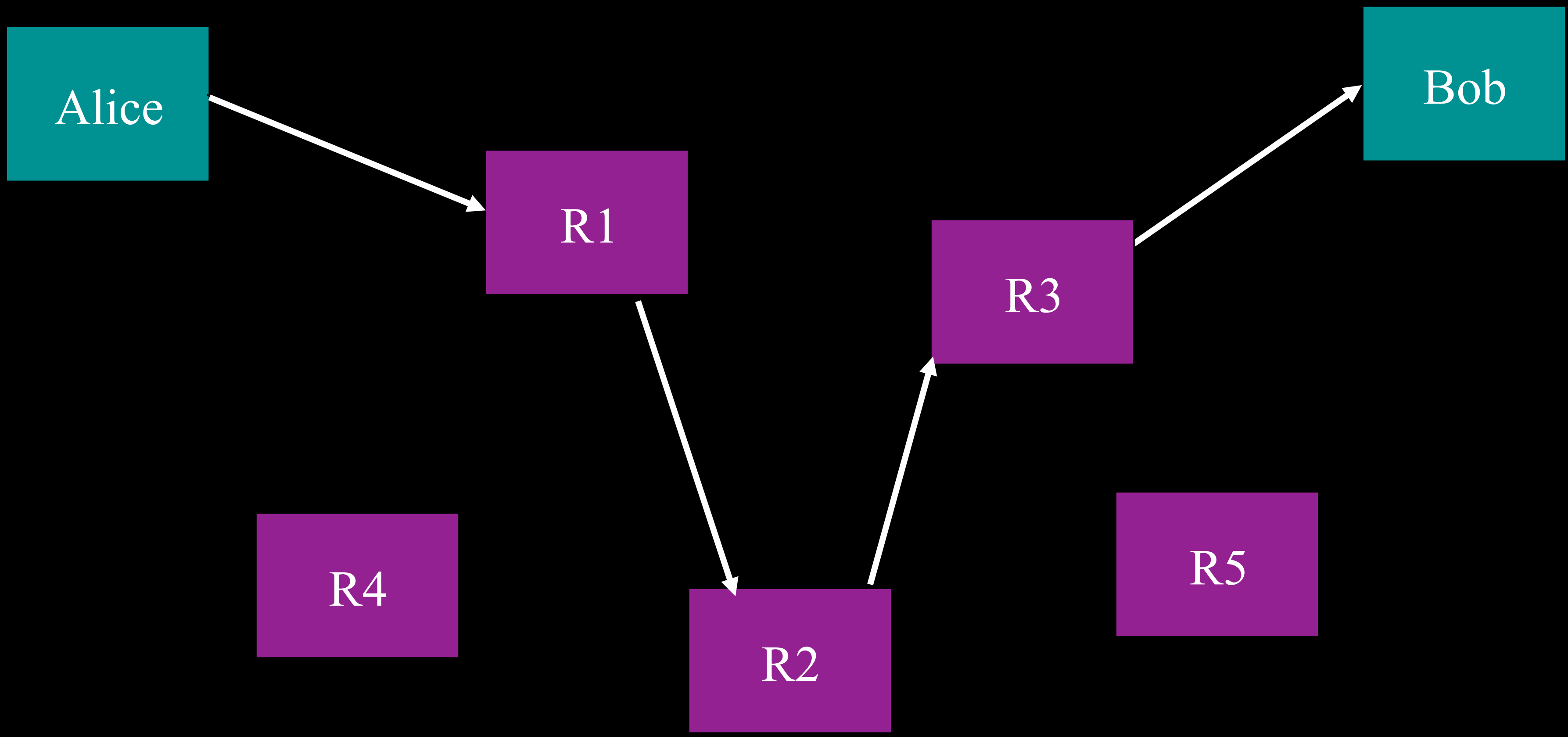

Tor Network Map
Refresh
Zoom In
Zoom Out
Zoom To Fit
Help
Close
Relay
blutmagie
blutmagie2
torserversNet2
blutmagie3
torserversNet3
williamhaines
Roo8Peik
torserversNet4
gatereloaded
FordModelA
Amunet4
Amunet3
Amunet2
Amunet1
Amunet10
Amunet9
Amunet12
Amunet8
Amunet6
fejk4
Amunet11
Lifuka
Amunet7
Amunet5
blutmagie4
piyaz
kyra
desync
Tonga
guesswho0o
privacy4theint...
Shaman0
NORTH
ATLANTIC
OCEAN
Connection
Status
chuckthecanuck,gatereloaded,sp... Open
sb-ssl.google.com:443 Open
safebrowsing.clients.google.c... Open
CompSciR0x,lolnode,torserversN... Open
Location: Bratislava, Slovakia
IP Address: 194.154.227.109
Platform: Tor 0.2.1.26 on Linux i686
Bandwidth: 8.35 MB/s
Uptime: 56 days 19 hours 7 mins 44 secs
Last Updated: 2010-09-22 08:10:30 GMT
spfTOR2 (Online)
Location: Erfurt, Thuringen, Germany

Low-latency systems are vulnerable to end-to-end correlation attacks.

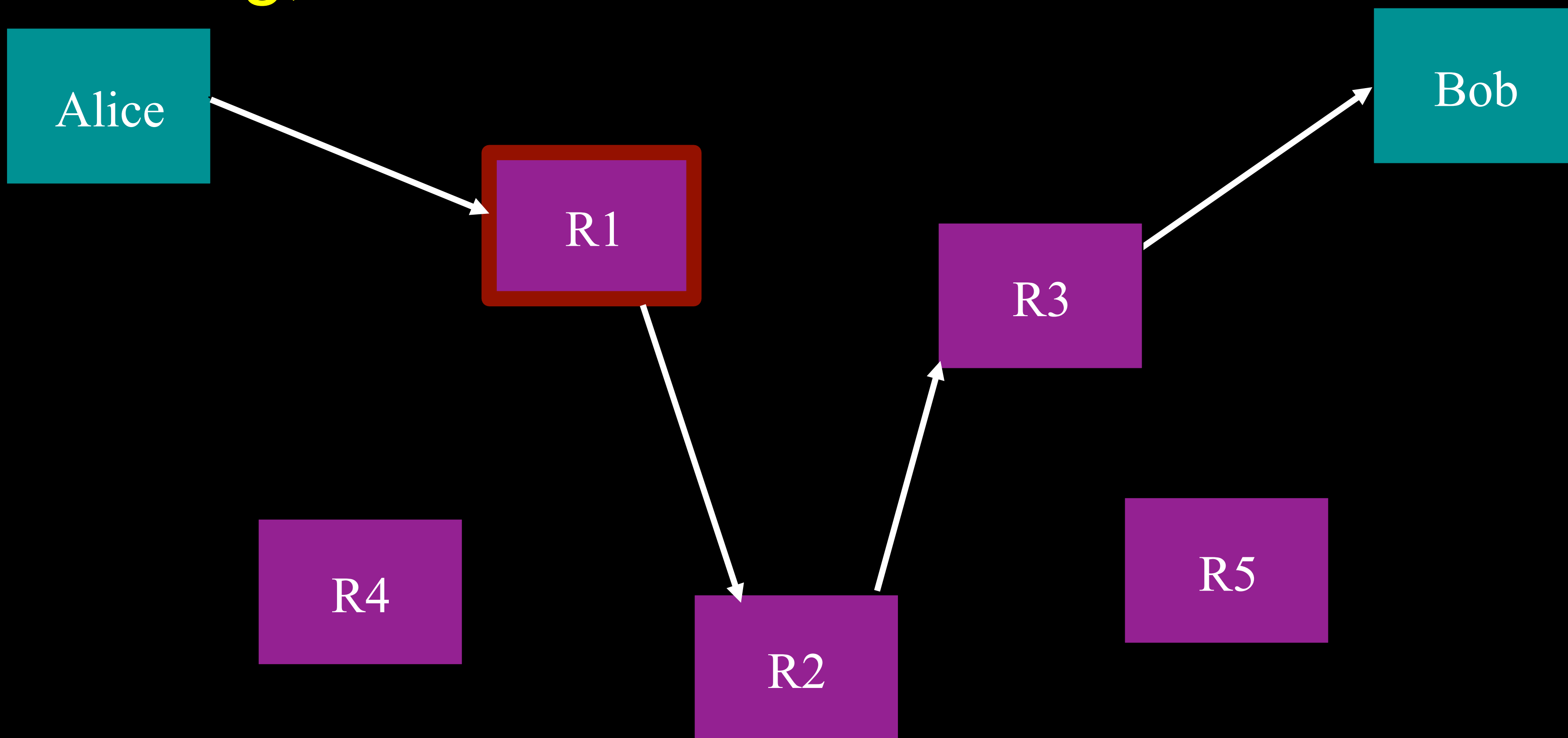


These attacks work in practice. The obvious defenses are expensive (like high-latency), useless, or both.

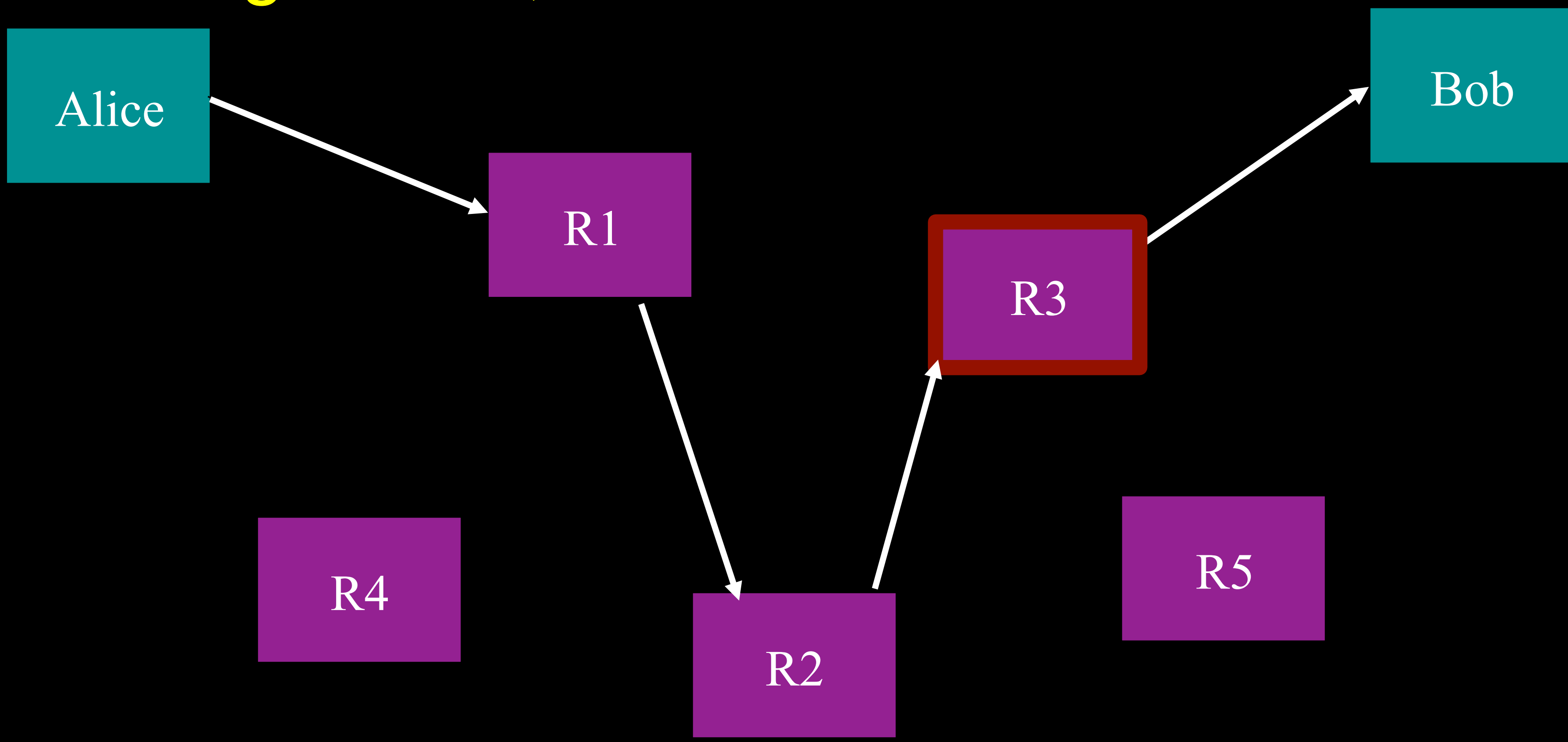
Multiple relays so that no single one can betray Alice.



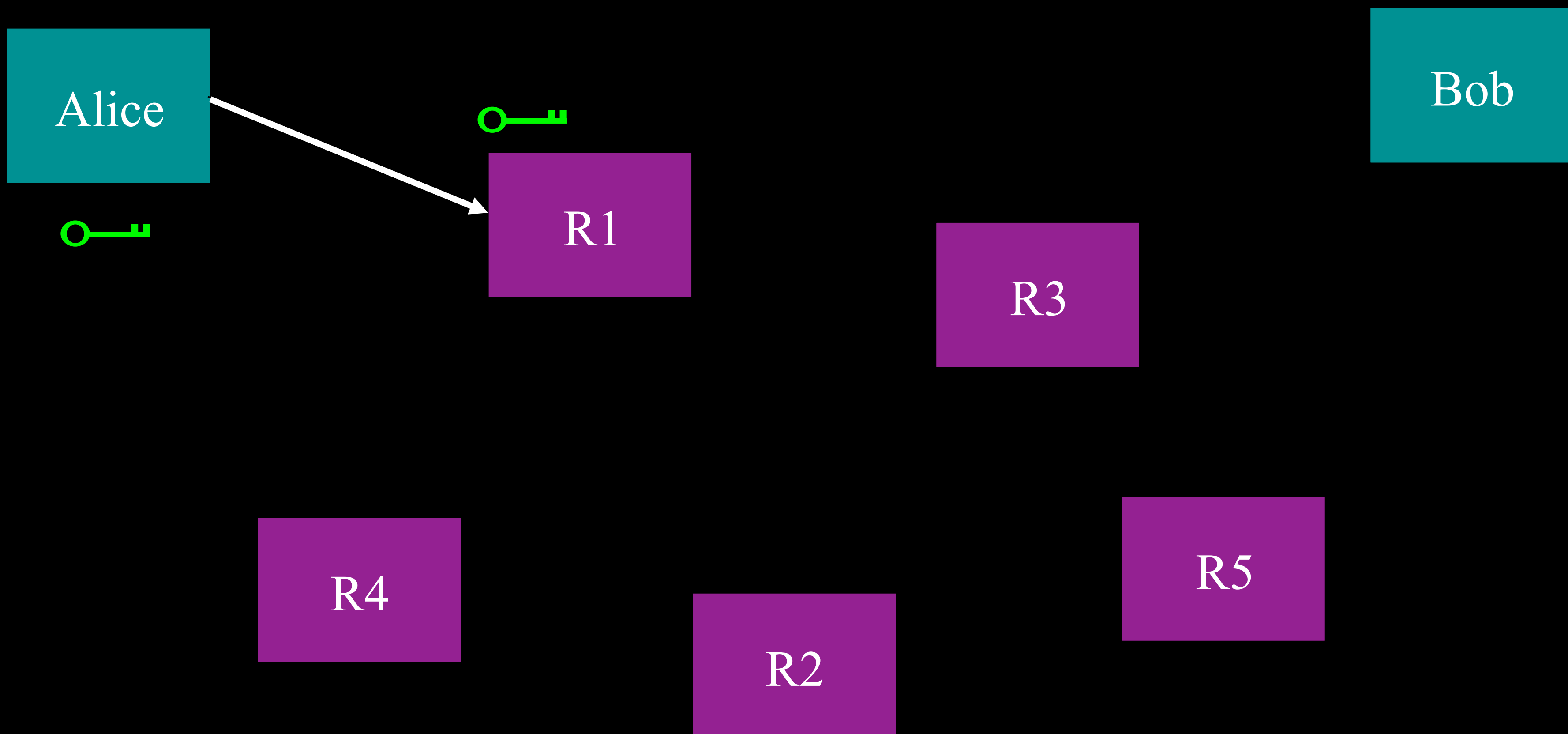
For Onion Routing:
A corrupt first hop can tell that Alice is talking, but not to whom.



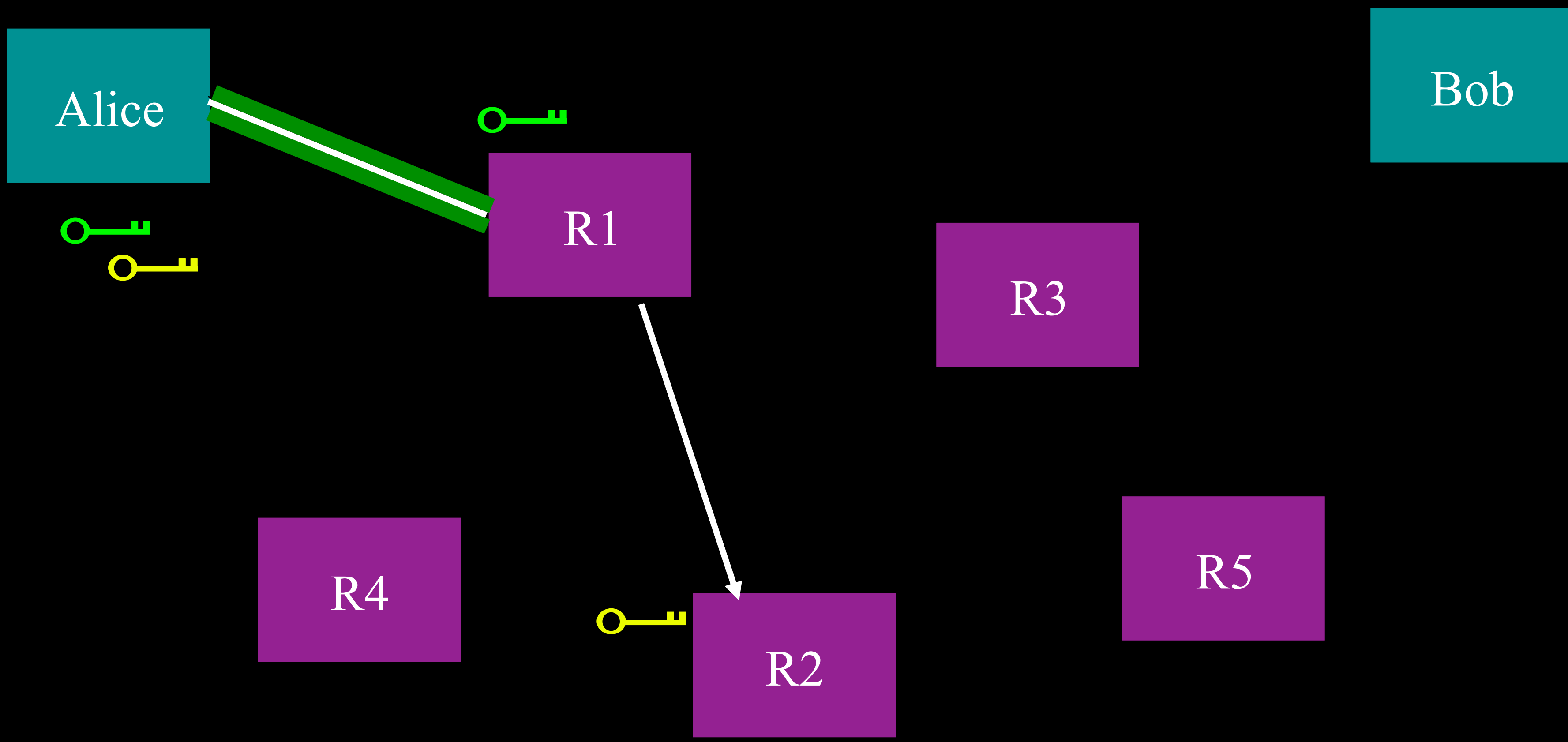
For Onion Routing:
A corrupt last hop can tell someone is talking to Bob, but not who.



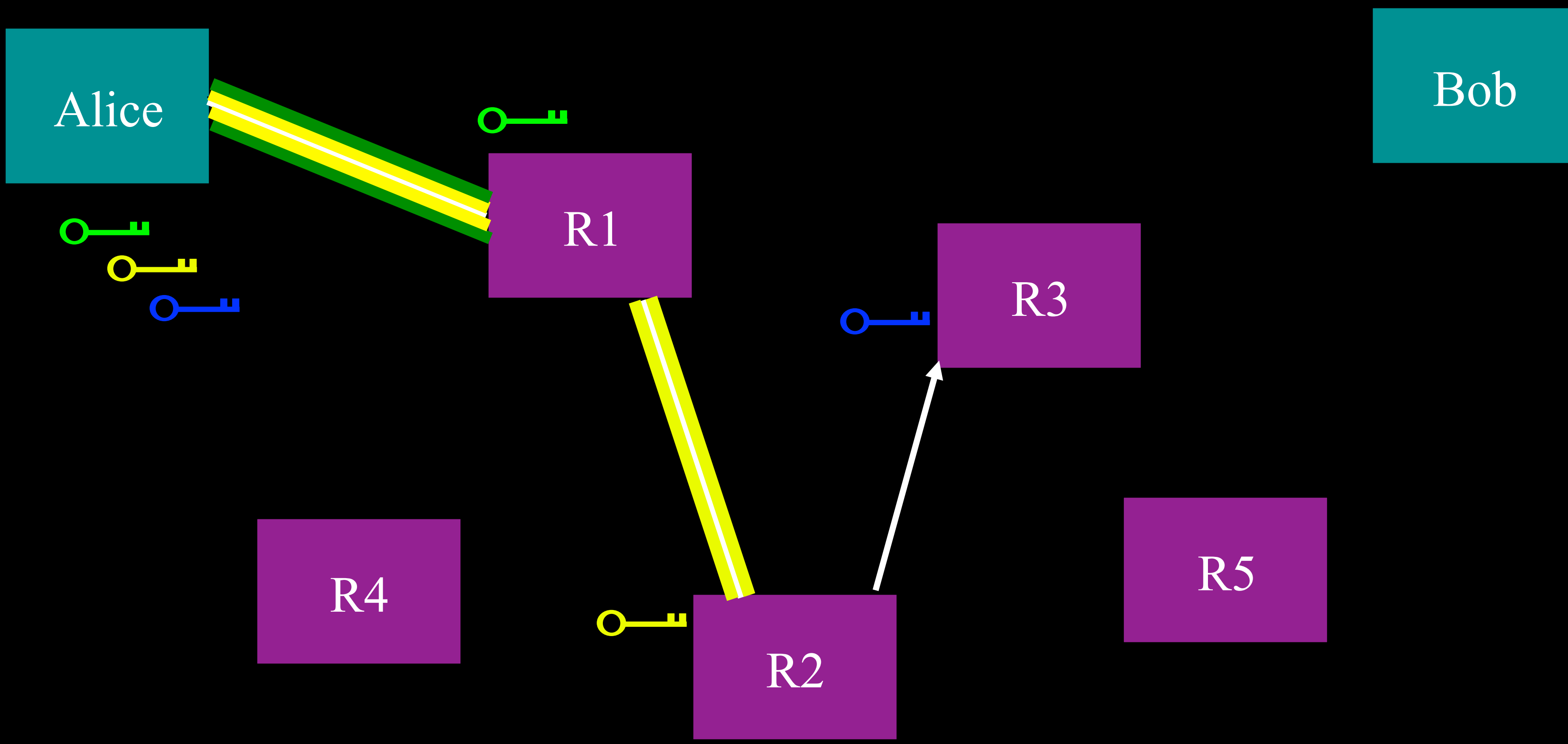
How onion routing works: Alice makes a session key with R1



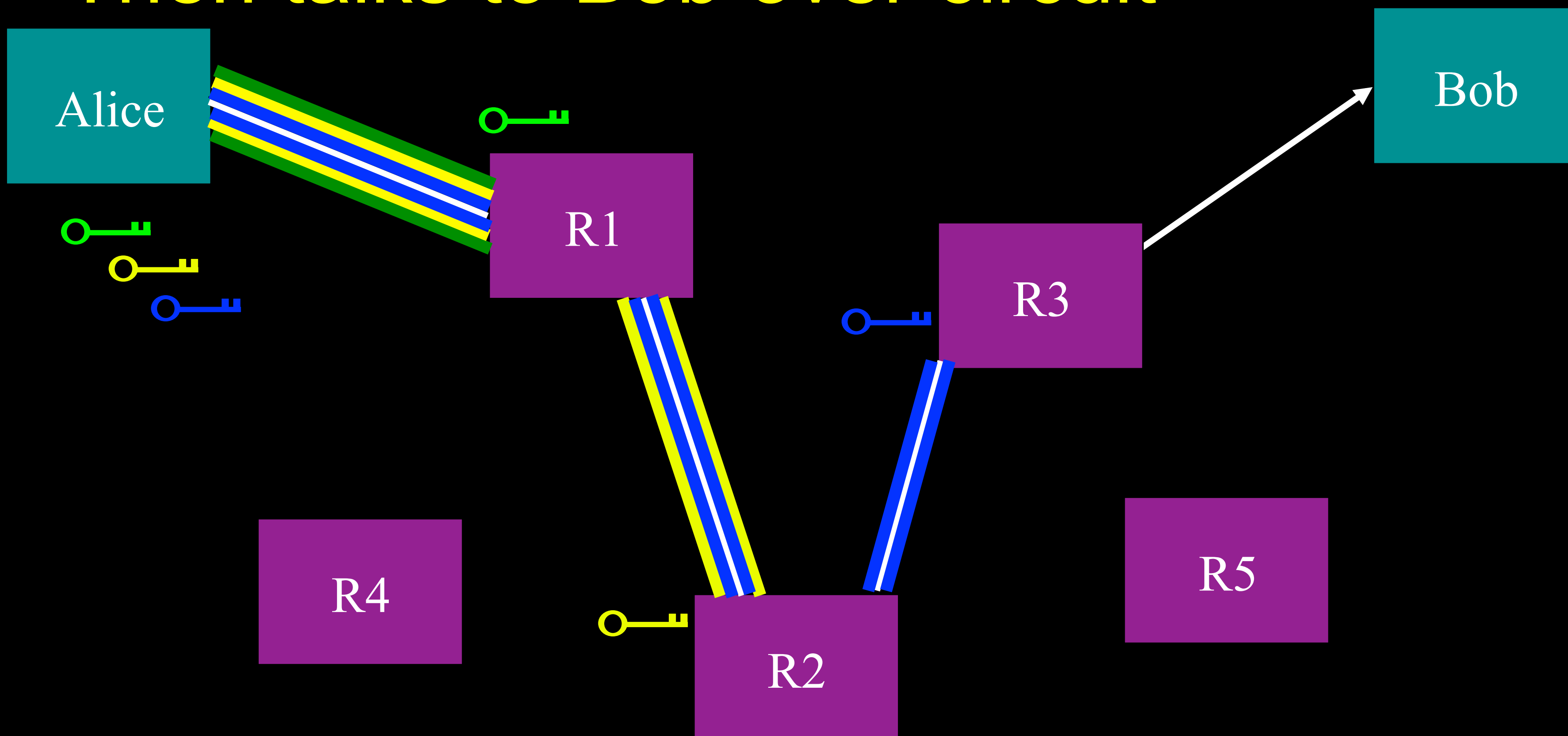
Alice makes a session key with R1 ...And then tunnels to R2



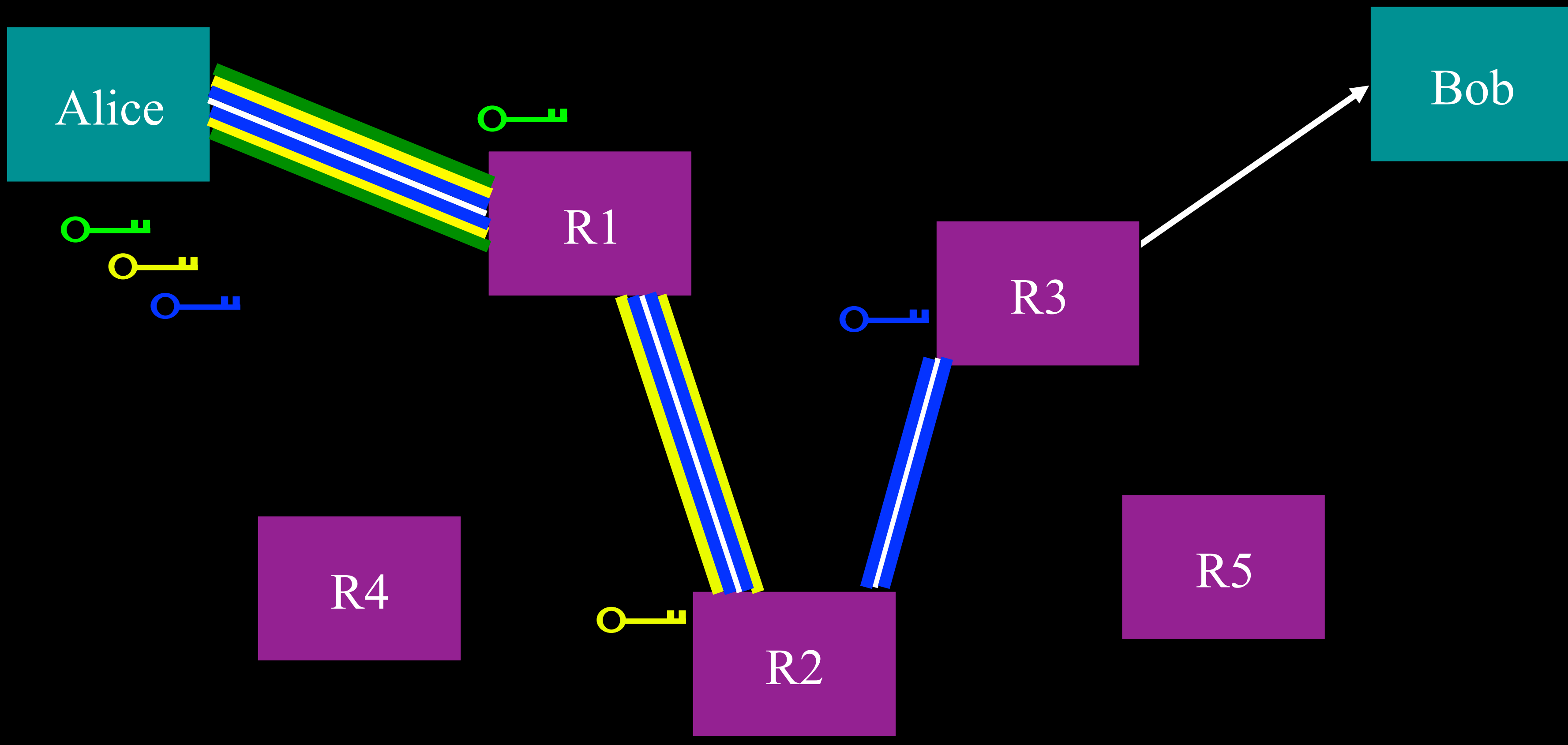
Alice makes a session key with R1 ...And then tunnels to R2...and to R3



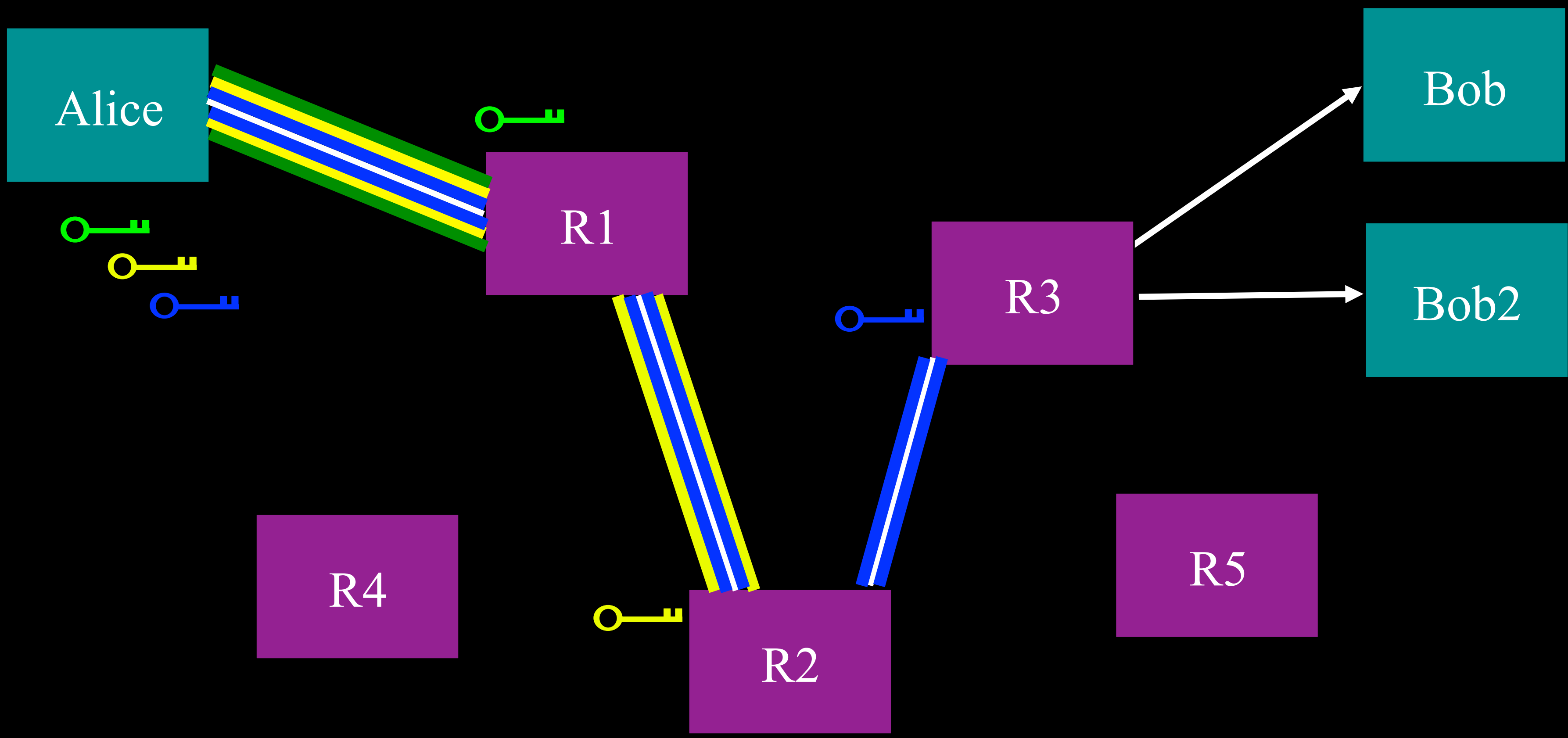
Alice makes a session key with R1
...And then tunnels to R2...and to R3
Then talks to Bob over circuit



Feasible because onion routing uses (expensive) public-key crypto just to build circuits, then uses (cheaper) symmetric-key crypto to pass data



Can multiplex many connections through the encrypted circuit



That's Tor* in a nutshell

* Tor's Onion Routing

What onion routing is not: Crowds

Public-key based circuit building means

- Forward security
- Better practical scalability
- Less centralized trust

Multiply encrypted circuits means

- less risk of route capture
- smaller profiling threat (also from shorter circuit duration)
- security not dependent on hiding path position
- able to support multiple applications/application encryption options

What onion routing is NOT: Mixes

Entirely different threat model

- mixes are based on an adversary not being able to correlate inputs and outputs he sees
- onion routing is based on an adversary not being able to see both inputs and outputs to correlate

Entirely different communications paradigm: Circuit based encryption vs. per message

- onion routing supports bidirectional communication
- onion routing supports low-latency communication

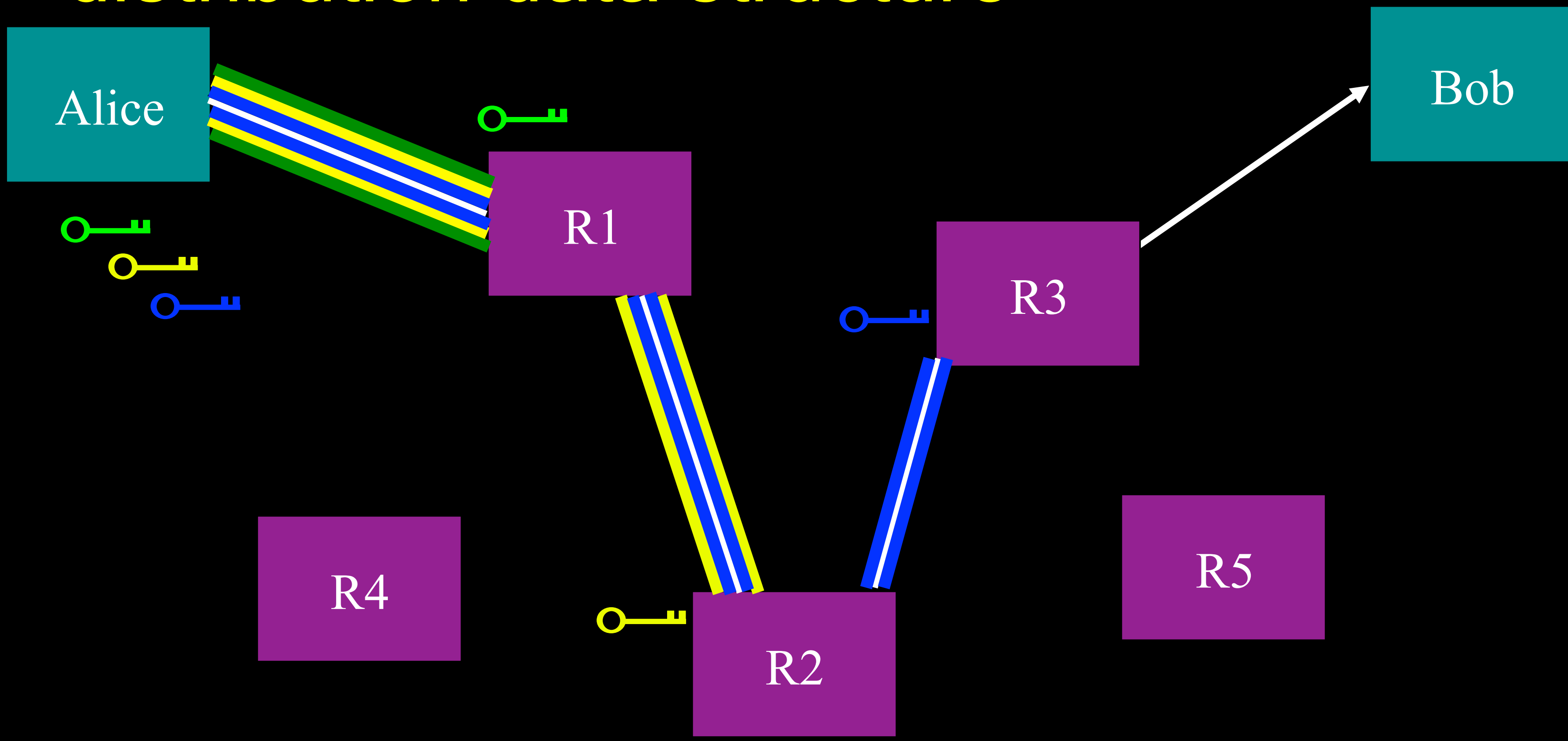
Can be combined to make mixing onion routers, but not typically done or desired

What onion routing is

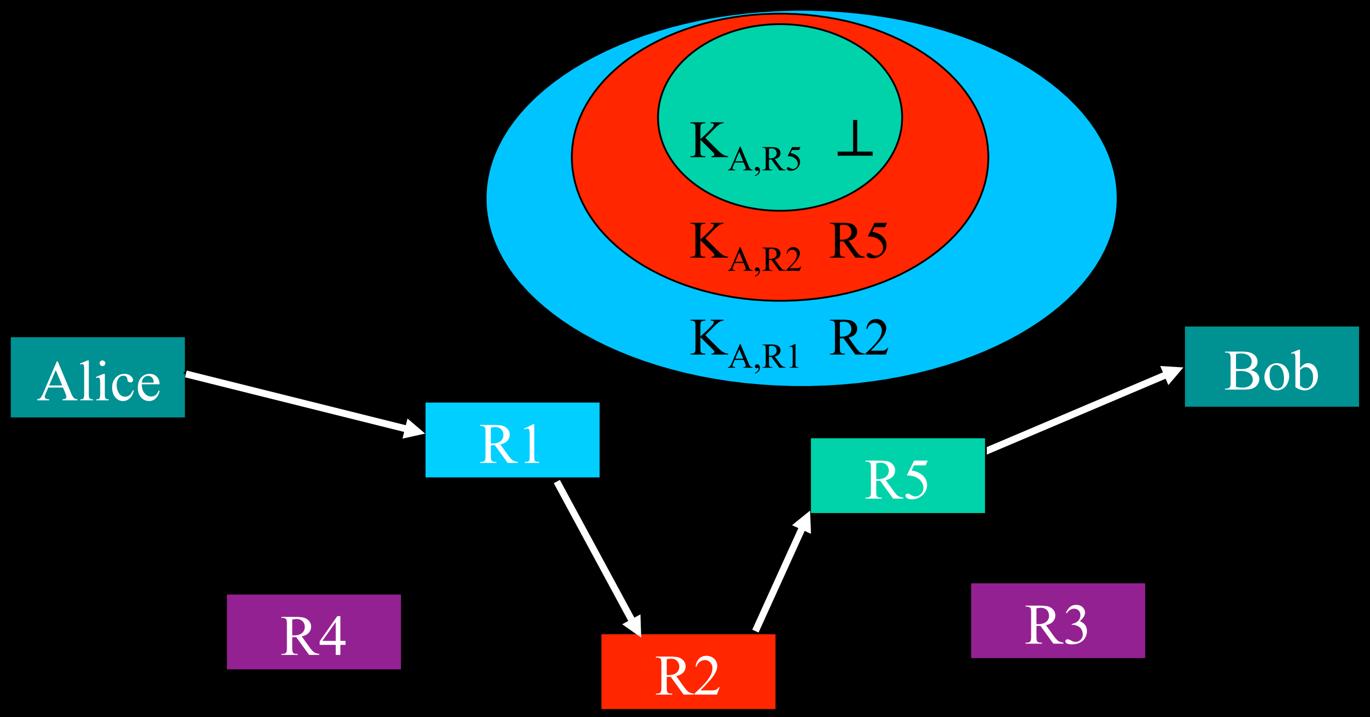
Uses expensive crypto (public-key) to lay a cryptographic circuit over which data is passed

Typically uses free-route circuit building to make location of circuit endpoints unpredictable

Why call it "onion routing"? Answer: Because of the original key distribution data structure



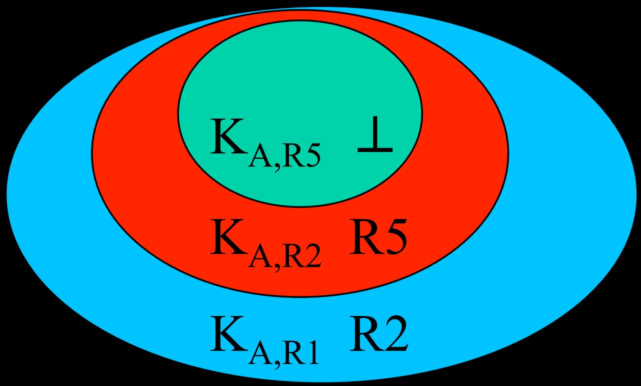
Why is it called onion routing?



Onion: Just layers of public-key crypto

- Nothing in the center, just another layer

Circuit setup



NRL v0 and v1 onion routing and also ZKS Freedom network used onions to build circuits

- Lacked Forward Secrecy
- Required storing record of onions against replay

Tor (NRL v2) uses one layer “onion skins”

- ephemeral Diffie-Hellman yields forward secrecy
- No need to record processed onions against replay
- From suggestion out of Zack Brown’s Cebolla

Aside: Why is it called 'Tor' and what does 'Tor' mean?

Frequent question to Roger c. 2001-2: Oh you're working on onion routing... which one?

Roger: *THE* onion routing. The original onion routing project from NRL.

Rachel: That's a good acronym.

Roger: And it's a good recursive acronym.

Plus, as a word, it has a good meaning in German (door/gate/portal) and Turkish (fine-meshed net)

Aside: Why is it called 'Tor' and what does 'Tor' mean?

We foolishly called the first Tor paper "Tor: the second generation onion router"

But this was very confusing

- 'Tor' stands for "The onion routing" or "Tor's onion routing". It does not stand for "the onion router"
- The paper is about the whole system, not just the onion routers
- Tor is not the second generation

Onion routing origins: Generation 0

Fixed-length five-node circuits

Integrated configuration

Static topology

Loose-source routing

★Partial active adversary

Rendezvous servers and reply onions

Onion routing, the next generation

- ★ Running a client separated from running an OR
- Variable length circuits (up to 11 hops per onion---or tunnel for more)
- Application independent proxies (SOCKS) plus redirector
- ★ Entry policies and exit policies
- Dynamic network state, flat distribution of state info
- Multiplexing of multiple application connections in single onion routing circuit
- Mixing of cells from different circuits
- Padding and bandwidth limiting

Third-generation onion routing (Tor)

★ Onion skins, not onions: Diffie-Hellman based circuit building

Fixed-length three-hop circuits

Rendezvous circuits and hidden servers

Directory servers, caching (evolved w/in Tor)

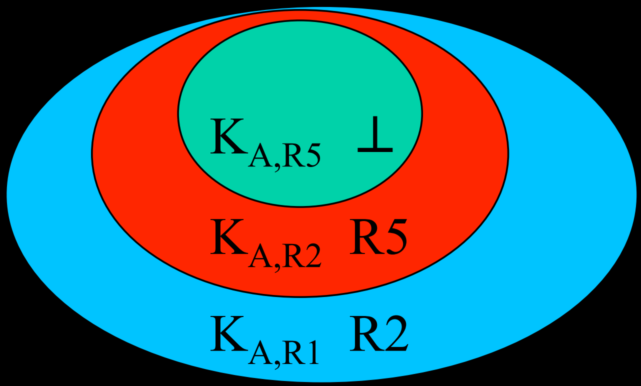
Most application specific proxies no longer needed (still need e.g. for DNS)

Congestion control

End-to-end integrity checking

No mixing and no padding

Circuit setup



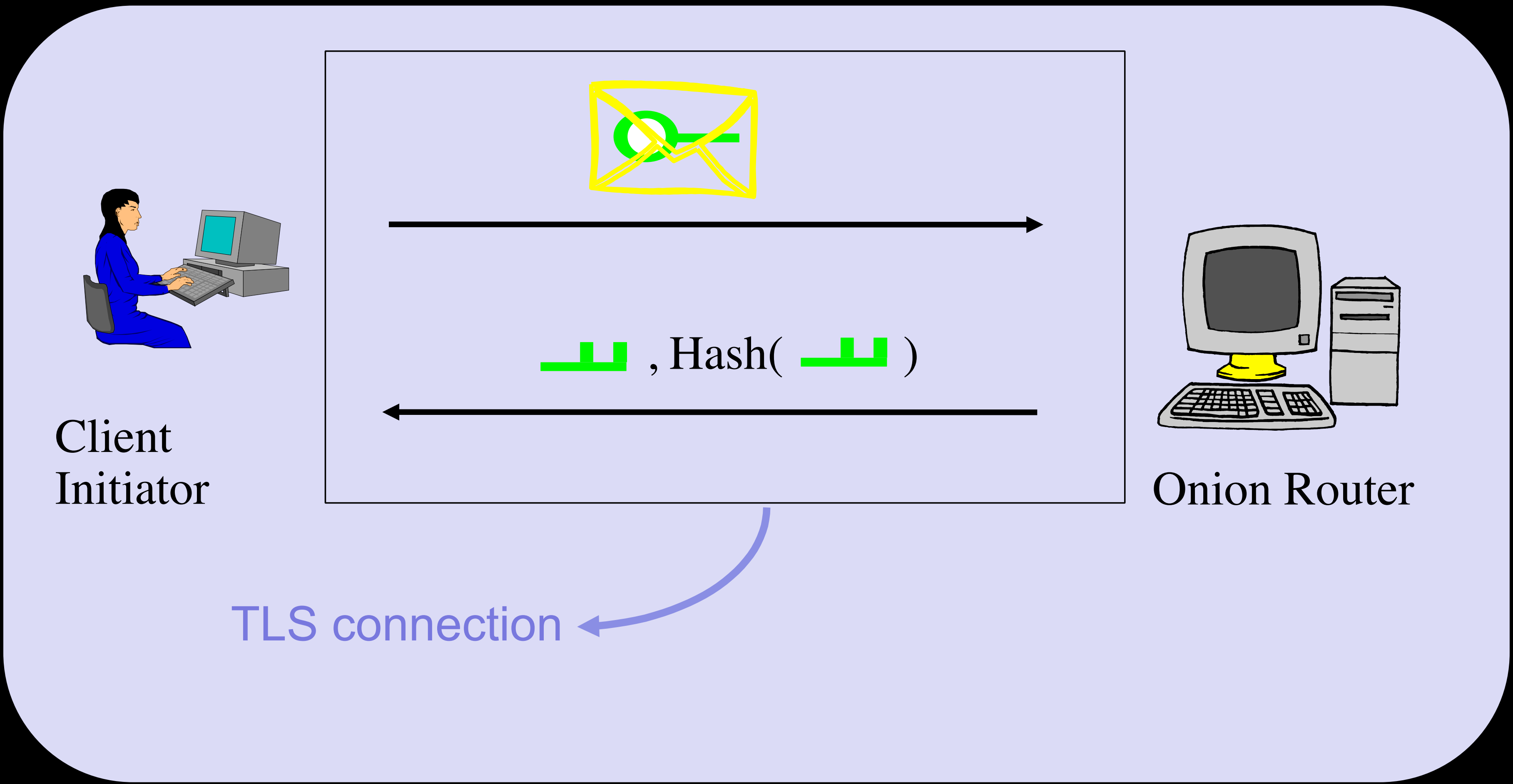
NRL v0 and v1 onion routing and also ZKS Freedom network used onions to build circuits

- Lacked Forward Secrecy
- Required storing record of onions against replay

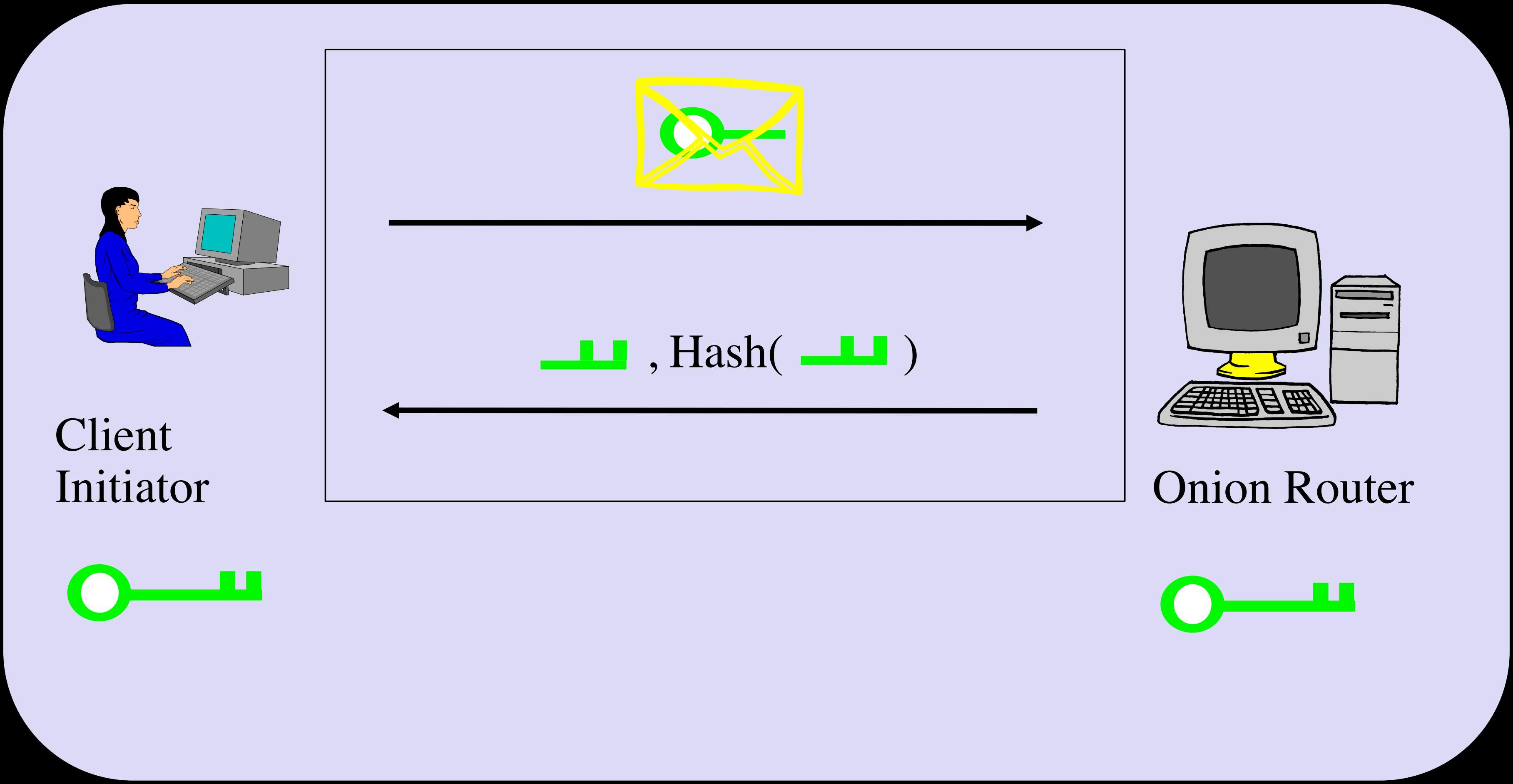
Tor (NRL v2) uses one layer “onion skins”

- ephemeral Diffie-Hellman yields forward secrecy
- No need to record processed onions against replay
- From suggestion out of Zack Brown’s Cebolla

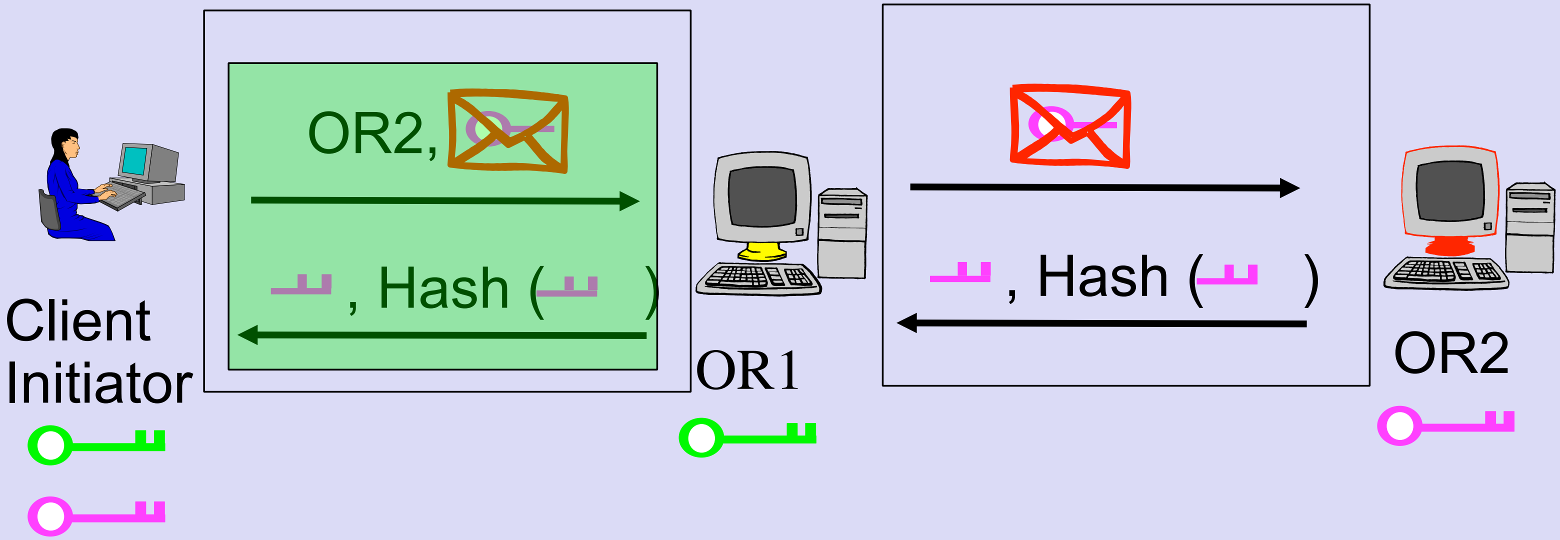
Tor Circuit Setup (Create)



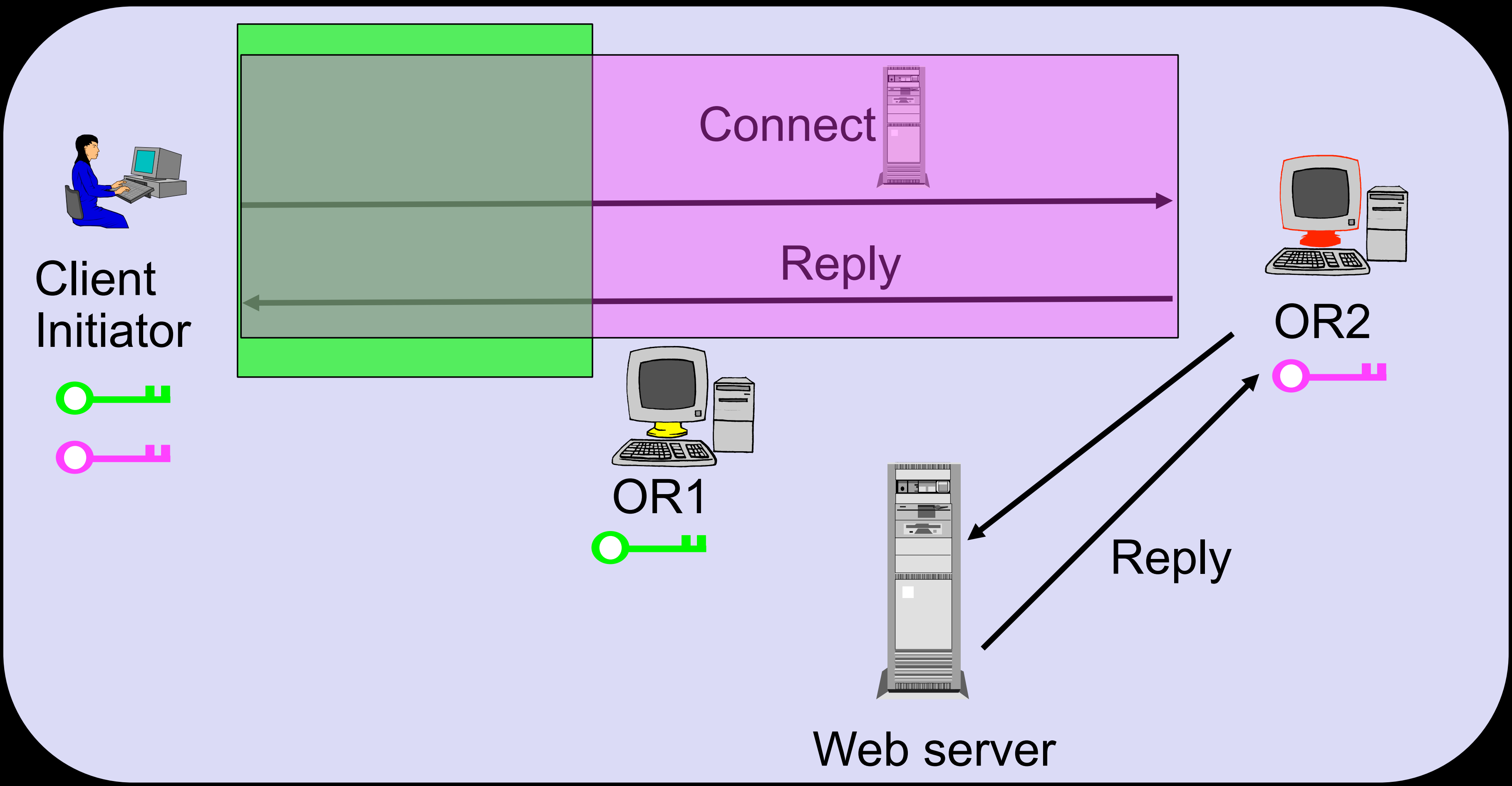
Tor Circuit Setup (Create)



Tor Circuit Setup (Extend)



Tor Circuit Setup (Begin) and Data Flow



More on Tor circuit establishment

Designing your own authentication protocol is error prone. Why not use an established protocol?

Answer: To fit whole messages inside Tor cells. A public key and a signature don't both fit in one 512-byte cell.

Protocol was verified using the NRL protocol analyzer in the Dolev-Yao model.

In 2005 Ian Goldberg found flaw in the way Tor implemented this protocol (checking that a public value was not based on a weak key).

In 2006 Ian proved the (properly implemented) protocol secure in the random oracle model.

Circuit establishment efficiency

I and others have proposed protocols to reduce the public-key overhead of circuit establishment.

Interesting refinements on forward secrecy, but these need more study (and proofs!) before adoption

Next question: How do we know where to build a circuit?

How do we know where to build a circuit? Network discovery.

Flat flooding of network state: complex, tricky, scales in principal but ?

Tor has a directory system

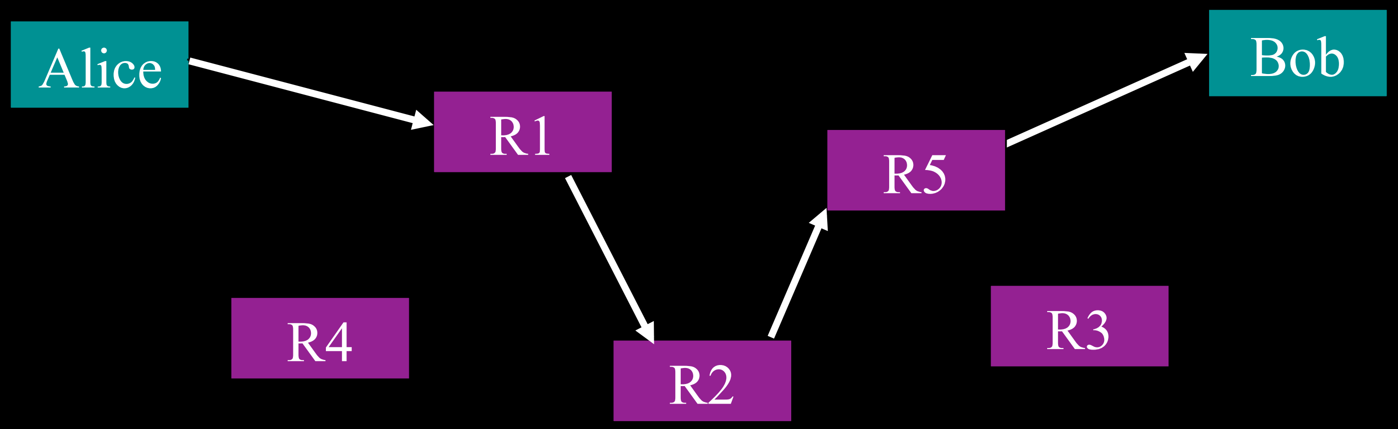
Originally a single directory signing information about network nodes. Then a multiple redundant directory with mirrors. Then a majority vote system. Then a consensus document system. Then separate things that need to be signed and updated frequently. Then...

Bridge distribution: ~~see tomorrow's lecture~~.

Network and Route Discovery

Alice has to know a set of nodes and pick a route from them

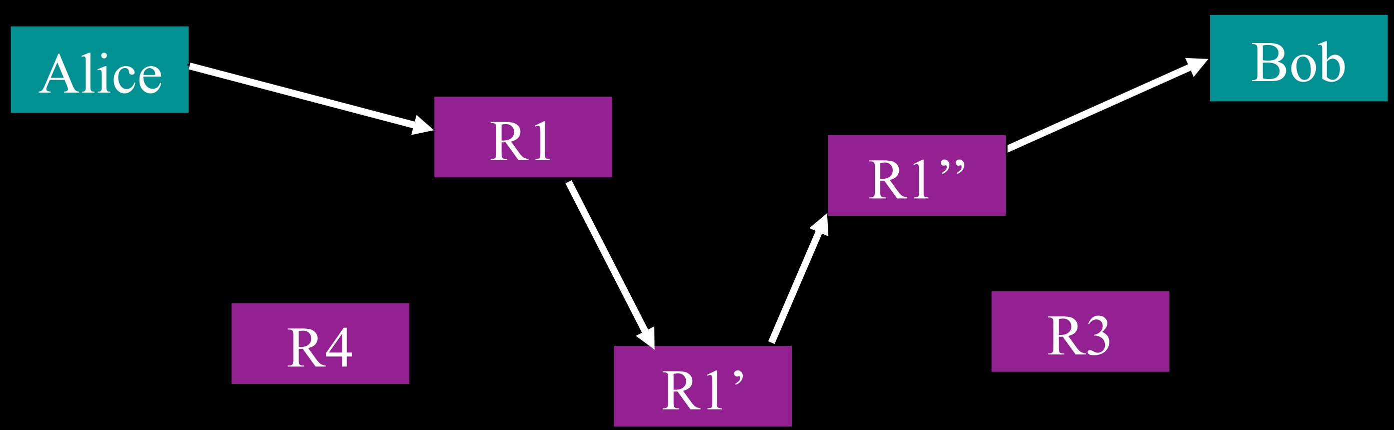
- Must know how to find R1
- Must learn more network nodes to pick a route
- Cannot trust R1 to tell about the rest of the network



Network and Route Discovery

Alice has to know a set of nodes and pick a route from them

- Must know how to find R1
- Must learn more network nodes to pick a route
- Cannot trust R1 to tell about the rest of the network



Network and Route Discovery

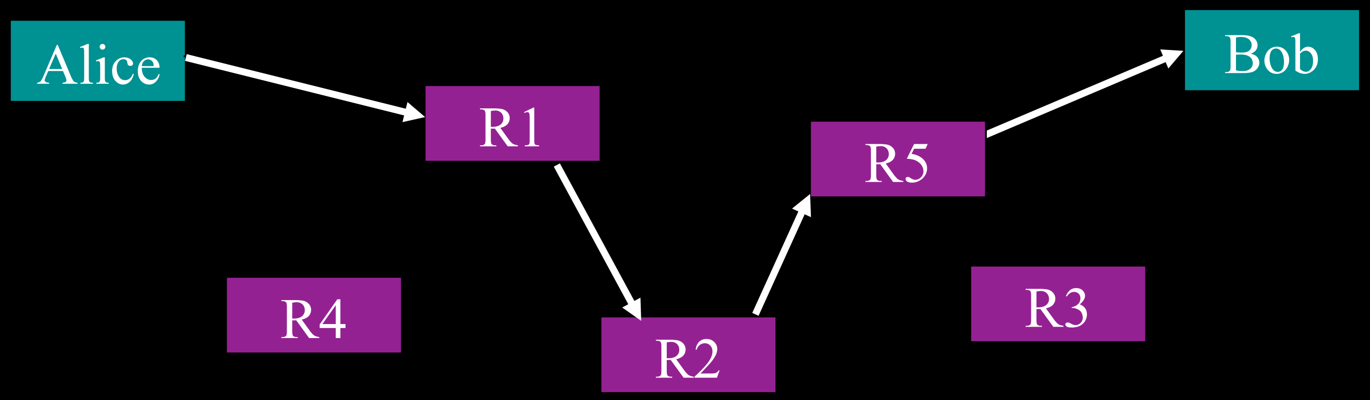
Current simple solution: Trusted servers that tell every Alice about all the nodes in the network

Problem: minimize and distribute that trust. (not current focus)

Problem: Tor currently has c. 2000 nodes. Getting info to its c. 200K-500K clients (some on dial up) is a concern

Scaling: What happens when there are 5000 nodes, 50000 nodes, 5000000 nodes?

- It's not just node names: keys, access policies, state info, etc. to distribute



Scaling Network Discovery and Route Discovery

Simple solution*: Give only partial network information to clients

Possible problems:

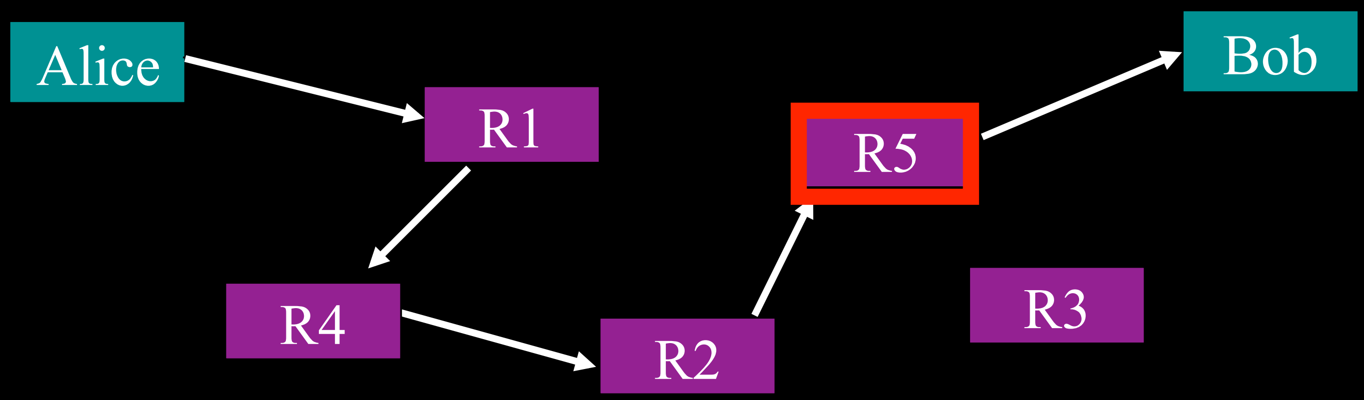
- Network information is not authentic or nodes are not unique (sybils)
- Attacks on how information is distributed (targeting who receives what, oddly skewed distributions of bundles of node information, etc.)

Assume: everyone is fairly given information about a subset of a "clean" network

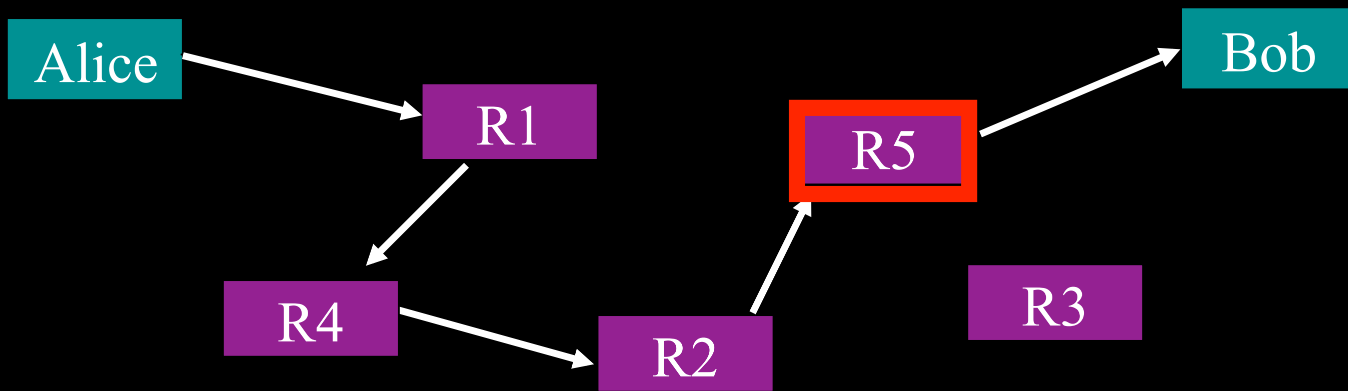
Is anything left to go wrong?

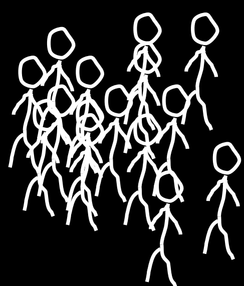
* to fix the problems just identified with our first simple solution

Fingerprinting Attack



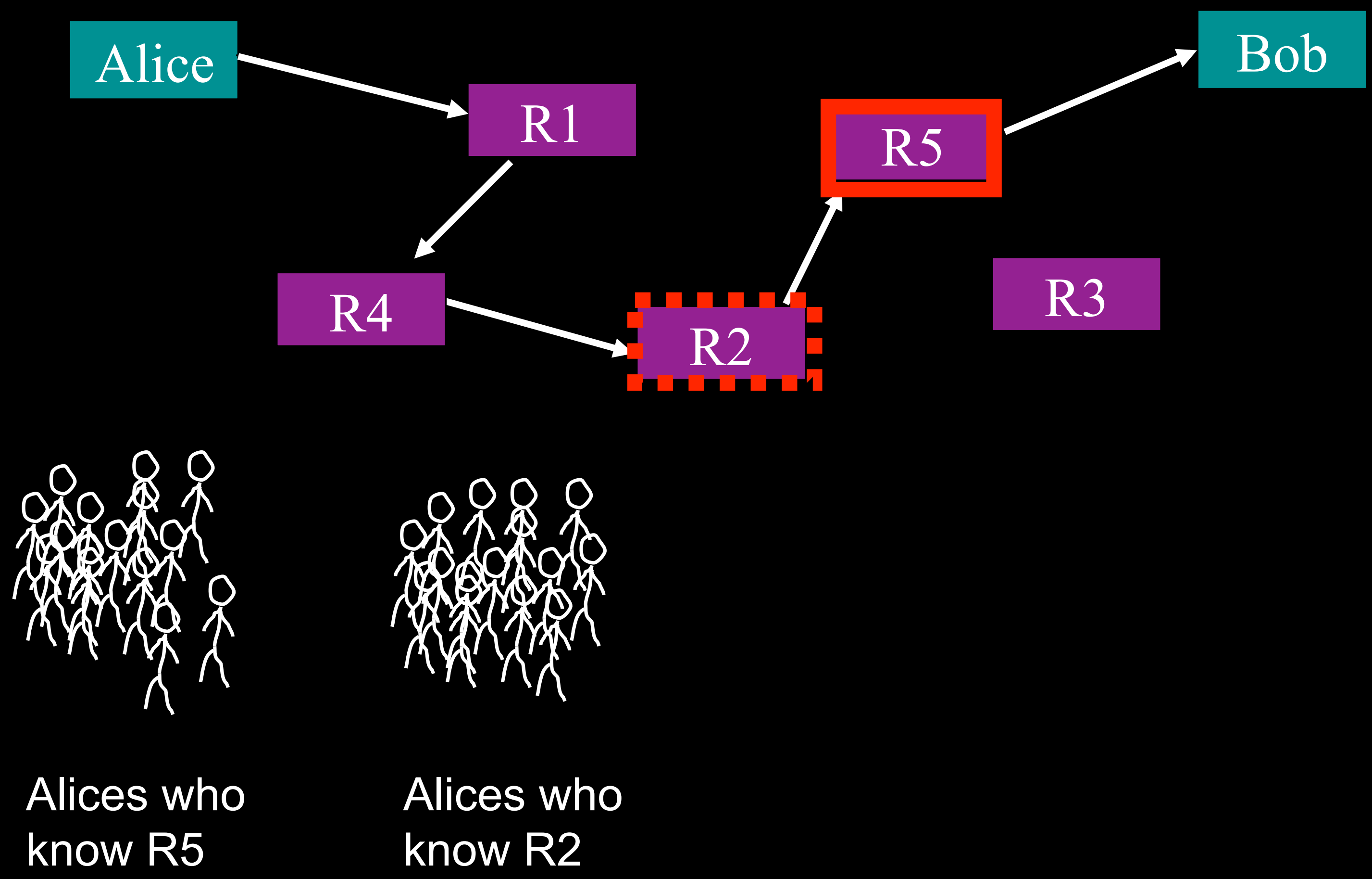
Fingerprinting Attack



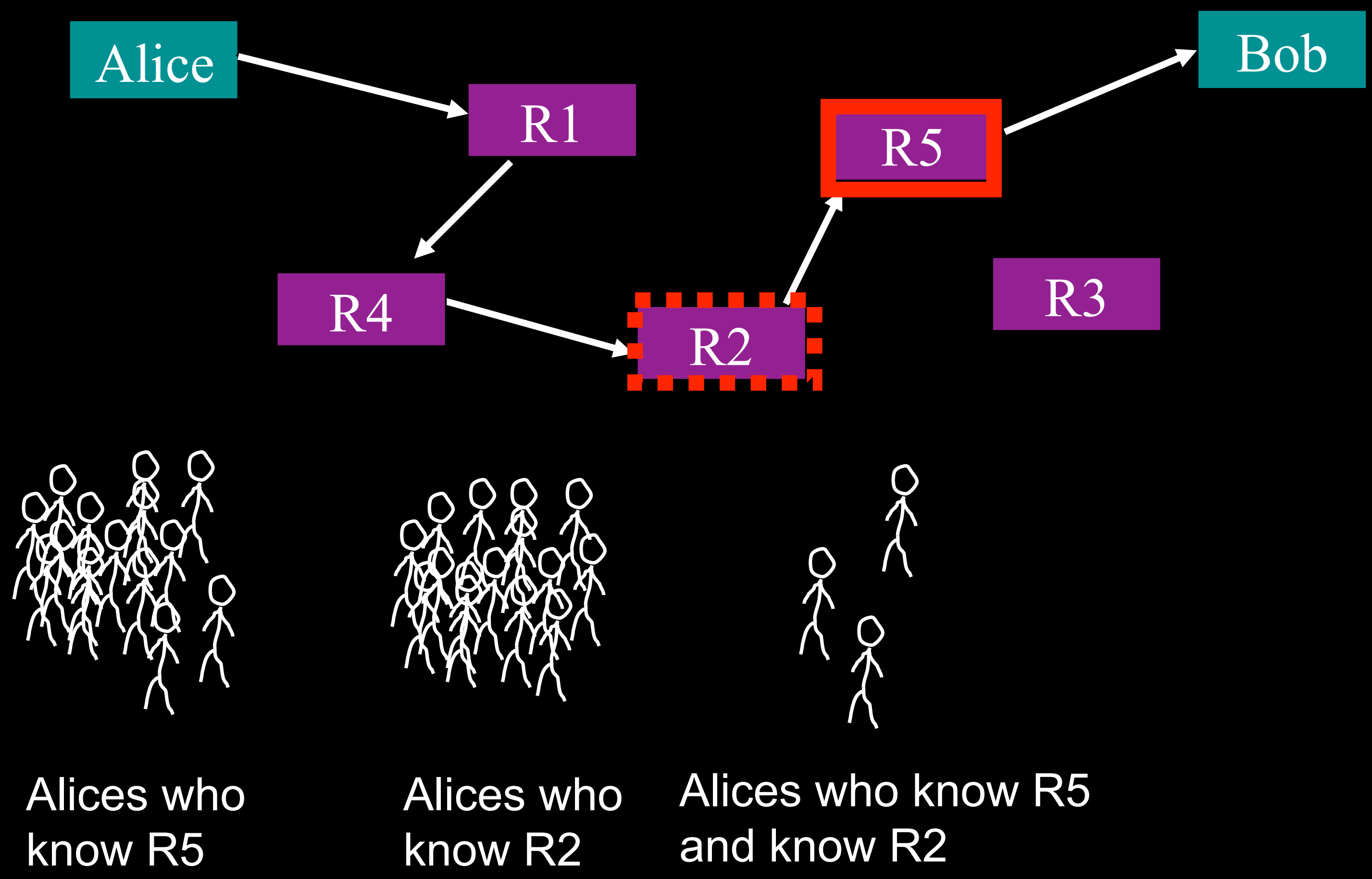


Alices who
know R5

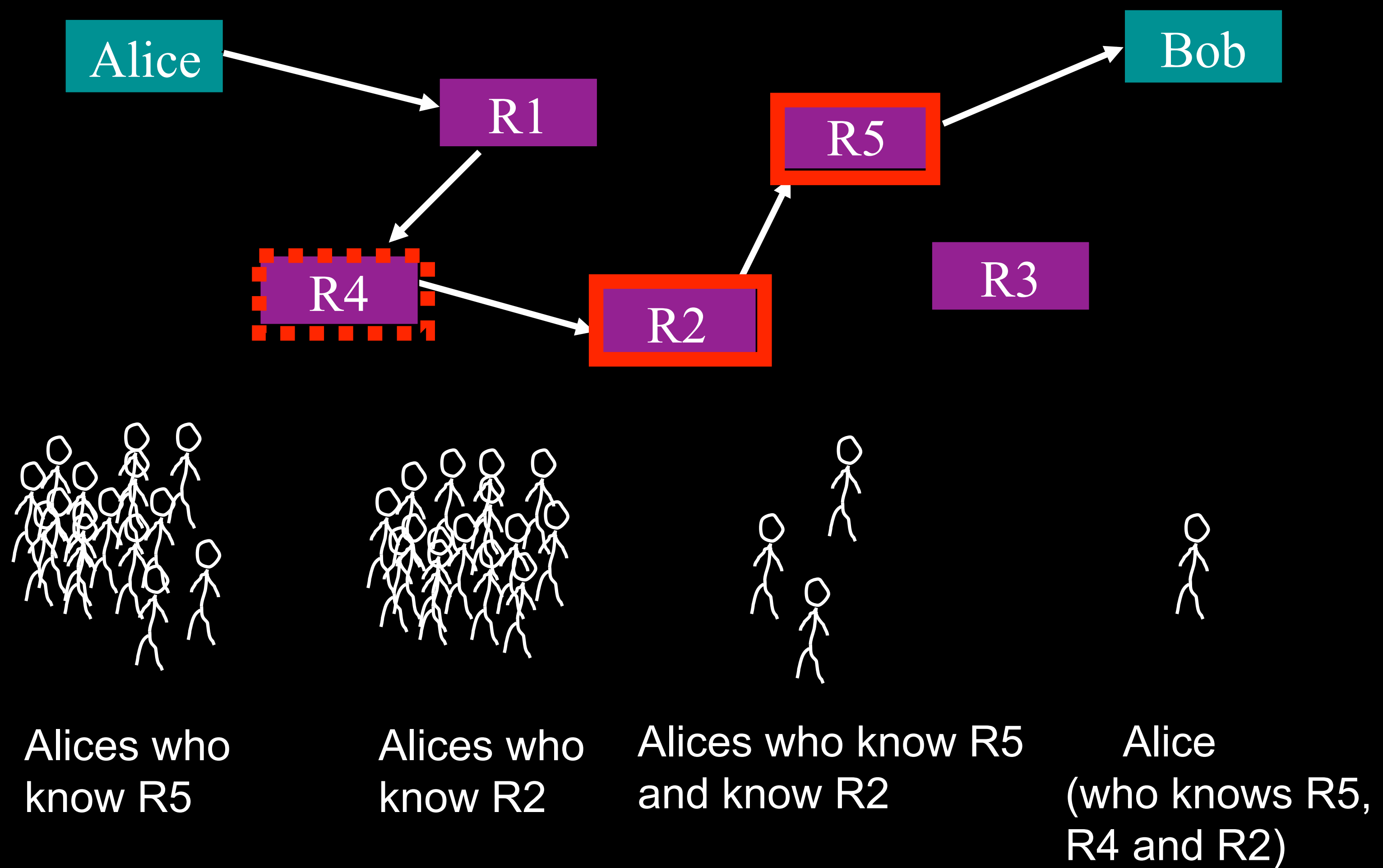
Fingerprinting Attack



Fingerprinting Attack



Fingerprinting Attack



Network Discovery in Early Tarzan (P2P anonymous comms network)

- Network nodes are listed in a DHT, e.g., hash (node name, IP address, public key)
- Join network, pick a small number of nonces
- Pick the node in the DHT with a key closest to each nonce and ask it about its neighbors
- Assume: discovery is "clean and fair"
  - ignoring any issues initial Tarzan has with that
- Given: lookup is visible
  - anyone can tell which part of the network is learned by someone joining the network

Tarzan’s Fingerprints

- Danezis & Clayton observed this vulnerability in Tarzan
- Final published Tarzan design reverts to clique topology (w/ problems noted above)
- Danezis, Syverson ‘08
  - presents analytic proof of results in prior paper
  - implications for scaling practical systems

http://xkcd.com/license

Young Tarzan leaves telltale fingerprints on the vine.

Analyzing the Fingerprinting Attack

Suppose there are N+1 nodes in a system

Suppose each peer knows n nodes

If an adversary knows k of the nodes in a route (it owns them or is adjacent to them in the route), then the number of possible initiators (as $k/N \rightarrow 0$) tends to

$$n^k / N^{k-1}$$

Proof: See the paper.

Epistemic Attacks

To avoid problems based on what senders know, designs have been cautious about allowing only partial discovery.

“There are known knowns. These are things we know that we know.

Epistemic Attacks

To avoid problems based on what senders know, designs have been cautious about allowing only partial discovery.

"There are known knowns. These are things we know that we know.

There are known unknowns. That is to say, there are things that we know we don't know." ---Donald Rumsfeld

Bridging Attack (Adversary making use of what we don't know.)

Anonymity loves company but hates a crowd

As the network grows these attacks become more effective (n/N shrinks)

Against fingerprinting, client-server infrastructure design appears to beat P2P

- A system like Tor has two orders of magnitude more clients than servers, so way more clients share knowledge of server sets than if all were peers

Better to have nothing to do with each other than to stay together in ignorance

Suppose a setting roughly like current Tor
- 200K clients, 2000 nodes
- assume we want anonymity set size of 50K

Against fingerprinting each client must know 1000 nodes (about half)

If client and node sets each partitioned, then the same anonymity set size against fingerprinting if clients know only 500 nodes

Not just more efficient. Much easier to design discovery and show secure in simple partitioned clique case than partial knowledge case.

Incentives, usability, network effects

Just saw one network effect: client-server currently beats P2P for efficient, simple resistance to epistemic attacks on discovery

Also, client-server more flexible to be usable by larger variety of users

➔ more users ➔ more security

Client-server and exit/entry policies is more flexible to be usable by larger variety of providers

➔ more nodes ➔ more security

If not everyone is provider, who are the providers?

Why a volunteer network?

A decade ago anonymity needs not obvious to even those with strong needs, so they wouldn't pay for it.

Even if they would, anonymity has a special network effect problem

- High security needs users cannot use the network unless it has lots and varied users
- Low (perceived) security needs users will not use the network if it is expensive or hard to use

➔ Need to allow "free-riders" (not really free-riders since they contribute to the security of others)

➔ Need easy usability and acceptable perceived performance

Incentive design decisions in early onion routing

Carry traffic for others to make system usable for Navy/government purposes.

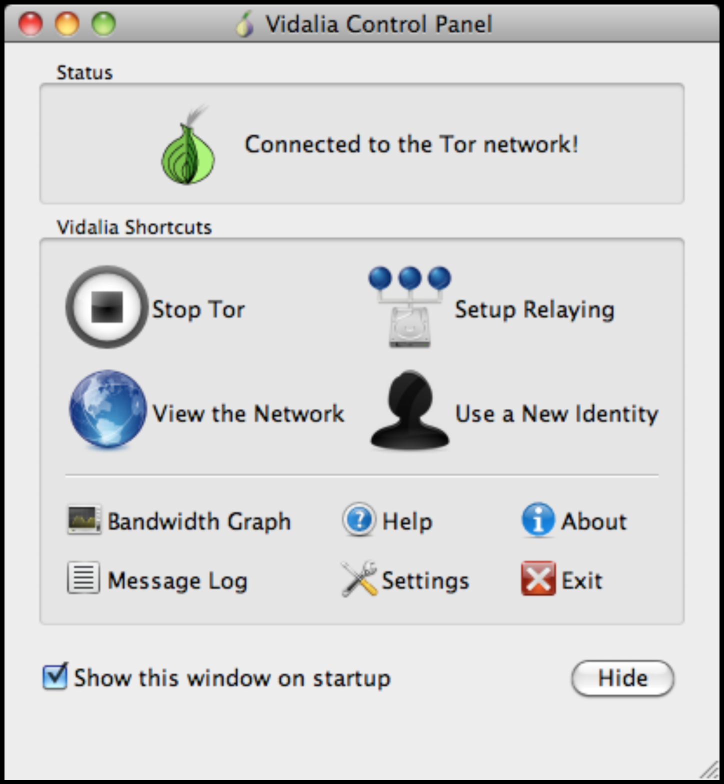
Let others run part of the infrastructure so they can trust it.

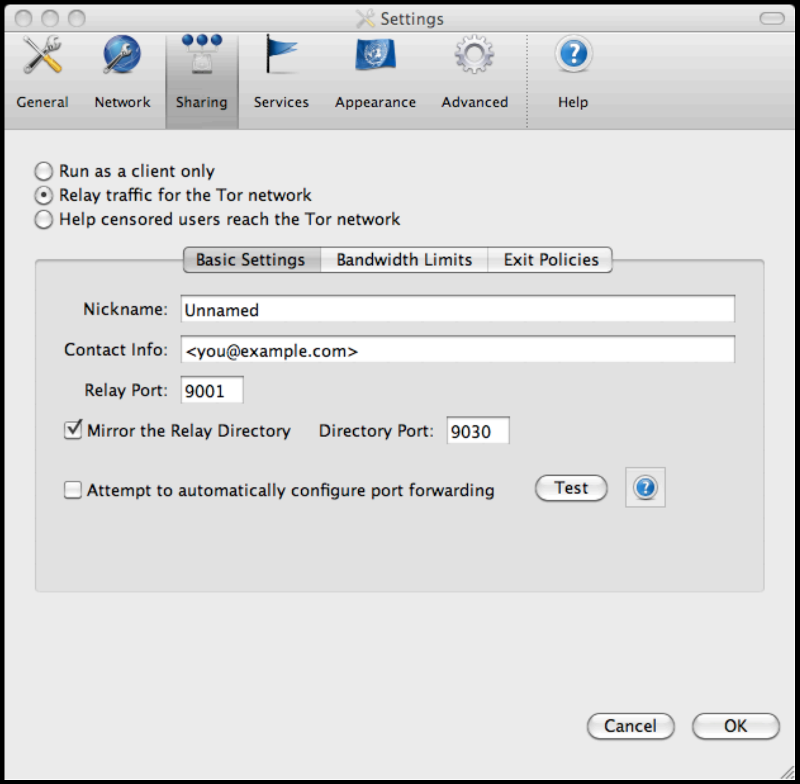
Make code open source so they can trust it. (only later: so they can contribute to research and development)

Client-server architecture for those who can't/won't run nodes.

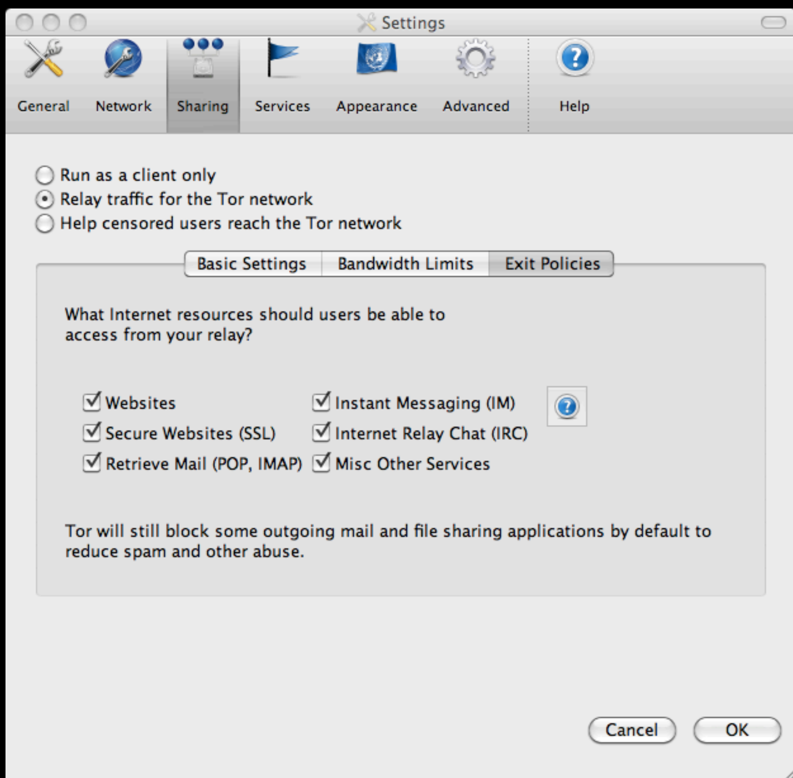
Entry and exit policies for variety of network operator policy environments and comfort levels.

Operator options good, if easy to configure





Operator options good, if easy to configure



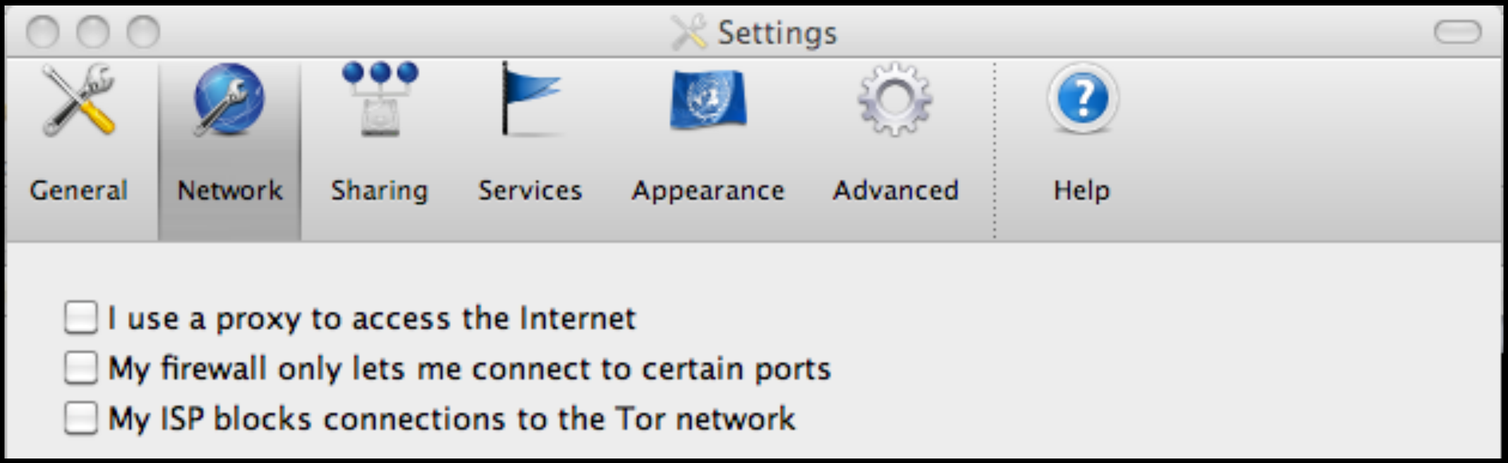
User options mostly a bad idea

Most users don't know how to configure properly

→ System should just start and work (if it can)

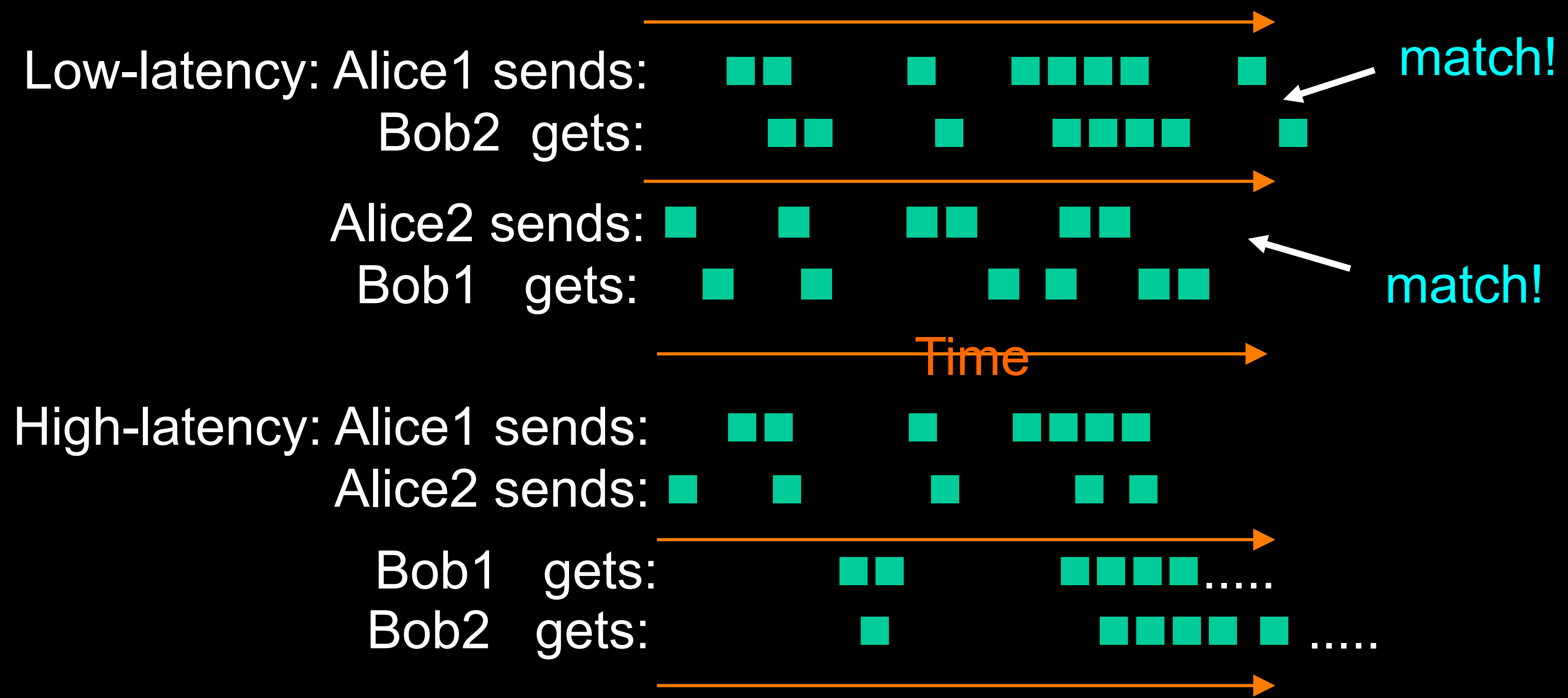
More options → more ways to partition and ID

→ System should not make it easy for end users to choose other than starting defaults



The most secure system design (ignoring incentives and usability issues) is not the most secure system design

The most secure system design (ignoring incentives and usability issues) is not the most secure system design



Prevailing Wisdom: High latency systems more secure but less practical

- Much harder to do correlation attacks
- Somewhat harder to do intersection and statistical disclosure attacks
- Cannot be used for interactive or low-latency applications: web browsing, remote login

- Prevailing Wisdom: High latency systems more secure but less practical
  - Much harder to do correlation attacks
  - Somewhat harder to do intersection and statistical disclosure attacks
  - Cannot be used for interactive or low-latency applications: web browsing, remote login

What is a realistic adversary for practical anonymous internet communication?

The Man

The Man

Owns big chunks of the anonymity infrastructure
- purchased, compromised,...

Can access many ISPs, backbones, websites, ...

Can know ancillary things
- employer, relatives, religion, political activities,...

If targeting you, can tap your phone, tail and photograph you,...

Think intelligence orgs., secret police, state actors, organized crime, ...

The Man

Big

Powerful

The Man

Big

Powerful

NOT global

NOT omnipotent

Don't mix with The Man

For internet communication: If you are not worried about being suspected by The Man, mix networks are overkill

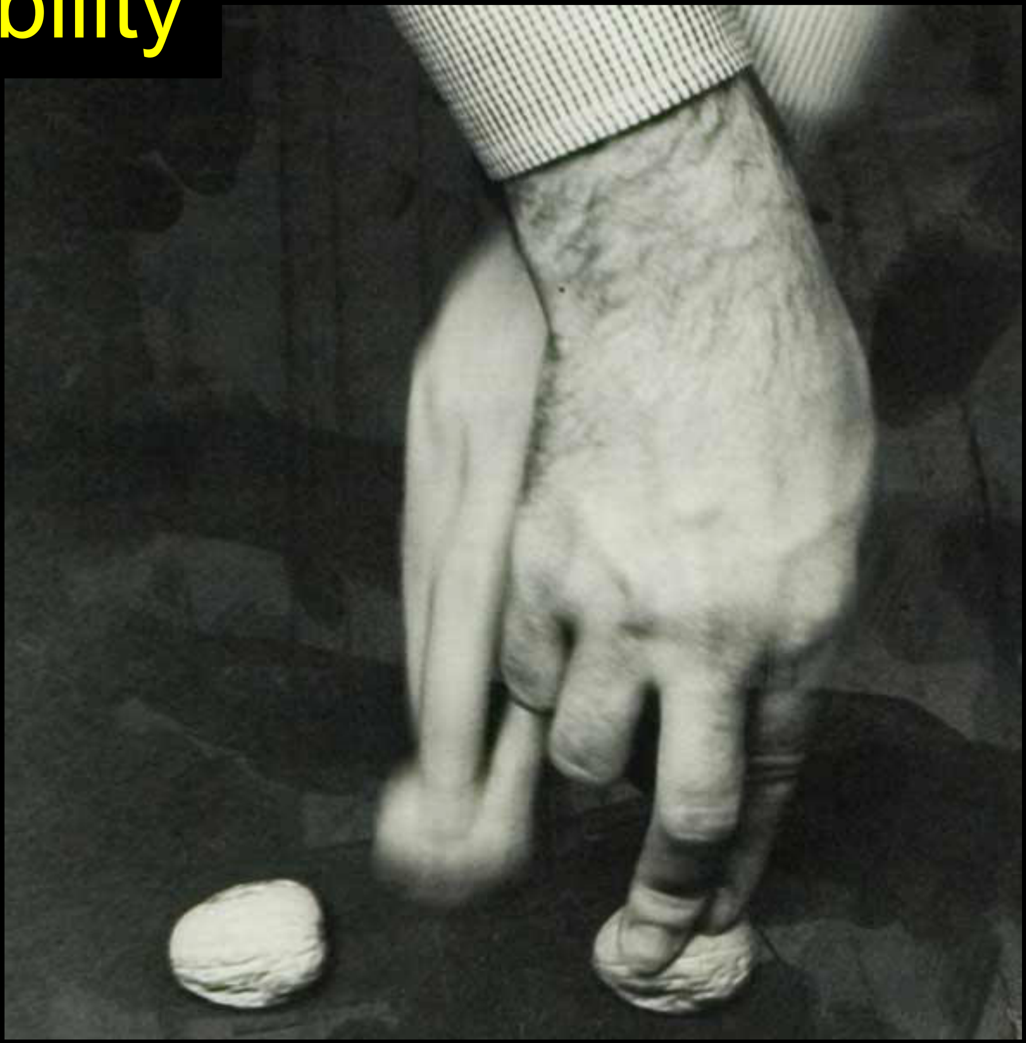
If you are worried about being suspected by The Man, mix networks are inadequate

- because they don't scale in practice

Mixes can provide plausible deniability: The Man won't know which of 50-100 suspects is the sender

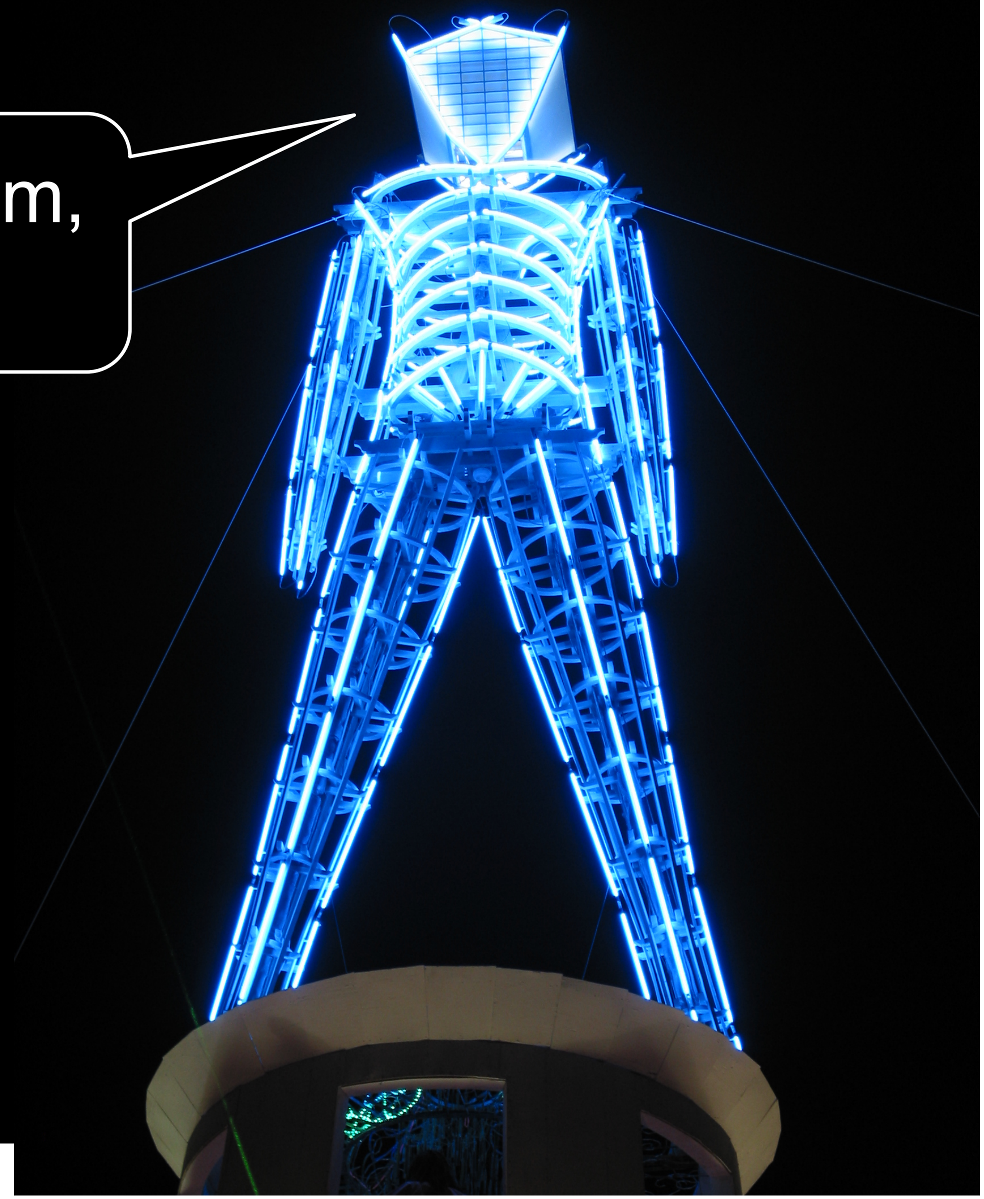
- For most anonymous internet communication this is irrelevant

The Man doesn't care about plausible deniability



The Man doesn't care about plausible deniability

I'll pick, hmmm,
All three!

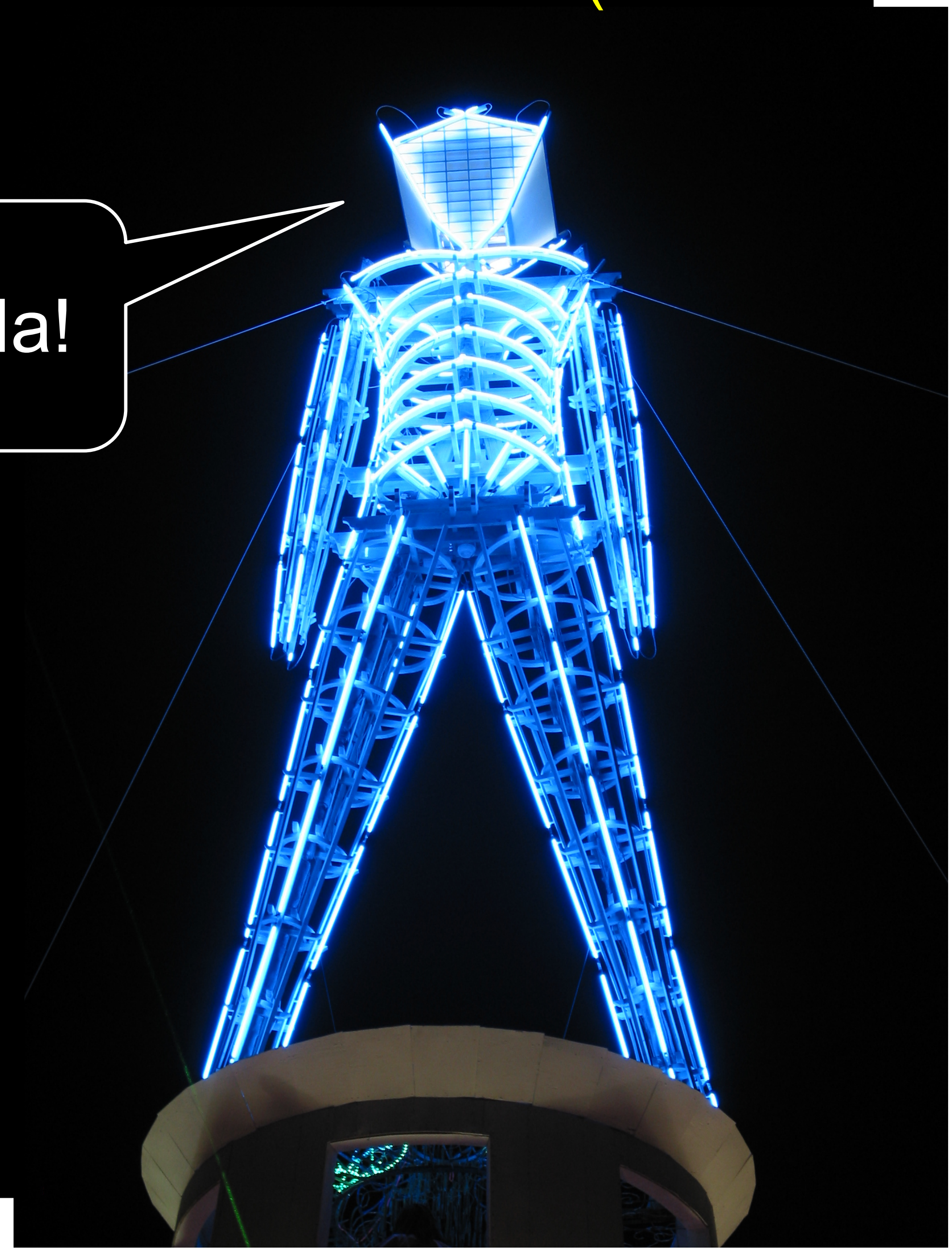


Mix networks will not scale, so onion routing is actually more secure

- Technically they can scale, but they won't because of usability and incentives
- Most people are (correctly) not worried about The Man. They want anonymity from
  - Employers (current or potential), Marketing or government hoovers, Identity thieves, Abusive ex spouses, Business competitors, Unscrupulous websites, Flaming lunatics...
- Most will choose a low-latency, interactive system for protection
- So, Mixmaster has at most 100-200 users per day protected by a few dozen mixes
- By contrast, Tor has 100K-600K users at once protected by thousands of onion routers

Tor ain't gonna save you from The Man neither (not statistically).

Need to add trust.



Mix networks will not scale, so onion routing is actually more secure

Technically they can scale, but they won't because of usability and incentives

Most people are (correctly) not worried about The Man. They want anonymity from

- Employers (current or potential), Marketing or government hoovers, Identity thieves, Abusive ex spouses, Business competitors, Unscrupulous websites, Flaming lunatics...

Most will choose a low-latency, interactive system for protection

So, Mixmaster has at most 100-200 users per day protected by a few dozen mixes

By contrast, Tor has 100K-600K users at once protected by thousands of onion routers

Why volunteer to run a node?

Desire to contribute to something important.

Desire to be cool.

Provide more/better service → Attract users to the network → Cover for your own traffic.

Running your own node other nodes cannot distinguish your own traffic from traffic from those you attracted to the network.

- True but ...

Circuit clogging attacks (simple version)

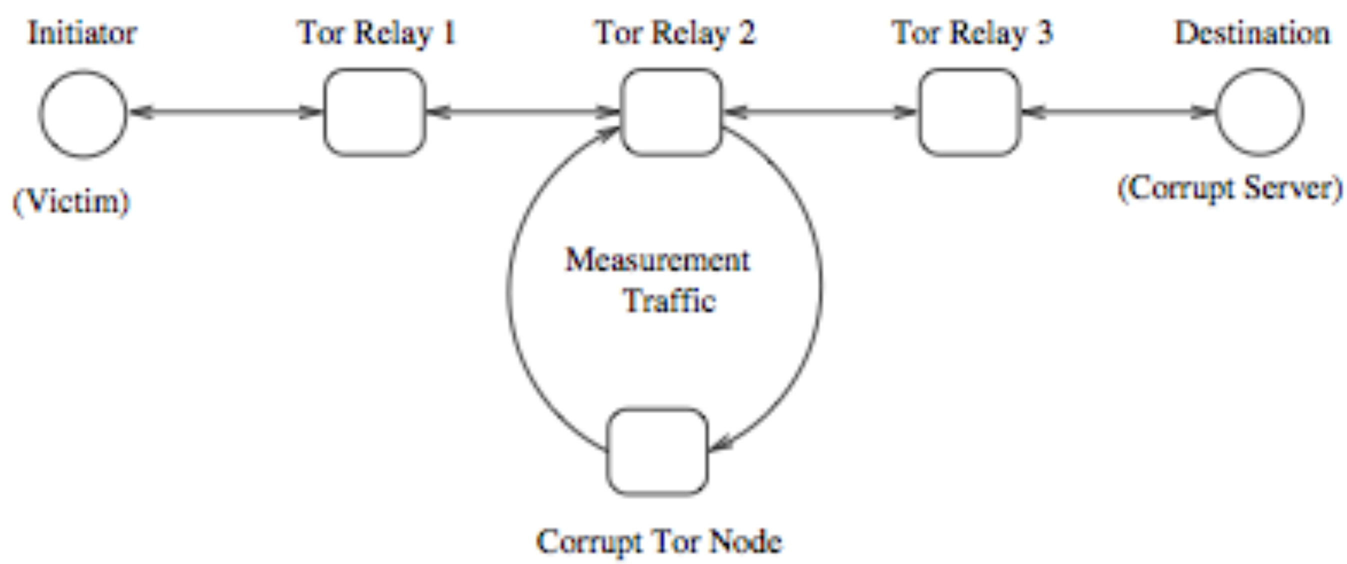


Figure 1. The attack setup

From "Low-Cost Traffic Analysis of Tor", Murdoch & Danezis, Oakland '05

Limitations of simple circuit clogging

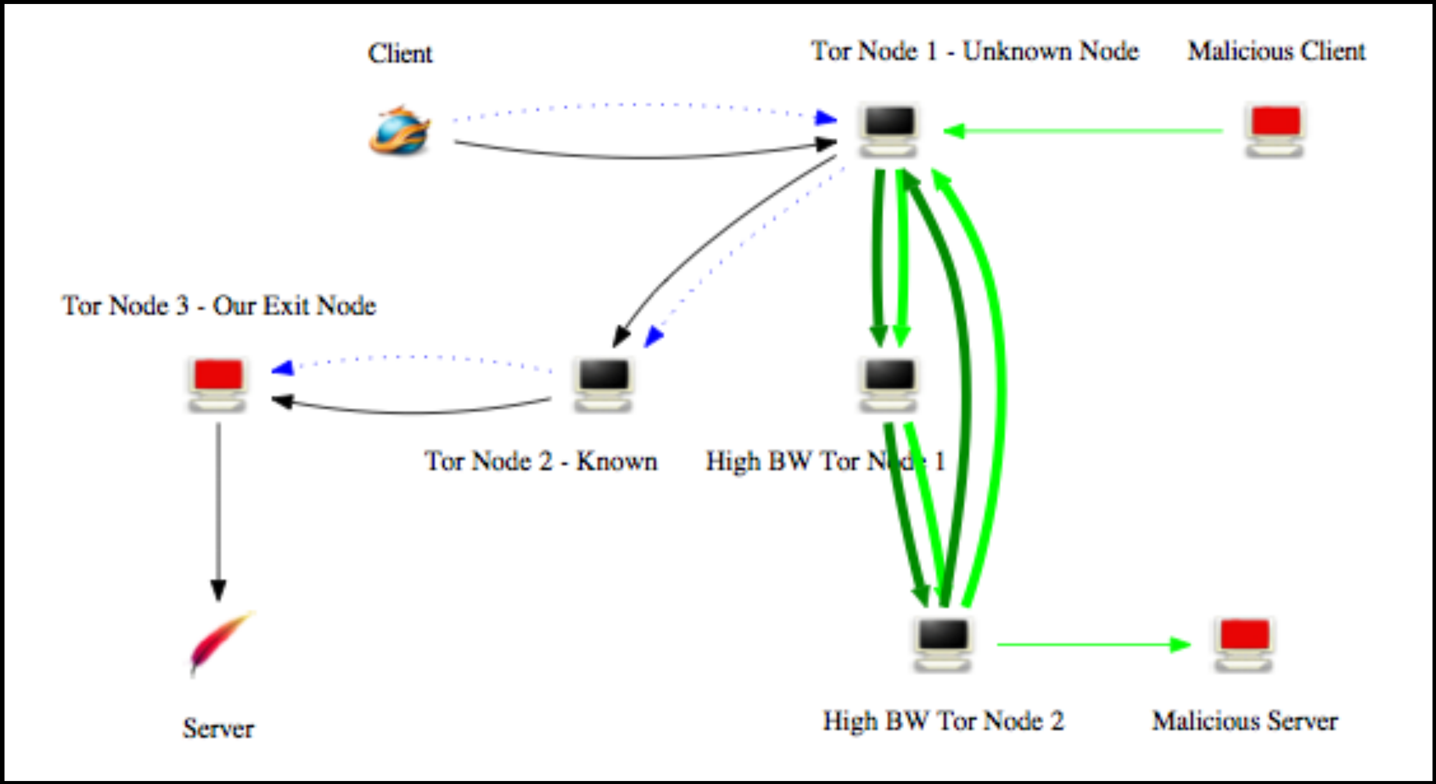
Required a hostile destination

Only identified the onion routers, NOT the client

Only worked on a small network

- Public Tor network was c. 40 nodes at the time
- Later verified not to work on Tor network in 2008 (1500 nodes, many high capacity)
  - Numerous false positives and negative

Long paths for clogging attack bandwidth multiplier



From "A Practical Congestion Attack on Tor Using Long Paths", Evans, Dingledine, & Grothoff, USENIX Sec '09

Long-path congestions details

Requires client to use hostile exit node (to inject javascript or other pinging mechanism)

- Could also work with hostile destination

Also requires another hostile client and hostile destination to clog circuits

Currently countered by preventing Tor from generating long circuits

- Can still work but requires adversary to contribute more resources

Could also be countered by traffic prioritization

- gold star routers
- trust
- payment

While we're back on incentives for being a router, what about incentives for clients?

- Tang and Goldberg CCS'10 use exponentially weighted moving average to select for latency over throughput, which has greatly improved Tor performance

Morals: incentives and usability

Incentives and usability greatly influence system performance and system adoption

Almost always overlooked: They also greatly influence system security

A threat model that tells you which system is more secure without accounting for these issues is almost certainly wrong

What's up next

Lecture 3:

- Formalization and analysis, possibilistic and probabilistic definitions of anonymity
- Hidden services: responder anonymity, predecessor attacks revisited, guard nodes

Lecture 4:

- Link attacks
- Trust

Questions?