Vulnerabilities in Tor: past, present, future

Roger Dingledine
The Tor Project
https://www.torproject.org/

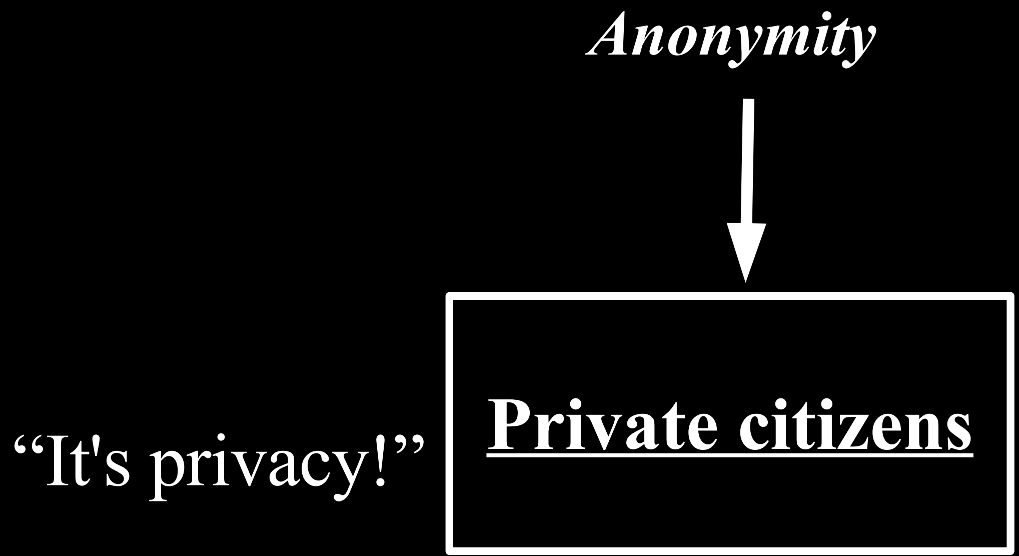
Outline

- Crash course on Tor
- Solved / solvable problems
- Tough ongoing issues, practical
- Tough ongoing issues, research
- Future

Tor: Big Picture

- Freely available (Open Source), unencumbered.
- Comes with a spec and full documentation: Dresden and Aachen implemented compatible Java Tor clients; researchers use it to study anonymity.
- 1500 active relays, 200000+ active users, >1Gbit/s.
- Official US 501(c)(3) nonprofit. Seven funded developers, dozens more dedicated volunteers.
- Funding from US DoD, Electronic Frontier Foundation, Voice of America, a French NGO, Google, NLnet, ...you?

Anonymity serves different interests for different user groups.

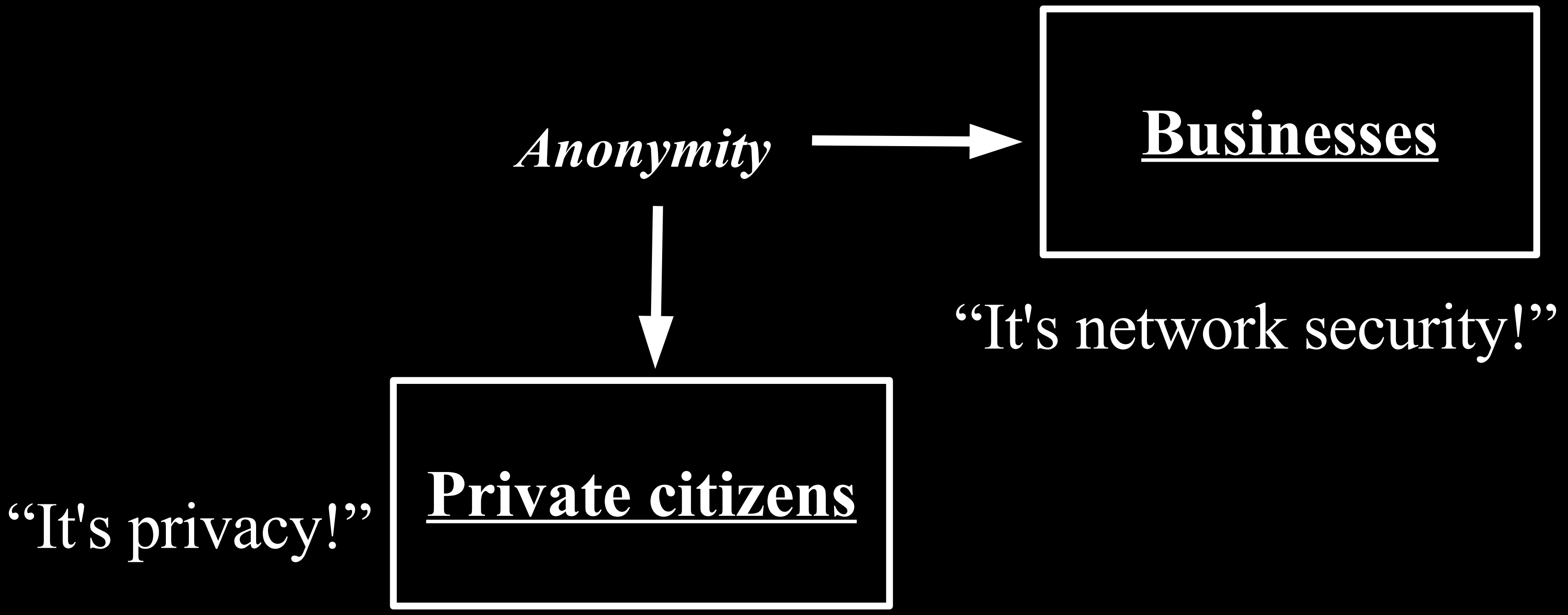


Anonymity

Private citizens

“It's privacy!”

Anonymity serves different interests for different user groups.



Anonymity

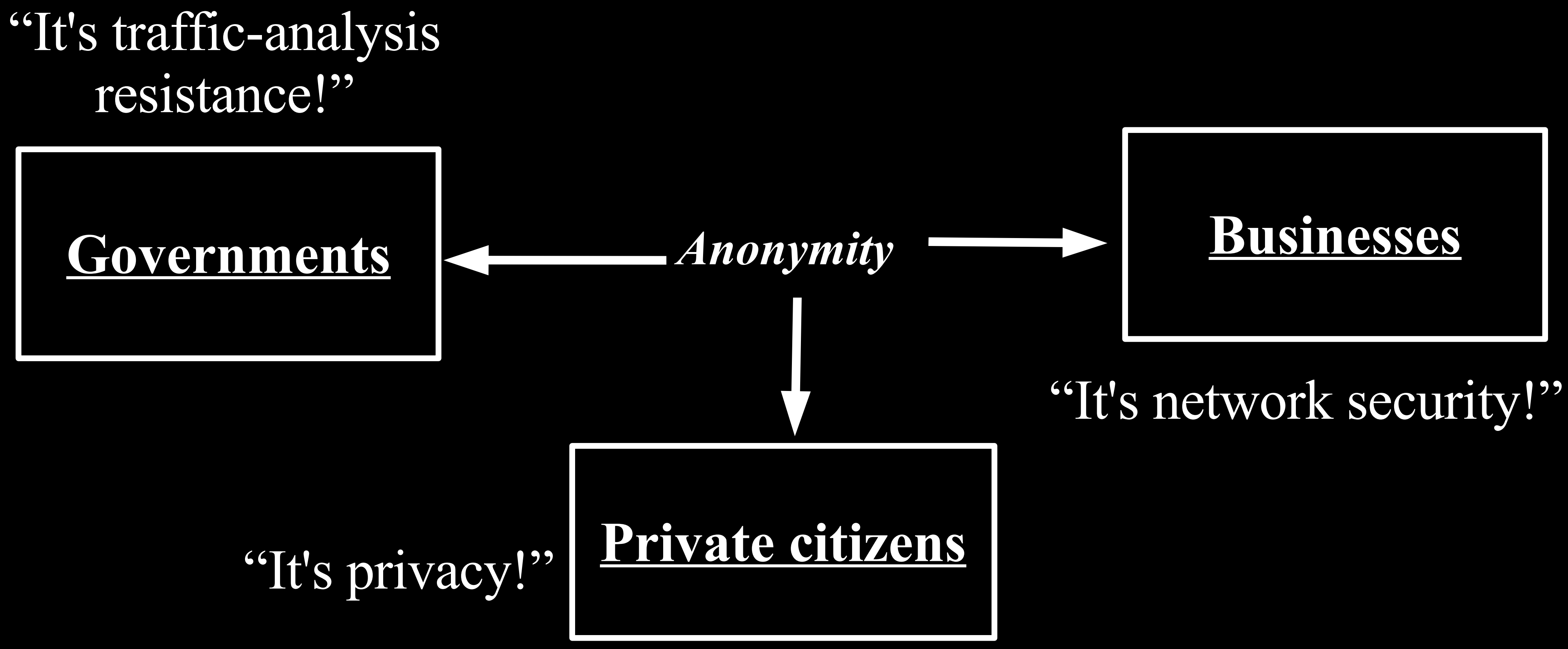
Businesses

"It's network security!"

Private citizens

"It's privacy!"

Anonymity serves different interests for different user groups.



“It's traffic-analysis resistance!”

Governments

Anonymity

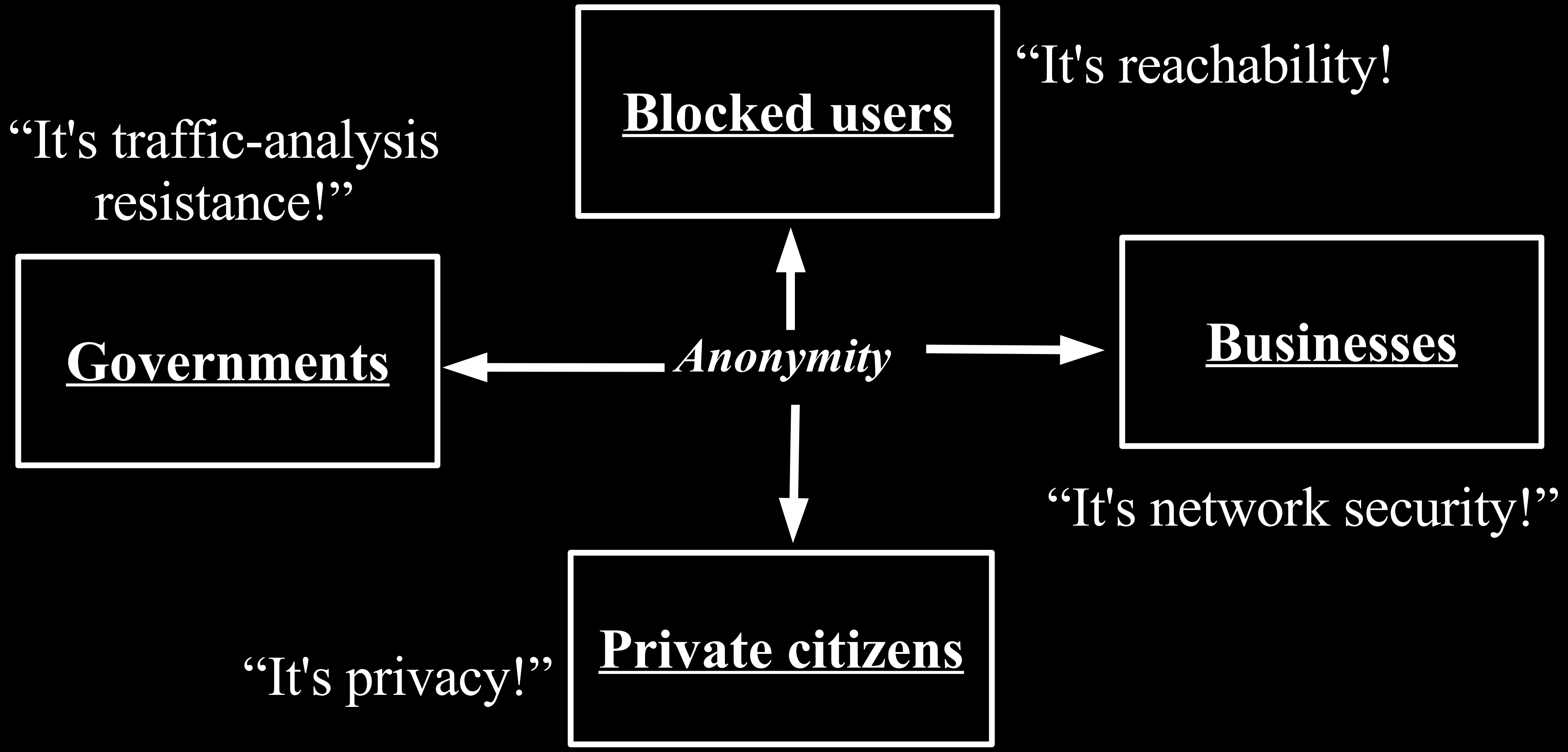
Businesses

“It's network security!”

Private citizens

“It's privacy!”

Anonymity serves different interests for different user groups.



Anonymity

Blocked users

“It's reachability!

Governments

“It's traffic-analysis resistance!”

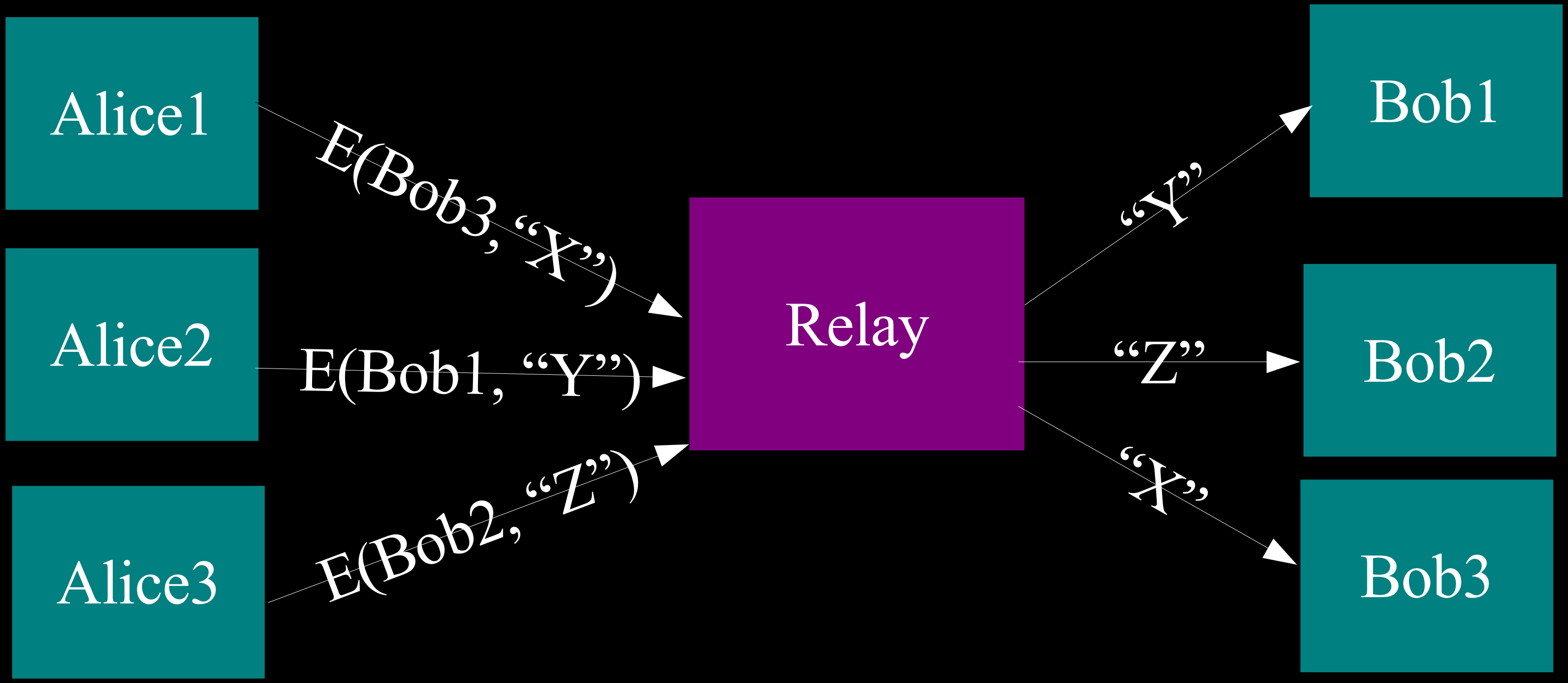
Businesses

“It's network security!”

Private citizens

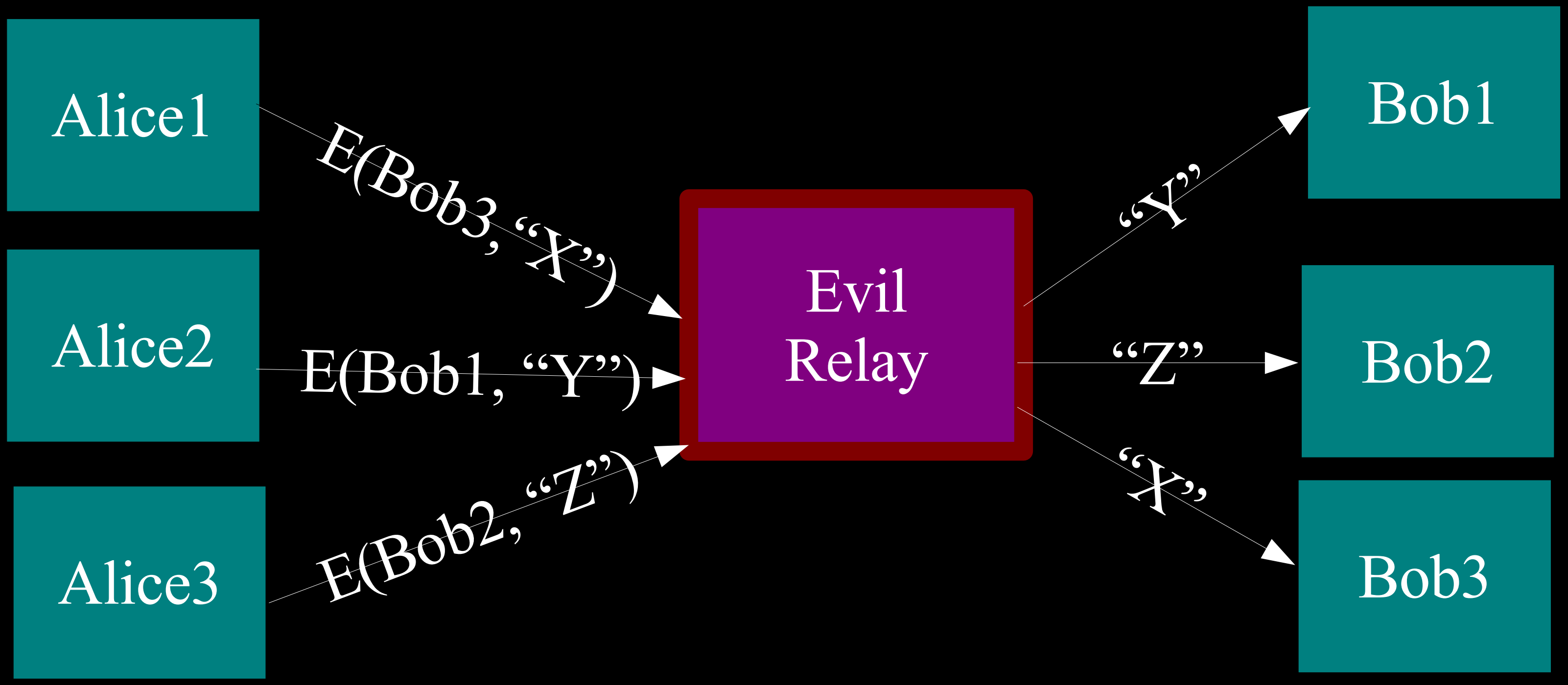
“It's privacy!”

The simplest designs use a single relay to hide connections.

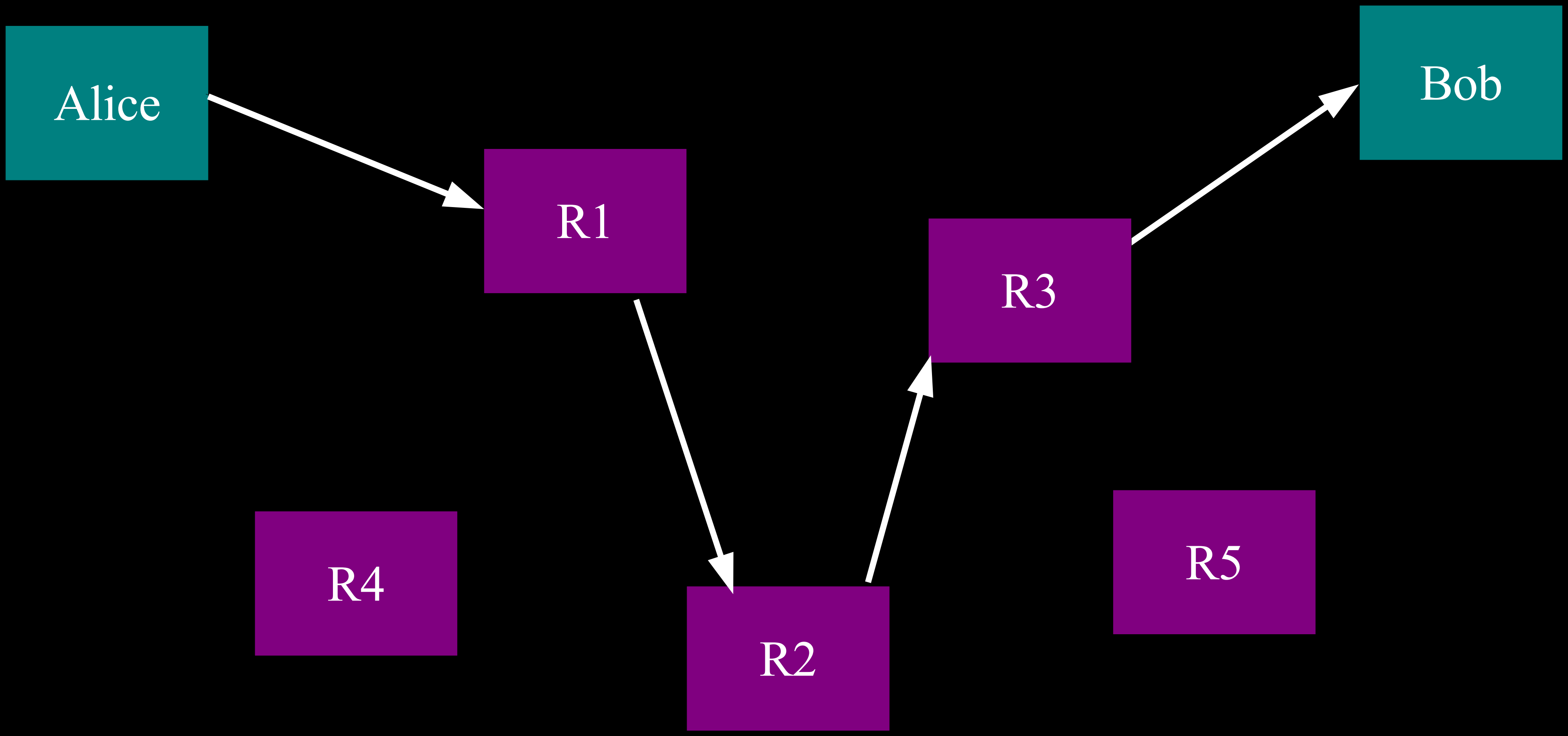


(example: some commercial proxy providers)

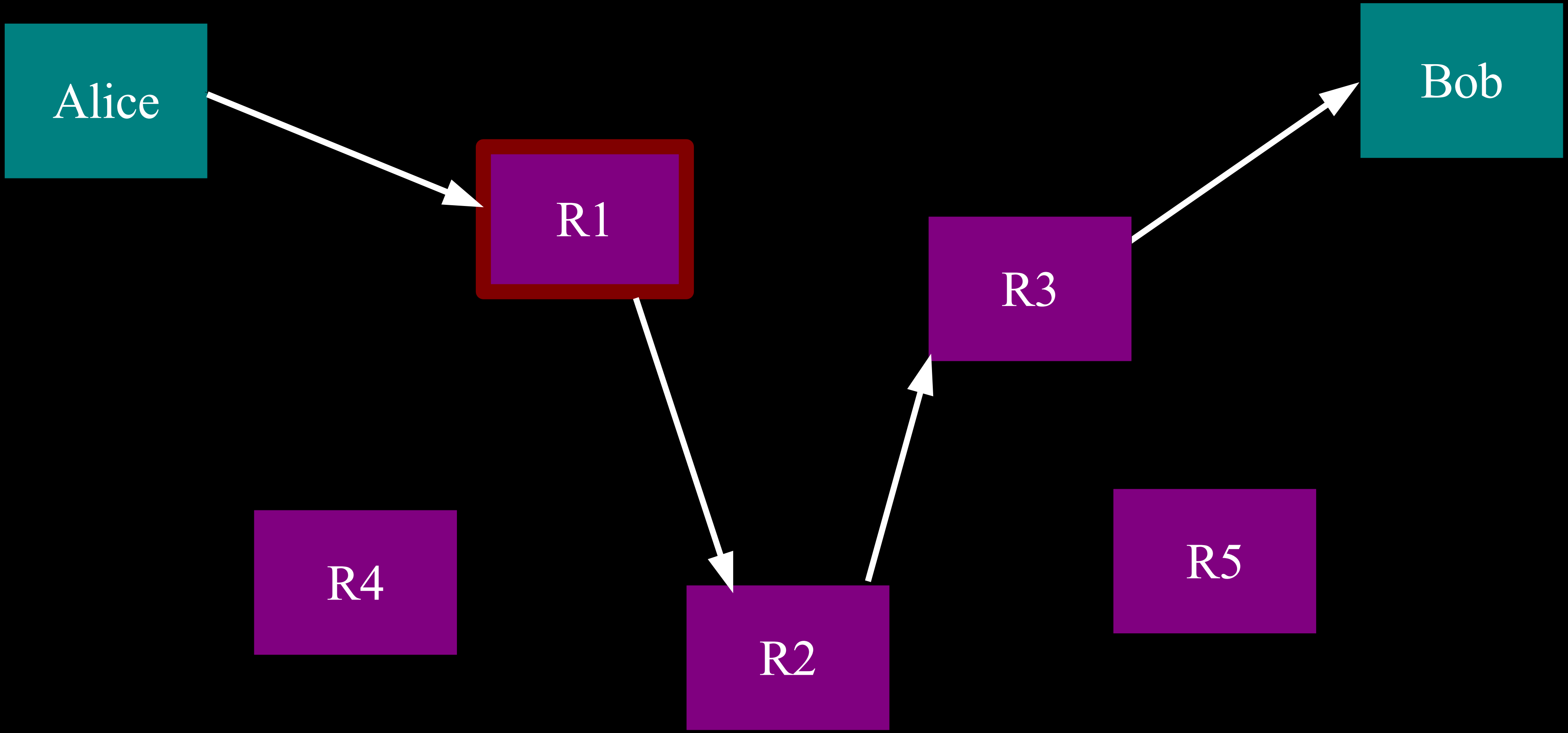
But a single relay (or eavesdropper!) is a single point of failure.



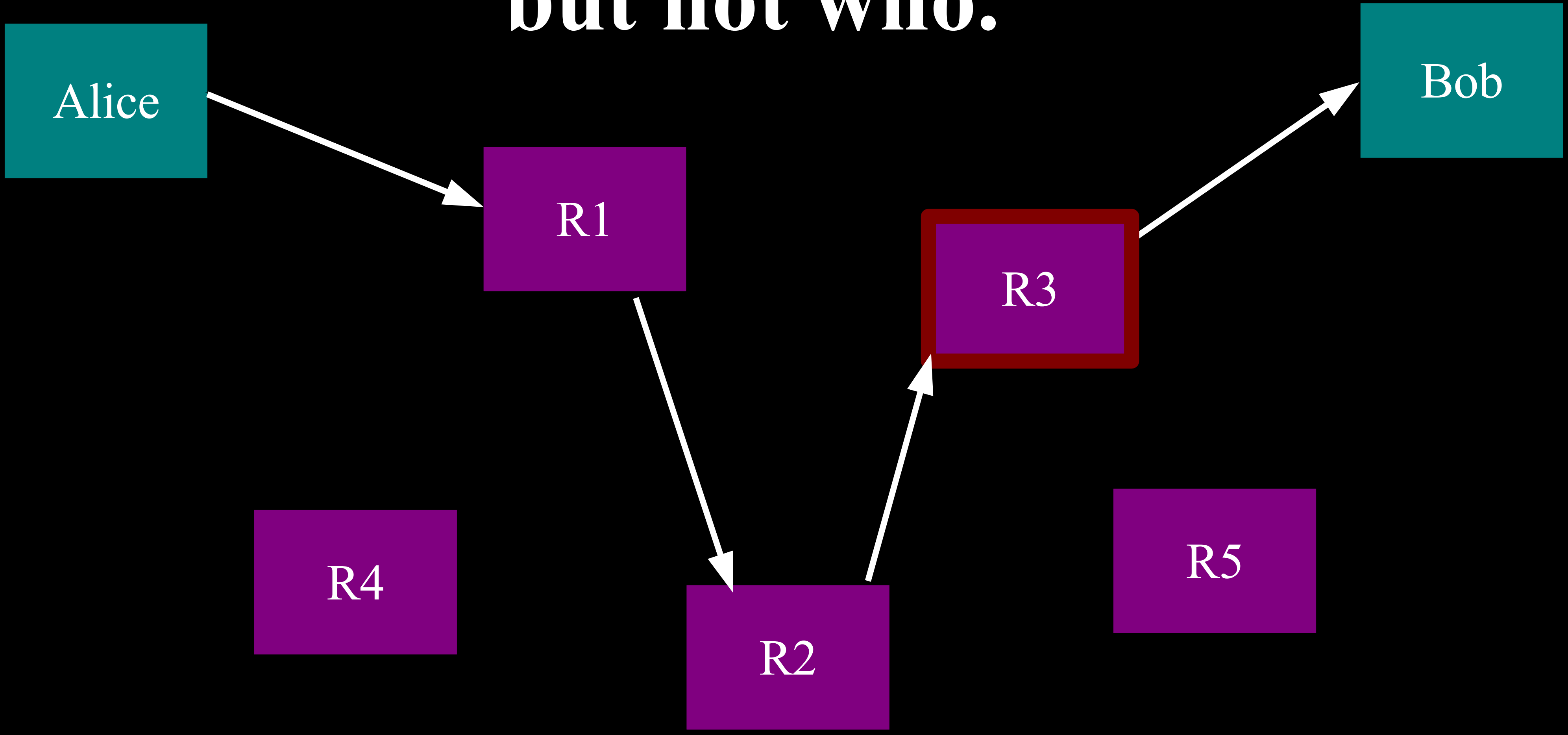
So, add multiple relays so that no single one can betray Alice.



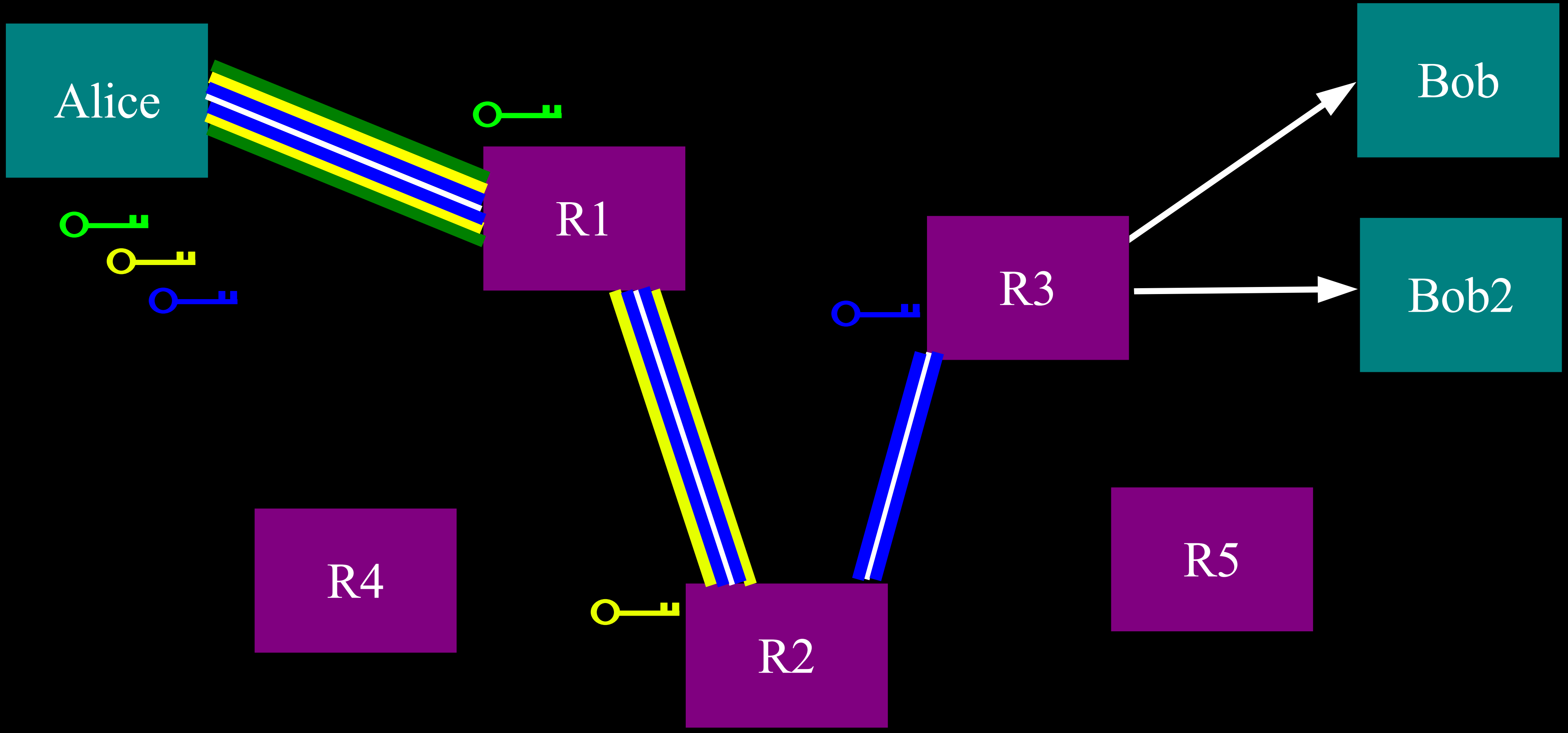
A corrupt first hop can tell that Alice is talking, but not to whom.



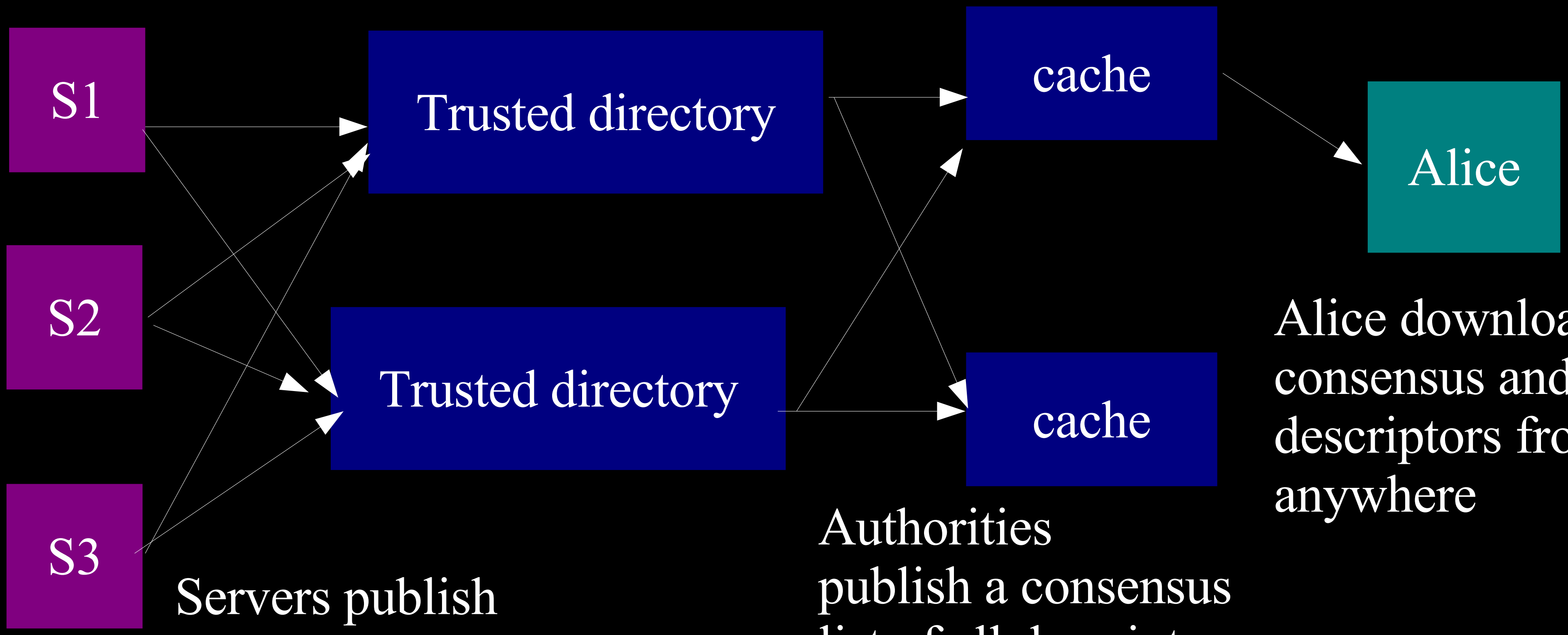
A corrupt final hop can tell that somebody is talking to Bob, but not who.



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



The basic Tor design uses a simple centralized directory protocol.



Servers publish self-signed descriptors.

Authorities publish a consensus list of all descriptors

Alice downloads consensus and descriptors from anywhere

Outline

- Crash course on Tor
- ***Solved / solvable problems***
- Tough ongoing issues, practical
- Tough ongoing issues, research
- Future

16-bit AES counter mode

- [Fixed in Tor 0.0.6.1, 6 May 2004]
- At the time, OpenSSL didn't have AES. Later, it still didn't have counter mode.
- We were resetting our counter after 16 bits.
- Conclusion: a second implementation is a *really* good idea.

2-byte relay cell length overflow

- [Fixed in Tor 0.1.0.10, 14 June 2005]
- When we moved our cell size from 256 bytes (length can fit in 1 byte) to 512 bytes (length fits in 2 bytes), we forgot to check if the cell claims a length > 512.
- ...which we then write out onto the network

Diffie-Hellman handshake bug

- [Fixed in Tor 0.1.0.14, 8 Aug 2005]
- OpenSSL didn't check for trivial keys (like g^0) in DH keys. (Now it does.)
- This meant your entry hop could MitM you and spoof the whole rest of the network.

Keep building circuits until you lose

- [Fixed in Tor 0.1.1.11-alpha, 10 Jan 2006]
- Attacker runs a few relays and waits for you to choose them as first and last hop
- (Or runs just one relay and induces your hidden service to build circuits)
- The fix is *entry guards*: pick a few relays for your first hop and stick with those.

Clients would route traffic

- [Fixed in Tor 0.1.1.23, 3 Aug 2006]
- Normally the client connects to the first hop and sends a “create” cell to establish a circuit, then sends “extend” relay cells to make further hops.
- Turns out the entry node could send “create” and “extend” cells back to the client!

Pump the network full of Stable / Fast / Guard nodes

- [Fixed in Tor 0.2.0.3-alpha, 27 Jul 2007]
- Tor dir authorities assign Stable flag to the relays with median uptime; Guard to relays with median uptime and median bandwidth.
- So start up 1500 relays with 10 years uptime and 1GB/s bandwidth, and suddenly you bump the Guard status off of all the other relays!

Cross-protocol HTTP form attack

- [Fixed in Tor 0.1.2.16, 2 Aug 2007]
- Tor runs a Control Port so other apps can connect and help configure, display, etc.
- Binds only to localhost. So we're safe!
- But the user runs a browser, and browsers can be induced to do all sorts of things.
- Now use password / cookie auth by default. But how to share passwords between apps?

Exit policies allowed local connect

- [Fixed in Tor 0.1.2.19, 7 Jan 2008]
- The default exit policy refused 127/8, 10/8, 192.168/16, etc etc.
- But you could still reach the public IP address of the relay, from the relay.
- ...which was often a linksys router.

Debian RNG flaw

- [Addressed in Tor 0.2.0.26-rc, 13 May 2008]
- 300 out of ~1500 Tor relay identity keys were bad.
- Logged traffic breakable too--if the client was Debian, *or* if it used only Debian relays!
- Three out of the six v3 dir authority keys were bad. Four would have really sucked.

Infinite length circuits

- [Fixed in, uhm, soon]
- Clients can just keep extending their circuit forever. (Tor relays can't figure out what hop in the path they are.)
- First, this is a DoS multiplier.
- Then, it's an anonymity attack! (See later talk by Christian Grothoff, Nate Evans)

Outline

- Crash course on Tor
- Solved / solvable problems
- ***Tough ongoing issues, practical***
- Tough ongoing issues, research
- Future

Snooping on Exit Relays (1)

- Lots of press last year about people watching traffic coming out of Tor. (Ask your lawyer first...)
- Tor hides your location; it doesn't magically encrypt all traffic on the Internet.
- Though Tor *does* protect from your local network.

Snooping on Exit Relays (2)

- https as a “premium” feature
- Should Tor refuse to handle requests to port 23, 109, 110, 143, etc by default?
- Torflow / setting plaintext pop/imap “traps”
- Need to educate users?
- Active attacks on e.g. gmail cookies?
- Some research on exit traffic properties is legitimate and useful. How to balance?

Who runs the relays? (1)

- At the beginning, you needed to know me to have your relay considered “verified”.
- We've automated much of the “is it broken?” checking.
- Still a tension between having lots of relays and knowing all the relay operators

Who runs the relays? (2)

- What if your exit relay is running Windows and uses the latest anti-virus gadget on all the streams it sees?
- What if your exit relay is in China and you're trying to read BBC?
- What if your exit relay is in China and its ISP is doing an SSL MitM attack on it? (What if China 0wns a CA?)

Who runs the relays? (3)

- What happens if ten Tor relays show up, all on 149.9.0.0/16, which is near DC?
- “EnforceDistinctSubnets” config option to use one node per /16 in your circuit (Tor 0.1.2.1-alpha, 27 August 2006)
- No more than 2 relays on one IP address (Tor 0.2.0.3-alpha, 29 July 2007)
- How about ASes? IXes? Countries?

Tor Browser Bundle traces

- We want to let you use Tor from a USB key without leaving traces on the host
- “WINDOWS/Prefetch” trace
- Windows explorer's “user assist” registry entry
- Vista has many more?

Application-level woes (1)

- Javascript refresh attack
- Cookies, History, browser window size, user-agent, language, http auth, ...
- Mostly problems when you toggle from Tor to non-Tor or back
- Mike Perry's new Torbutton 1.2.0 tackles many of these (30 July 2008)

Some Firefox privacy bugs remain

- No way to configure/spoof timezones
- “Livemarks” / “Live bookmarks” does a lookup over Tor when Firefox starts.
- Client-side SSL certs are messy to isolate (Firefox happily sends them to the remote website even via Tor)
- The TLS ClientHello message in FF2 uses uptime for the “time” variable!

Application-level woes (2)

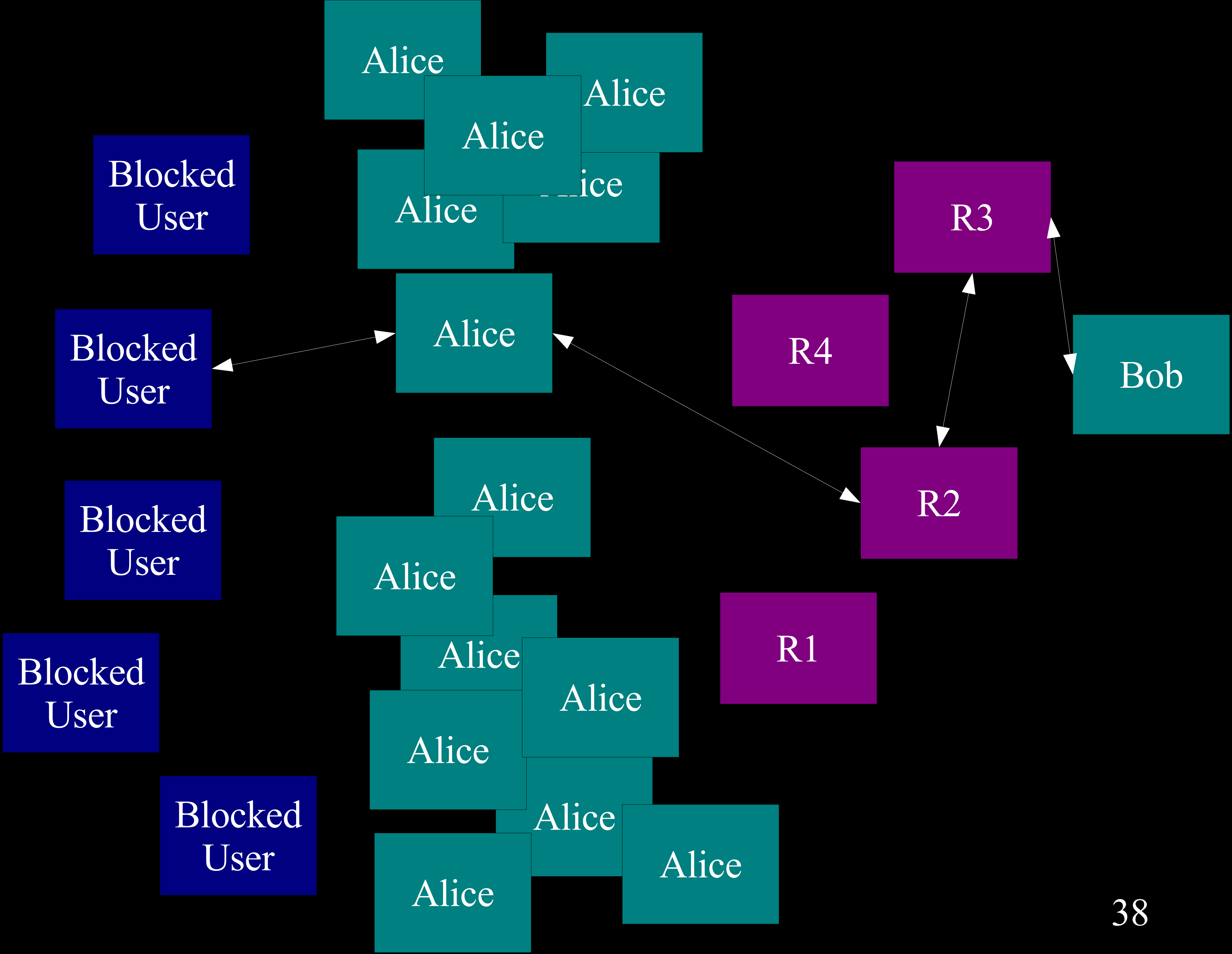
- Some apps are bad at obeying their proxy settings.
- Adobe PDF plugin. Other plugins. Extensions. Especially Windows stuff.

Transparent proxying

- Easy to do in Linux / BSD: iptables/pf, getsockopt()/getsockname(), done.
- Put Tor client in a Linux QEMU running inside Windows. Then intercept outgoing traffic from Windows apps. Or,
- Put Tor client *and* apps inside a Linux QEMU, and launch it from Windows.

Filtering connections to Tor

- By blocking the directory authorities
- By blocking all the relay IP addresses in the directory
- By filtering based on Tor's network fingerprint
- By preventing users from finding the Tor software



Alice

Alice

Alice

~~Alice~~

Alice

Blocked User

Blocked User

Alice

R3

R4

Bob

R2

Blocked User

Alice

Alice

R1

Blocked User

Alice

Alice

Alice

Blocked User

Alice

Alice

Alice

Outline

- Crash course on Tor
- Solved / solvable problems
- Tough ongoing issues, practical
- ***Tough ongoing issues, research***
- Future

Traffic confirmation

- If you can see the flow into Tor and the flow out of Tor, simple math lets you correlate them.
- Defensive dropping (2004)? Adaptive padding (2006)?
- Nick Feamster's AS-level attack (2004), Steven Murdoch's sampled traffic analysis attack (2007).

Website fingerprinting

- If you can see an SSL-encrypted link, you can guess what web page is inside it based on size.
- Does this attack work on Tor? “maybe”
- Considering multiple pages (e.g. via hidden Markov models) would probably make the attack even more effective.

Clogging / Congestion attacks

- Murdoch-Danezis attack (2005) sent constant traffic through every relay, and when Alice made her connection, looked for a traffic bump in three relays.
- Hopper et al (2007) extended this to (maybe) locate Alice based on latency.
- Chakravarty et al (2008) extended this to (maybe) locate Alice via bandwidth tests.

Profiling at exit relays

- Tor reuses the same circuit for 10 minutes before rotating to a new one.
- (It used to be 30 seconds, but that put too much CPU load on the relays.)
- If one of your connections identifies you, then the rest lose too.
- What's the right algorithm for allocating connections to circuits safely?

Declining to extend

- Tor's directory system prevents an attacker from spoofing the whole Tor network.
- But your first hop can still say “sorry, that relay isn't up. Try again.”
- Or your local network can restrict connections so you only reach relays they like.

Outline

- Crash course on Tor
- Solved / solvable problems
- Tough ongoing issues, practical
- Tough ongoing issues, research
- ***Future***

Traffic correlation

- It's just going to get better.
- E.g., maybe somebody publishes mrtg graphs or other apparently innocent data, and that turns out to be enough?
- Smoke ping data for all the relays?

Countries blocking Tor network

- Blocking the website is a great start
- Eventually, they'll block the Tor relays, and bridges will be needed
- Then the arms race for blocking bridge relays will start.
- E.g., Vidalia bridge lookup enumeration bug (fixed in Vidalia 0.1.3, 25 May 2008)

Data retention

- Remember our threat model: even one hop in Germany (Europe? US?) may be too many
- How many layers of logging are there? If your ISP logs, and *its* ISP logs, ...
- How safe are these logs? Who can access them?
- Nothing is really enforced in Germany until 2009, so no need to change technical designs immediately. But that means we need to act!

Last thoughts

- Many of the hard research problems are attacks against all low-latency anonymity systems. Tor is still the best that we know of -- other than not communicating.
- People find things because of the openness and thoroughness of our design, spec, and code. We'd love to hear from you.
- Pretty much any Tor bug seems to turn into an anonymity attack.