Everything About Tor

Tom Ritter

v1.6 - 5/18/2015
Source: https://ritter.vg/p/tor.key
Latest: https://ritter.vg/p/tor-vlatest.pdf

What is Tor (Browser)?

- Makes you Anonymous to services you visit
  - the originating IP is not yours
  - prevents sites from correlating your browsing
- Prevents your network from seeing services you access
  - Bypasses network censorship at local, ISP, or national level

little-t tor

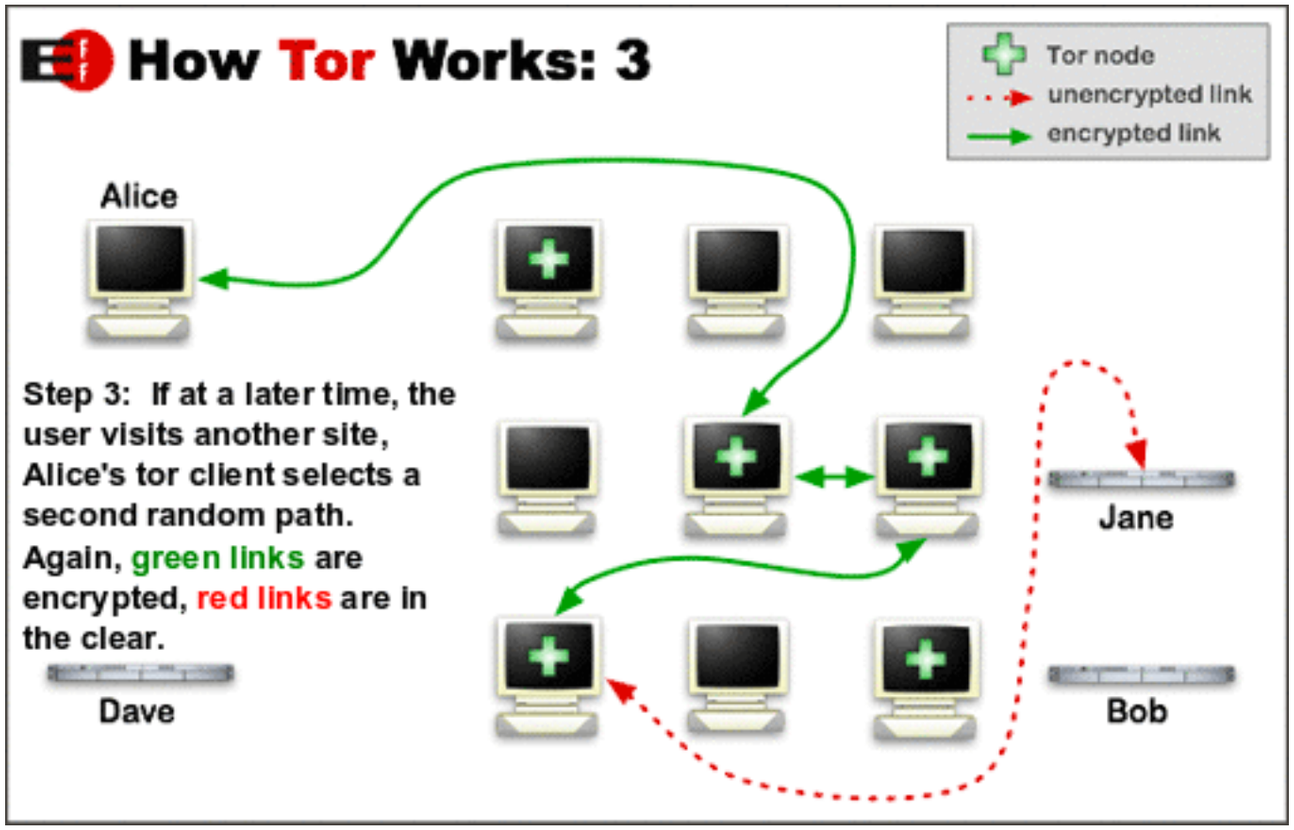
- Network Daemon
- Operates at the TCP Layer
- Presents a SOCKs proxy
- Transports any TCP Protocol

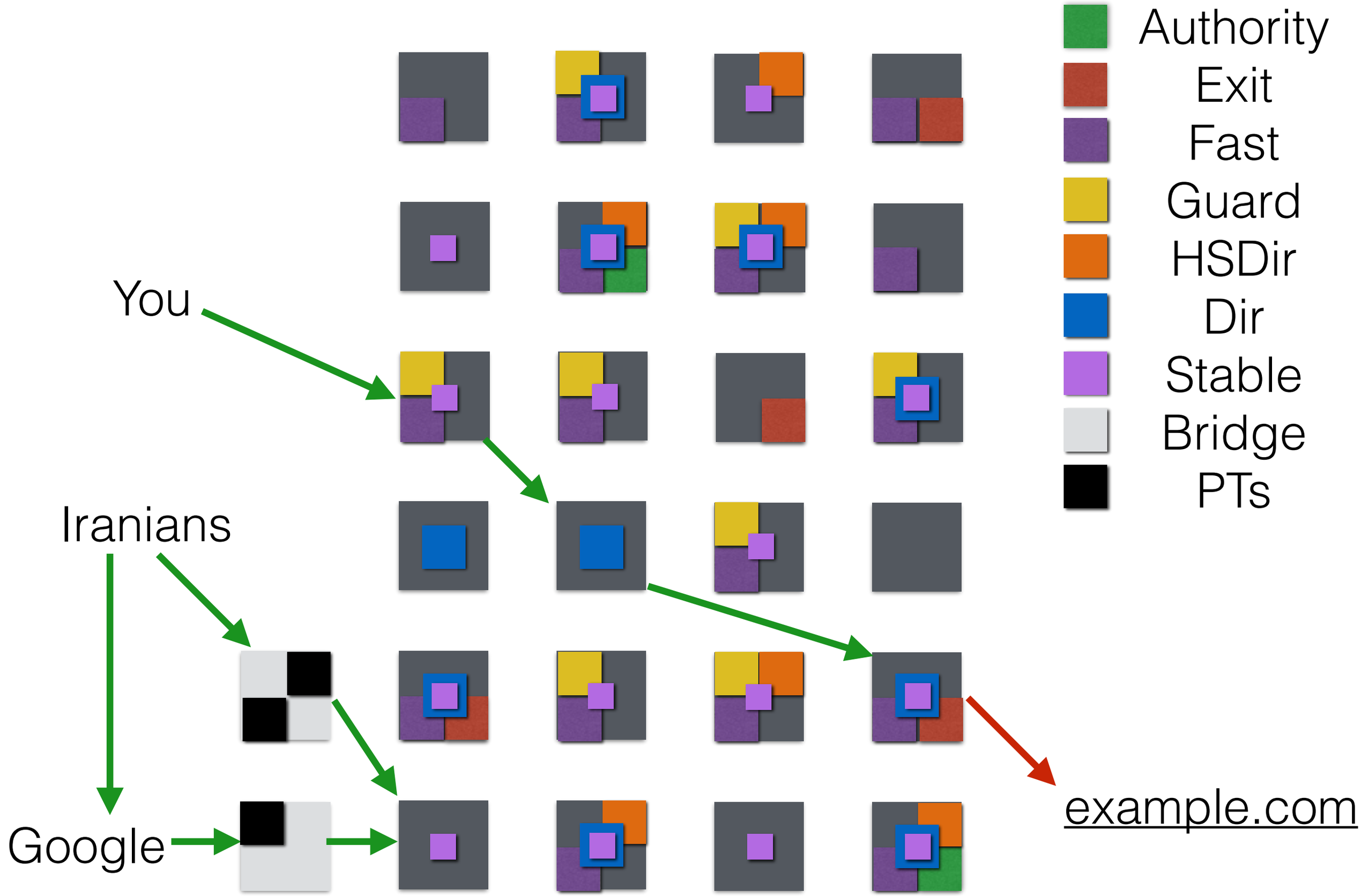
Tor Browser

- Modified Version of Firefox
- Patched to prevent third-party tracking
- Bundles tor, automatically routes through it

Tor… Stuff

- HTTPS Everywhere
- Orbot (Tor on Android)
- Tails (LiveCD)
- Tor2Web (access Hidden Services w/o Tor)
- Torsocks (automatically torify an app)
- TorBirdy (Thunderbird Add-On)
- Shadow / Chutney / TorPS (tor network simulator)
- TorBEL / ExoneraTor (exit node detectors)
- arm / stem (tor controller & library)
- Metrics / Atlas / Globe / Compass / Onionoo / DocTor / DepicTor (statistics and network info)
- GetTor (sends you tor if torproject.org is blocked)
- Weather (notify you if your node is down)
- TorFlow (network health measurements)
- OONI
- Mirrors


How Tor Works: 3
Tor node
unencrypted link
encrypted link
Alice
Step 3: If at a later time, the user visits another site, Alice's tor client selects a second random path. Again, green links are encrypted, red links are in the clear.
Dave
Jane
Bob


Authority
Exit
Fast
Guard
HSDir
Dir
Stable
Bridge
PTs
You
Iranians
Google
example.com

1000 Foot Overview

3 Hops

- I talk to Node A
- Node A talks to Node B
- Node B talks to Node C
- Node C talks to example.com

Node Types

- **Directory Authority** - Special, trusted nodes that run the network

- **Guard** Node - Type of relay used for entry to the network. It ‘guards’ the network for you

- Middle Node - Type of Relay used for middle hop

- **Exit** Node - Special relay that allows talking to the public internet

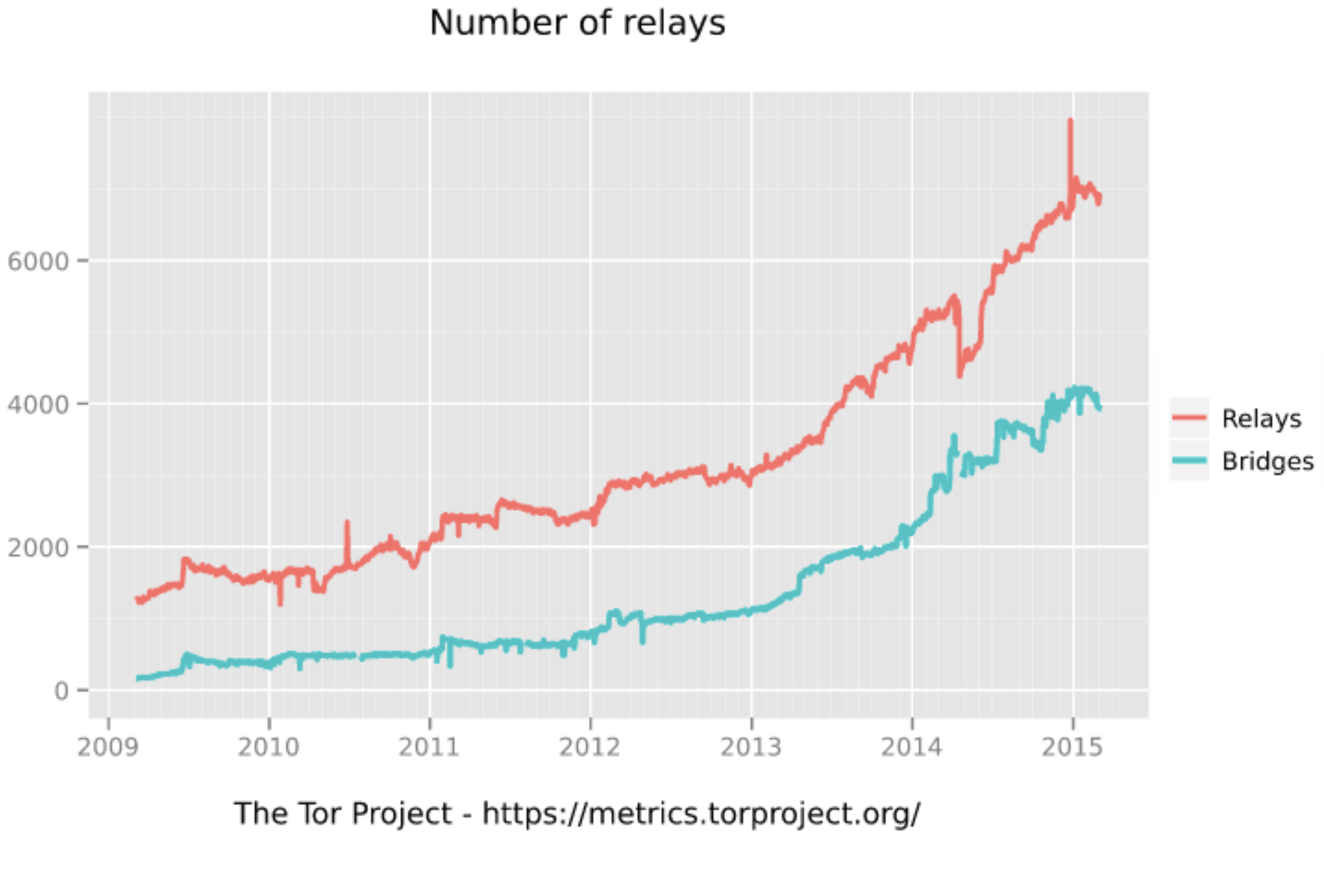
The ‘Tor Network’ vs a ‘tor network’

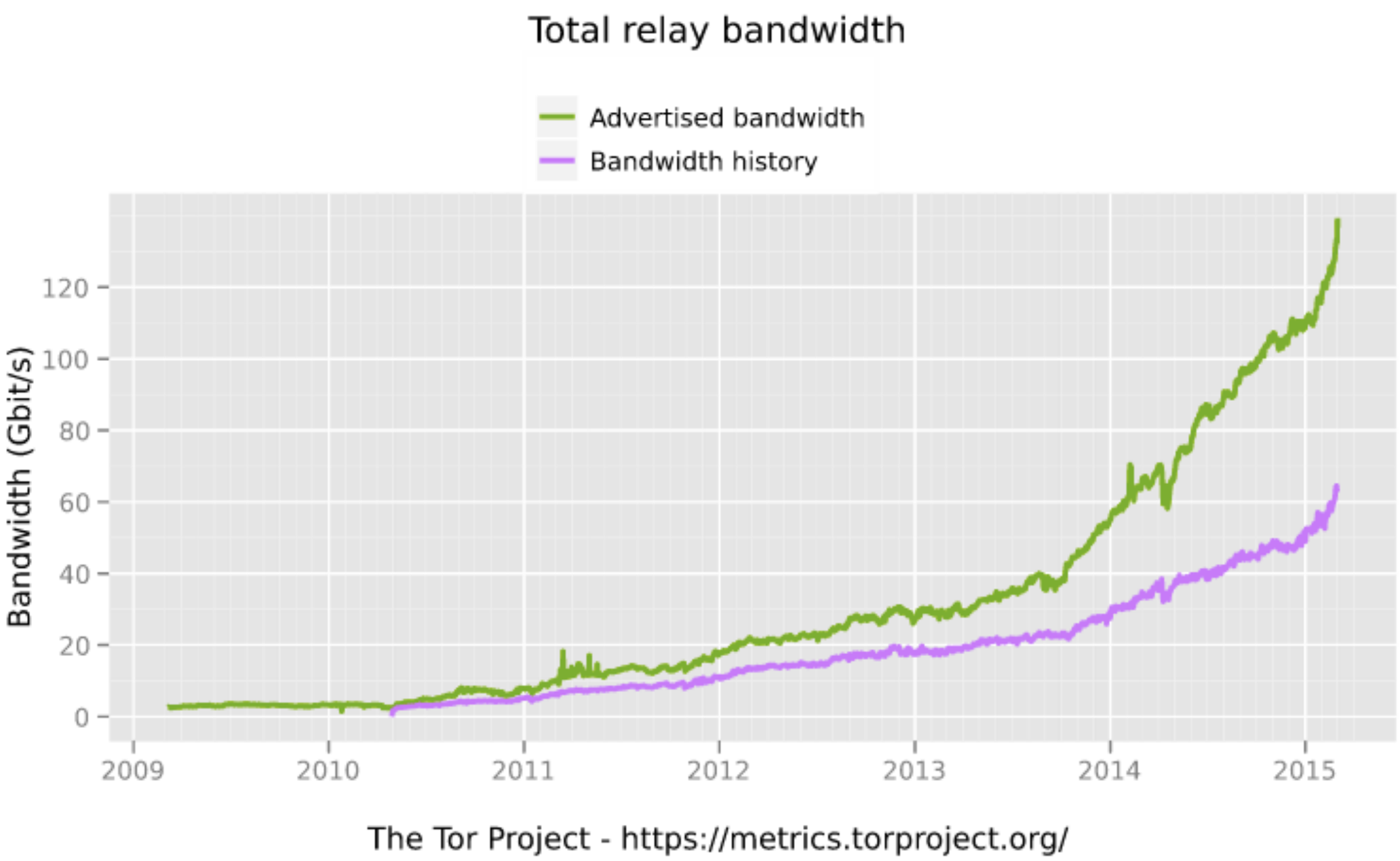
- The Tor Network is the collection of ~6500 relays that operate that you can use.
- tor is open source - you can run your own network.
- IronKey used to run their own network!

Common Attacks

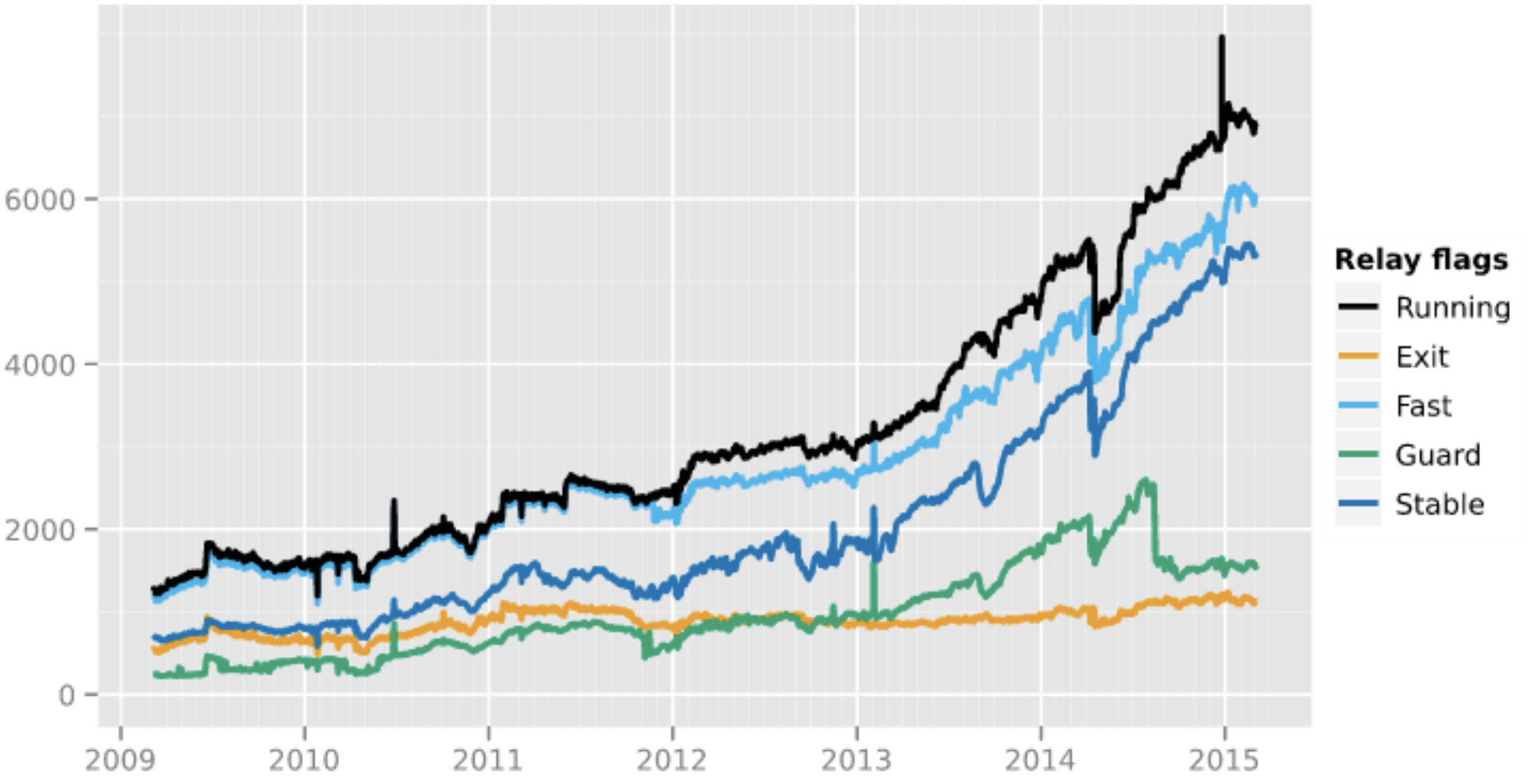
- “What if I run an exit node and log/sslstrip?”
  - Yup, you can do that
- “What if I serve a Firefox exploit to TBB?”
  - You can do that too
- “What if I’m the NSA and I record ~all traffic on the internet and correlate traffic flows?”
  - Well… kinda….

The network is growing!



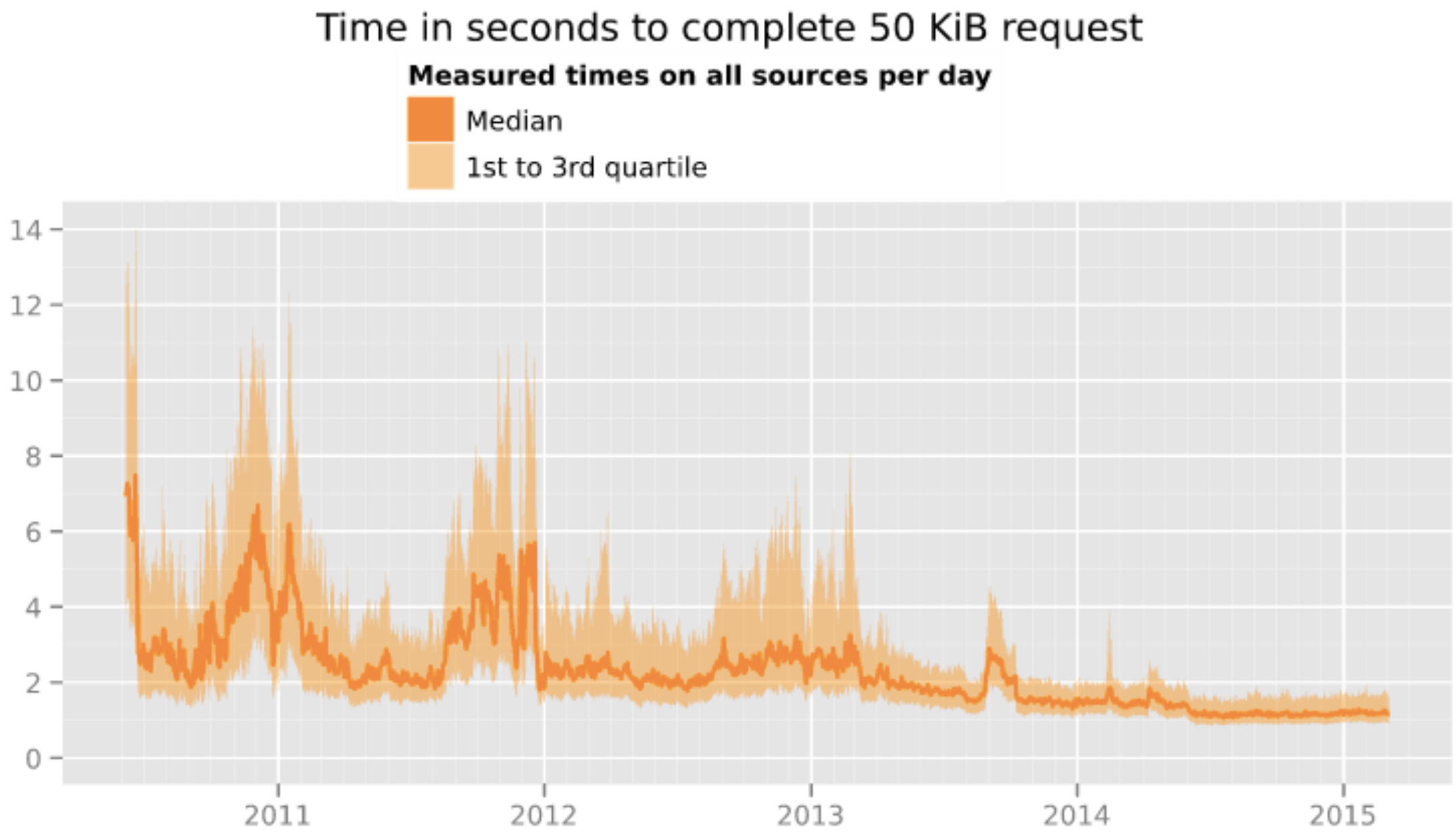

Total relay bandwidth
Advertised bandwidth
Bandwidth history
Bandwidth (Gbit/s)
120
100
80
60
40
20
0
2009
2010
2011
2012
2013
2014
2015
The Tor Project - https://metrics.torproject.org/

Number of relays with relay flags assigned

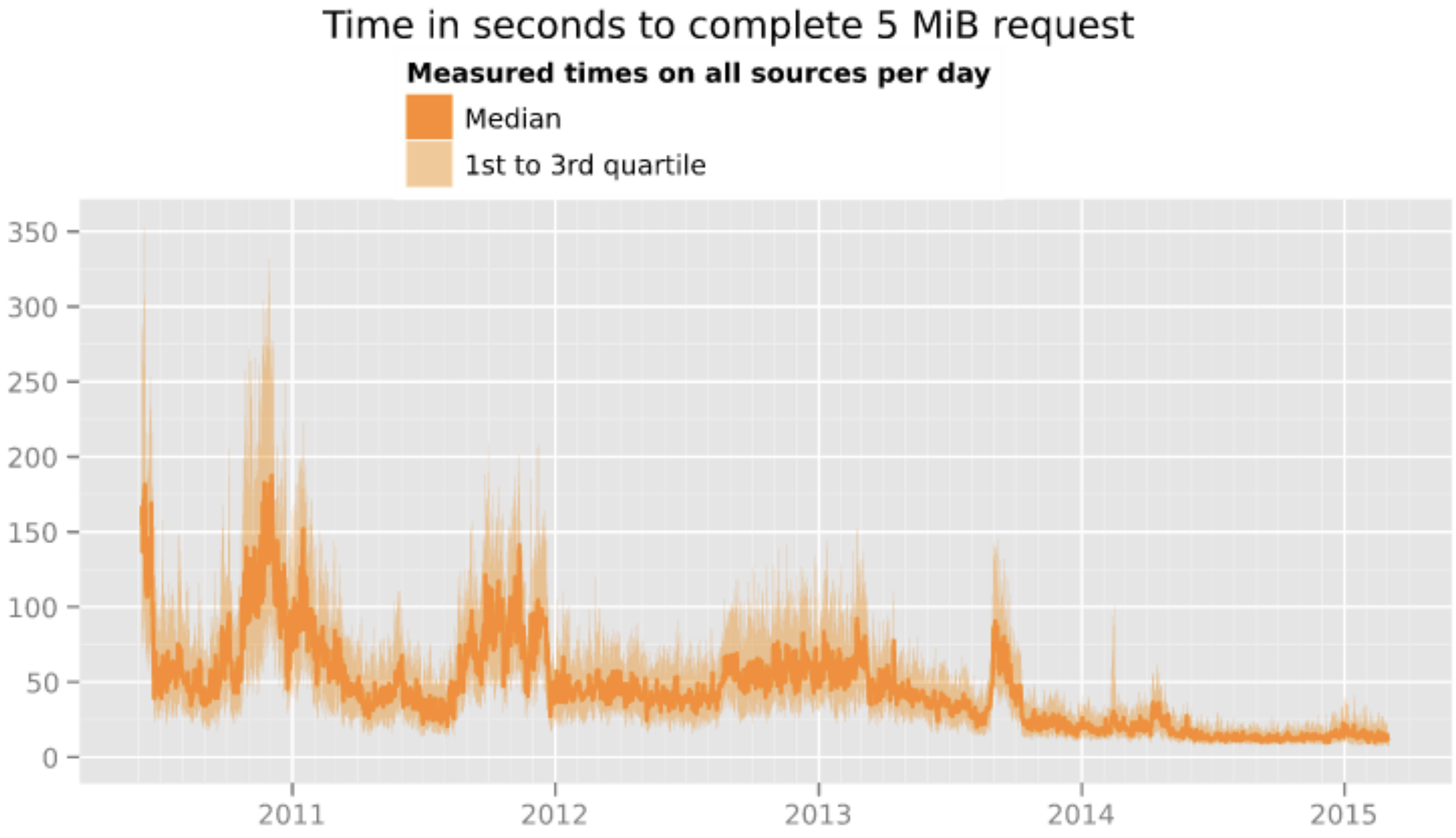


The Tor Project - https://metrics.torproject.org/

The network is speeding up!



The network is speeding up!



The Consensus & Directory Authorities

Directory Authorities

- Every hour the authorities perform a majority vote on the **consensus**

- Consensus is the snapshot of the network as it exists currently

- Contains the nodes in the network and related info

Consensus

- consensus method
- valid times
- acceptable server and client versions
- customizable params inc. bw weight tuning
- list of relays and their:
  - IP, port, key
  - flags, version
  - exit policy
  - bw weight

Tor's Authorities

- Calculate consensus every hour, valid for 24 hours
- 9 Authorities
- Running continuously for 12-13 years, no downtime

Authority Operators

- **maatuska** - Linus Nordberg, **Tor Project Volunteer**
- **tor26** - Peter Palfrader, **Tor Project Volunteer**
- **urras** - Jake Appelbaum, **Tor Project**
- **longclaw** - **RiseUp**
- **dizum** - Alex de Joode, **Old School Cypherpunk**
- **gabelmoo** - Sebastian Hahn, **Tor Project Volunteer**
- **moria1** - Roger Dingledine, **Tor Project**
- **dannenberg** - **CCC.de**
- **Faravahar** - Sina Rabbani, **Tor Project Volunteer**

Colors indicate rough level of entanglement with legal Tor Project organization
Red: paid employee or sysadmin, Orange: volunteer but close working relationship, Green: minimal intricacy

Consensus Algorithm

- Collect all the data
- Produce a vote on what params should be, relays included, flags
- Post vote to all authorities
- Fetch vote for any missing authority ***from all authorities***
- Determine majority of each parameter/flag/relay individually
- Sign it. Post your signature to each authority
- Fetch the signature for any missing authority from all authorities
- Everyone should agree, and we should have a majority
- Publish it

Consensus Algorithm

- We need a quorum of >1/2 of all Authorities to produce a consensus
    - But majority on any item is majority of people voting
- Some items (flags) are not voted upon by everyone
- Consensus algorithm itself determined by 2/3 vote

“from all authorities”

- This is what prevents a network adversary with access to one authority from fragmenting consensus
- You can stop urras from talking to half the authorities and let the other half through - but the blocked half will still learn its vote.

(Adversary with access to all authorities can just blackhole a majority of them)

Adding an Authority

- Manual upgrade of majority of DirAuths at ~same time
- Done recently, added longclaw
- Little risky, at time-of-upgrade only made consensus by single vote

Authority Keys

- (Authority) **Identity Key** - long-term key, kept offline
- **Authority** (Signing) **Key** - online key used to sign consensus
- **Relay** (Identity) **Key** - online key used to act as a relay
- The Identity Key is hardcoded in Tor Client
- Authority Key Certificates are published by DirAuths and downloaded by everyone to establish trust in them
- Small support for legacy keys, used to keep old clients going

Flags

Authority

BadExit

Exit

Fast

Guard

HSDir

Running

Stable

V2Dir

Valid

Flags

Authority	hardcoded
BadExit	managed by 3 auths, hardcoded
Exit	allows exit to 2 of [80,443,6667] to at least /8
Fast	in top 7/8ths by bandwidth, or >100kb/s
Guard	Fast, known for 8+ Days, 98% Reliability

Flags

HSDir	running > 96 hours
Running	if we can talk to it
Stable	top 50% of routers or MTBF > 5 days
V2Dir	Mirrors Consensus
Valid	not blacklisted and not running old version

Consensus Parameters

- ‘Switches’ and ‘Dials’ for the entire network
  - Enable or Disable a feature
  - Tune bandwidth settings
  - Adjust guard settings

Consensus Parameters

- CircuitPriorityHalflifeMsec=30000
- NumDirectoryGuards=3
- NumEntryGuards=1
- NumNTorsPerTAP=100
- Support022HiddenServices=0
- UseNTorHandshake=1
- UseOptimisticData=1
- bwauthpid=1
- cbttestfreq=1000
- pb_disablepct=0
- usecreatefast=0

Misc (Part 1)

- DirAuths only allow 2 relays per IP
- Consensus has three times:
  - When it starts being valid X
  - When it stops being ‘fresh’ X+1 hours
  - When it stops being valid X+24 hours
  - (Consensus contains a ‘valid-until’ field that is misleading)

Misc (Part 2)

- BadExits are reported to Tor Project, confirmed, and blacklisted
  - Several volunteers scan for Bad Exits
- Several values will soon be included in consensus to provide legitimacy:
  - Hashes of prior consensuses (to detect attacks)
  - Hashes of Tor Browser

Descriptors

Descriptor Operation

- Relays generate and upload their descriptor and extra-info
- Clients download a descriptor for every relay in the network
- Extra Info is used by Tor Project's backend stuff to calculate metrics and related
    - Ordinarily this would be 'secret', but Tor makes it public. You can easily download ExtraInfo Descriptors

(Relay) Descriptor

- nickname, platform, contact
- address, port
- publish_date
- estimated bandwidth
- identity key and fingerprint
- onion key
- family
- ntor onion key
- hibernating
- uptime
- Exit Policy
- IPv6 Exit Policy
- Signature
- caches-extra-info
- has-extra-info
- is HSDir
- allow-single-hop-exits

Extra Info

- geoip database digest
- bridge usage / directory requests / entry IPs (unique IPs) by country
- bridge usage (unique IPs) by v4, v6
- bridge usage (unique IPs) by PT
- directory requests (# requests) by country
- directory request response codes count
- directory request download statistics, bandwidth usage
- cell statistics, queue times
- uni- and bi- directional connection stats
- exit bw and stream counts per port
- pluggable transports supported
- signature
- *(All usage stats mod 8)*

Micro Descriptors

- Stripped-down Descriptor generated about a relay, by a DirAuth
- Aim to be valid for ~a week
- onion key & nor key, address, exit ports, identity digest
- Omits exact exit policy and identity key

URLs

- GET http://ip:port/tor/status-vote/current/authority
- GET http://ip:port/tor/status-vote/next/authority.z
- POST http://ip:port/tor/post/vote

Micro Descriptor Consensus

- Clients actually download a Micro Descriptor Consensus and use Micro Descriptors
- Normal Consensus, but contains Micro Descriptor hashes

Directory Caches (V2Dir Flag)

These Mirror the Authorities

- These, like normal relays, retrieve new consensus after their current one is not fresh
- Everyone fetches it at random inside a window to avoid swarming DirAuths
    - Directory Mirrors fetch it more aggressively
- V2Dir Flag: Means they provide a mirror for Consensus, Descriptors, Micro-Descriptors

Relay Startup

Consensus Fetch

- Cold Startup
    - Download a consensus from a DirAuth
- Warm Fetch (you have a current consensus)
    - Download a consensus from a Directory Cache
- Fetch a new consensus in a random interval after it stops being ‘fresh’

Descriptor Fetch

- Client has descriptor for every relay
- Downloads them from several Directory Caches
- New clients use Micro Descriptor Consensus, in which case it downloads those descriptors

Relay Keys

- Identity Key - signs documents and other keys
- Onion Key - decrypts incoming connection requests. Lives about a week, present in descriptor
- Connection Key - TLS Connection key. Rotates at least every day.

Connections

- Each relay will eventually open and hold open a TLS connection to every other relay in the network.
  - Sends Keep-Alives over these (app-layer, not Heartbeat)
  - New circuits will be routed over existing connections
- Does this scale? Not really….

Link Protocol

Terms

- internal circuit - final node is chosen like a middle node
- exit circuit - final node is an Exit node
- clean circuit - one that has not been used for traffic yet
- fast, stable - circuits where each node has these flags
- **Family** - relays operated by the same admin (opt-in)

Predicting Connections

- Tor remembers the ports you've used for the last hour
- Uses this to construct two fast clean exit circuits for each port (limit of 12)
- one clean fast exit circuit for port 80
- two clean fast stable internal circuits

Long-Lived Ports

- Long Lived Ports: FTP, SSH, IRC, SILC, MSNP, MMCC, ICQ, XMPP, Gobby, 8300

- Requests to a Long-Lived Port will result in only ‘Stable’ circuits.

Crypto

- AES-128 in CTR mode, Counter starts at 0
- RSA-1024, OAEP[SHA-1]
- ntor ECC in curve 25519
- DH in 1024 bit group
- SHA-1

More Crypto

‘Hybrid Encryption’ for a byte sequence M:

- If M fits in a Public Key ciphertext, do so
- Else:
  - Generate a key K
  - M1 = the first 70 bytes of M, M2 the remaining
  - Encrypt K|M1 with the Public Key, M2 with K
  - No MAC, allows truncation & bit flipping in M2

TLS Layer

- Guaranteed (EC)DHE, and disallow resumption
- Three Flavors, all supported, #3 preferred
  - Two-cert chain in TLS handshake
  - One cert handshake then two-cert renegotiation
  - ‘In-protocol’ where certs are handled in link protocol

TLS Layer

- Two Certs Method
  - Three Ciphersuites: DHE CBC w/ 3DES or AES-[128/256]
  - C -> S: [Some Key, Identity Key]
    - S -> C: [TLS Key, Identity Key]
- Renegotiation method indicated by including any additional ciphersuite
  - C -> S: Hi
  - S -> C: [TLS Key]
  - C -> S: <Renegotiate>, Continue as above w/ two certs each

TLS Layer Ugliness

- There is a ‘fixed cipher suite list’ that Tor clients can send even if they don’t support all those ciphers
  - Client is then only guaranteed to support 3DES / AES
- Client detects if server supports in-protocol method by looking at random certificate attributes:
  - subjectName == issuerName, commonName is not .net, PubKey Modulus != 1024 bits

Link Protocol

- Four versions, tied closely with TLS Layer negotiation type
- Cells
  - 512 Bytes Long (514 in latest version)
  - TLV Style: Circuit ID, Type, Length, Payload
  - Remaining bytes are padded

Cell Types

- Padding / VPadding *
- ~~Create / Created~~
- Create2 / Created2
- ~~Create_Fast / Created_Fast~~
- Relay / Relay_Early
- Destroy
- NetInfo
- Versions *
- Certs *
- Auth_Challenge *
- Authenticate *
- Authorize *

* - variable length cell

Cell Handshake (v3&4)

- VERSIONS to establish a link protocol version
- S -> C: CERTS: [TLS Key, ID Key] Signed by ID
- S -> C: AUTH_CHALLENGE: [Challenge bytes to sign]
- C -> S: Optional CERTS & AUTHENTICATE
  - CERTS: [Auth Key, ID Key] Signed by ID
  - AUTHENTICATE: Signature over TLS Conn Data & all cells
- NETINFO to confirm addresses and timestamps

Circuit Creation

- Sends a CREATE2 cell: Handshake Type & Data
- Two Handshake Types: TAP & ntor
  - TAP is old and slow
  - ntor is new and fast
  - If node chosen for circuit supports ntor, use that

TAP Handshake

- Old-style handshake
- Standard DH-1024:
  - C -> S: g^x - ‘hybrid encrypted’ to onion key
  - S -> C: g^y, KDF output for sanity check
- Pub-Key Decryption + DH Handshake: slow

ntor Handshake

- new hotness: curve25519
- Not standard ECDH, much more
    - C -> S: public key, other stuff
    - S -> C: public key, auth value
- 2 curve25519 ops, 3 HMACs: faster

RELAY Cells

- After handshake, RELAY cells are sent with their own subtypes

- BEGIN / END
- DATA
- CONNECTED
- SENDME
- EXTEND / EXTENDED
- EXTEND2 / EXTENDED2
- TRUNCATE / TRUNCATED
- DROP
- RESOLVE / RESOLVED
- BEGIN_DIR

Circuit Extension

- C -> S1: RELAY_EXTEND2 [addr, create2 data]
  - extend2 is actually only supported > 0.2.4.8
  - extend is used for TAP handshakes, but it can be coerced to support ntor through some hacks to enable you to extend an ntor handshake from a tap node
- S1 -> S2: CREATE2 [data from client]
- S2 -> S1: CREATED2 [data]
- S1 -> C: RELAY_EXTENDED2 [data]

Application Data

- Uses Relay Cells
- This is where the Crypto Happens!
  - C  S1  S2  S3 Internet

DNS Lookup

- Possible to create a circuit for the purpose of resolving DNS
    - tor-resolve tool does this
- Ordinarily, DNS is resolved by the exit node during a connection

Application Connection

- Client (C) talks to the Exit node (S)
- C -> S: RELAY_BEGIN: address/hostname & port
- S -> C: RELAY_CONNECTED: address, TTL
- C <-> S: RELAY_DATA: application data
  - Can optimistically send DATA before CONNECTED
- RELAY_END when done or error

Other Cell Types

- CREATE[D]_FAST - avoided the TAP handshake for first hops, but weakened security. Only used in cold-start situations where we have no onion key

- RELAY_EARLY - We don't actually send EXTEND commands in a RELAY cell, we use RELAY_EARLY. If a node sees more than 8 RELAY_EARLY cells, it assumes you're trying to make an infinite circuit and kills your circuit.

- DESTROY - tears down a circuit from error or all streams done

- AUTHORIZE - Unused, but reserved for future anti-scanning

Other Relay Cell Types

- TRUNCATE[D] - Client asking S1 to send a DESTROY to S2, and it being acknowledged
- SENDME - Used to adjust cell window sizes
- DROP - Long-range padding cells
- BEGIN_DIR - Basically BEGIN, but to the node's own Directory Cache

Padding

- Used for KeepAlives currently
- Tor does not use any padding strategies
  - Unclear how well any of them work
  - Uses up bandwidth

Path Selection

Constraints

- No relay in a path twice
- Only one family member in a path
- Only one router in any /16
- Guard/Exit must have Valid flag.
  - Invalid allowed for "middle" and "rendezvous"

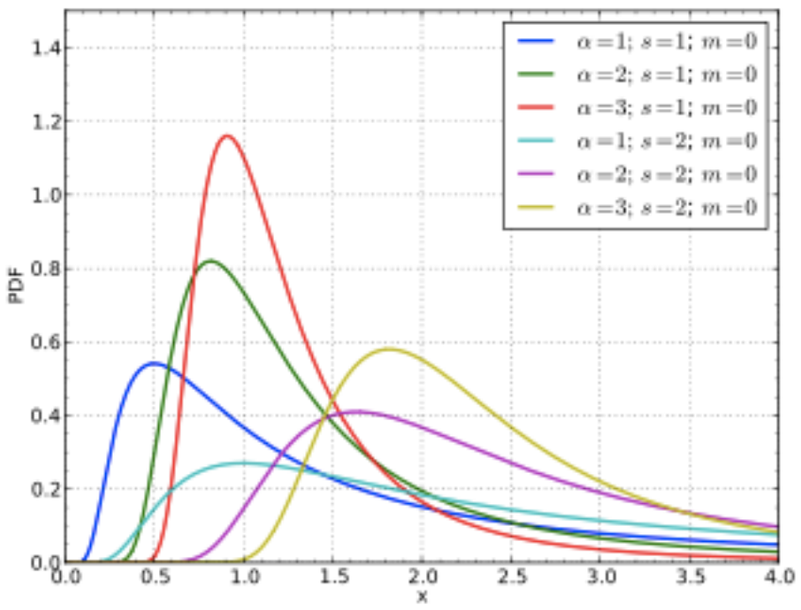
.exit

- Silly method to allow exiting via a specific node at request time (specified by fingerprint or ‘name’)
- Disabled by default

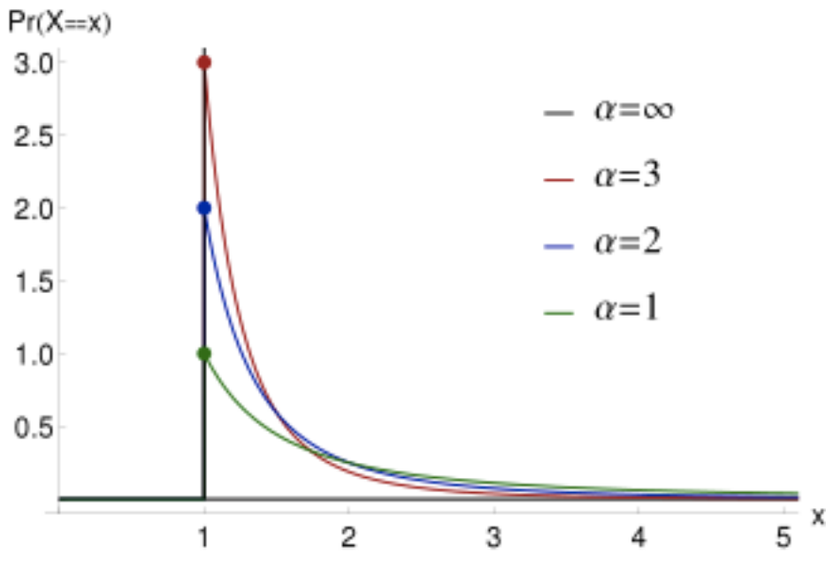
- http://ritter.vg.C0EDB08D7540D1DD3CA69809ED17D979F51B66E3.exit
- http://ritter.vg.nodename.exit

Circuit Timeouts

- Record Circuit Build Times to enable timeouts based on personal network connectivity
- Prime cache w/ 100 test circuits
  - One every 100 seconds
- 50ms binning,1000 entries
- Timeout if build time fits into the 20% slowest
- Also detects network loss



Frechet Distribution



Pareto Distribution

Guards

- On startup, tor chooses a Guard from the consensus. You use this guard for 2-3 months.

- Used to be 3 guards, was recently switched to a single guard. Soon it will also up Guard lifetime to 9 months

Guards - Math

- Attacker controls C out of N relays
- Choose an entry and exit at random
- You choose attacker relays with probability (C/N)^2
- If attacker runs 100 relays out of 5000, you hit their combo with probability 50% after 1250 streams

Guards - Math

- Attacker controls C out of N relays
- Choose a fixed entry and an exit at random
- You choose attacker entry with probability C/N, and we assume you will hit an attacker exit
- You get a 2% chance of being profiled

Attacks on Guards

- Enable attacker to fingerprint you if you move networks (Easy w/ 3 Guards, harder w/ 1)

- Blocking access to your guard(s), causing you to pick new ones

  - Not a really great solution here, but being discussed

Attacks by Guards

- Standard attempts at end-to-end correlation, but also:
- Path Bias - Malicious Guard makes connecting to a non-colluding Exit shitty.
  - Countermeasure: Detect the build-success ratio and the usage-success ratio for each Guard
  - Currently only warns, does not enforce, due to CPU overload on relays causing non-malicious failures

Bandwidth Scanner Specification

"This is Fail City and sqlalchemy is running for mayor"

- or -

How to Understand What The Heck the Tor Bandwidth Scanners are Doing

At a high level, the Bandwidth Scanner

- Calculates values for the Consensus
  - *(per relay:)*
    - `r rittervg …`
    - `w Bandwidth=410`
  - *(skip to the end:)*
    - `bandwidth-weights Wgg=6157 Wgm=615 ...`

- Does this by scanning relays to estimate speed, making circuits through like-speeded relays

Bandwidth Scanner Purpose

- Balance load across the network such that a user can expect to have the same average stream capacity regardless of path

- Can be consider a proportional-integral-derivative controller (PID controller)

- F_node is the stream capacity through a node

- F_Avg is the average of all F_nodes

- Current deviation from ideal is P, Past deviations is the Integral, Prediction of future error is the Derivative

- We adjust the weight of a node based off it’s current value

Bandwidth Weighting

- Wgg - Guard nodes in guard position
- Wgm - unflagged nodes in guard Position
- Wgd - Guard+Exit nodes in guard Position

- Wmg - Guard nodes in middle Position
- Wmm - unflagged nodes in middle Position
- Wme - Exit nodes in middle Position
- Wmd - Guard+Exit nodes in middle Position

- Weg - Guard flagged nodes in exit Position
- Wem - unflagged nodes in exit Position
- Wee - Exit nodes in exit Position
- Wed - Guard+Exit nodes in exit Position

- Wgb - BEGIN_DIR-supporting Guard nodes
- Wmb - BEGIN_DIR-supporting unflagged nodes
- Web - BEGIN_DIR-supporting Exit nodes
- Wdb - BEGIN_DIR-supporting Guard+Exit nodes

- Wbg - Guard nodes for BEGIN_DIR requests
- Wbm - unflagged nodes for BEGIN_DIR requests
- Wbe - Exit nodes for BEGIN_DIR requests
- Wbd - Guard+Exit nodes for BEGIN_DIR requests

G = total bandwidth for Guard nodes.
M = total bandwidth for non-flagged nodes.
E = total bandwidth for Exit nodes.
D = total bandwidth for Guard+Exit nodes.
T = G+M+E+D

Wgg*G + Wgd*D == M + Wmd*D + Wme*E + Wmg*G
Wgg*G + Wgd*D == Wee*E + Wed*D
Wed*D + Wmd*D + Wgd*D == D
Wmg*G + Wgg*G == G
Wme*E + Wee*E == E

How Clients Use Weighting

- Consensus
  - *(per relay:)*
    - r rittervg …
    - w Bandwidth=410
  - *(skip to the end)*
    - bandwidth-weights ... Wgg=6157 Wgm=6157 ...

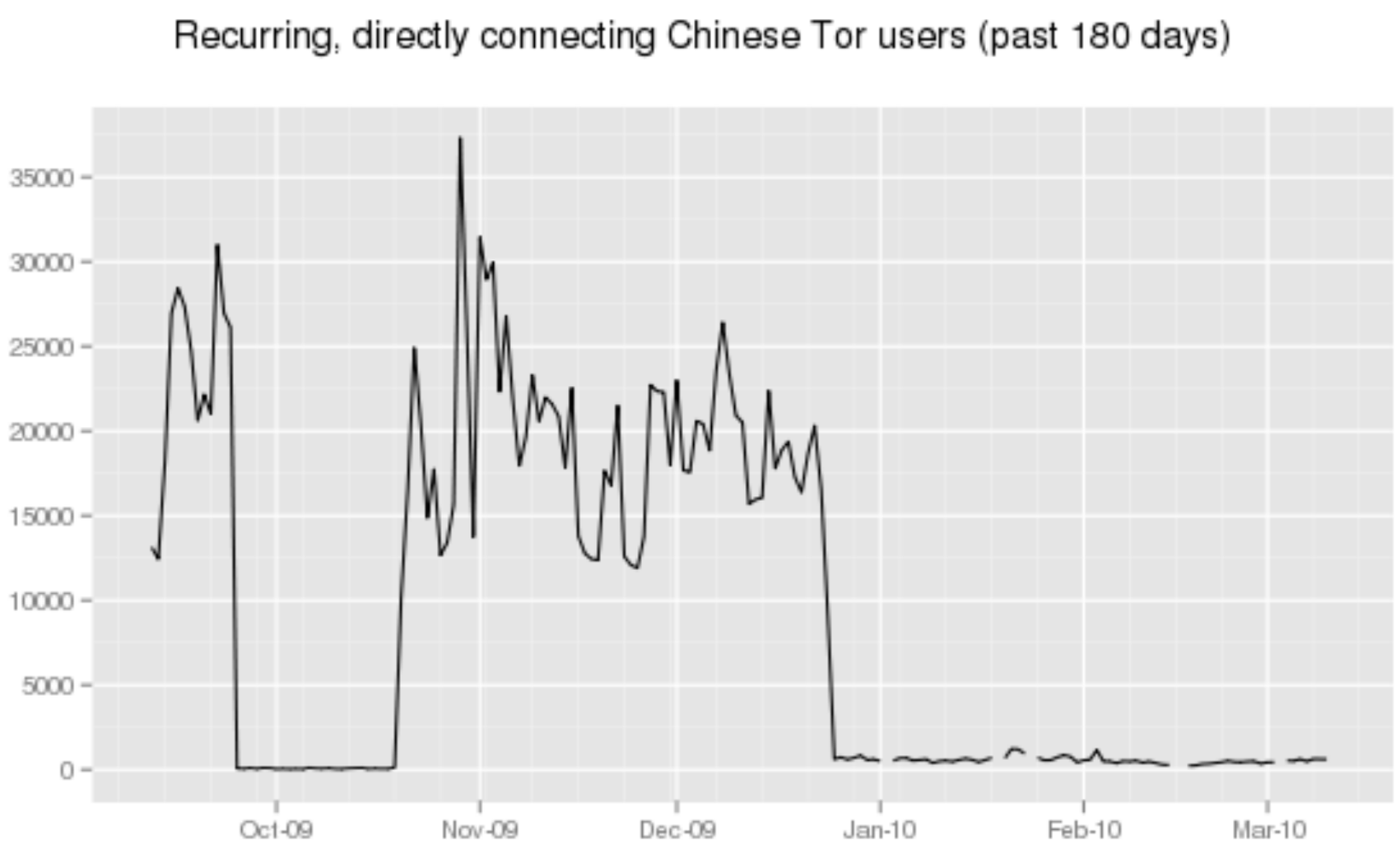
- Go through all the nodes:

  - this_weight = weight in the consensus * applicable modifiers for purpose it's being used

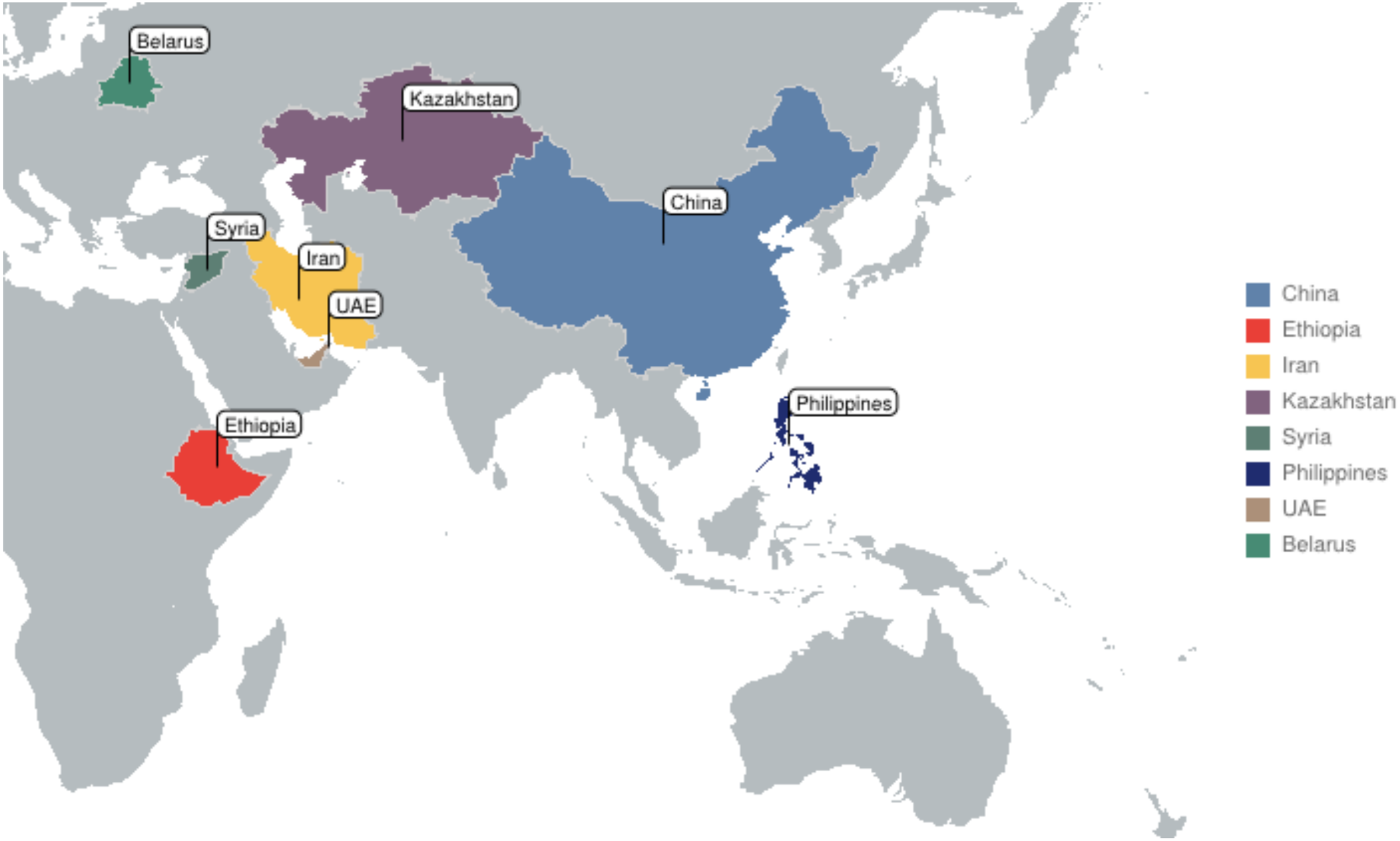
- Choose a node with probability proportional to this_weight

Bridges

Censorship of Tor



Censorship of Tor



Censorship Types

- Blocking Public IP Addresses from Consensus
- Blocking torproject.org
- Matching hardcoded TLS handshake strings, certificate attributes, etc

Bridges

- Unlisted Tor entrance nodes

- Automatically* published to Tor Network, but unlisted in consensus

Censorship Timeline…

- 2006 Thailand - DNS Redirection of torproject.org
- 2007 Saudi Arabia, Iran - Smartfilter / Websense rules blocking /tor/
- Feb 2012 Kazakhstan - DPI on Server Hello
- May 2012 Ethiopia - DPI on Server Hello
- June 2012 Philippines - DPI on Ciphersuites
- June 2012 UAE - DPI
- Dec, 2012 Syria - DPI
- Mar 2014 Turkey - torproject.org blocked

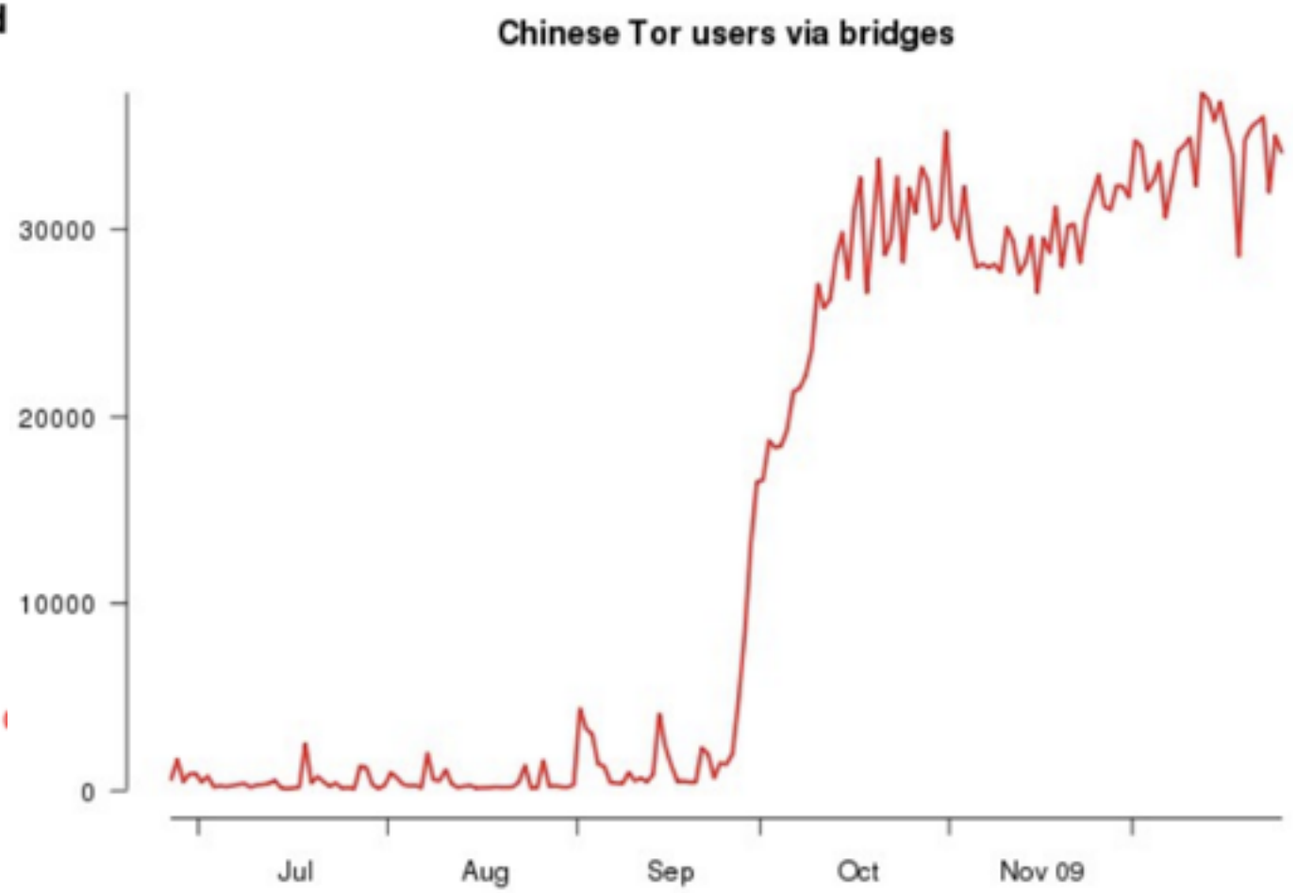
Censorship Timeline… (Iran)

- 2007 Saudi Arabia, Iran - Smartfilter / Websense rules blocking /tor/
- Jan 2011 Iran - DPI on SSL DH parameter
- Sept 2011 Iran - DPI on SSL certificate lifetime
- Oct 2011 Iran - Throttle all SSL
- Feb 2012 Iran - (Ineffective) DPI on SSL handshake
- Oct 2012 Iran - DPI on TLS for Client Key Exchange
- 2013 Iran - TCP Reset anything that isn’t HTTP
- Mar 2013 Iran - DPI on SSL certificate lifetime
- Jul 2014 Iran - IP block Directory Authorities

Censorship Timeline… (China)

- 2008 China - Block torproject.org
- Sept 2009 China - Block all public tor IPs
- Mar 2010 China - IP Block popular Bridges
- Oct 2011 China - Begin active probing of bridges after seeing a suspected handshake
- Mar 2013 China - Begin active probing of obfs2 bridges
- Feb 2015 China - Default obfs4 bridges (in public sources) blocked

China's Initial IP Blocks



Arms Race

- Make Bridges (actually did this ahead of time)
- Perform DPI -> Reduce Fingerprint (tor), ObfsProxy
- Probe Bridges -> Pluggable Transports w/ Key
- More: http://eecs.berkeley.edu/~sa499/tor_timeline.pdf

Bridge Distribution

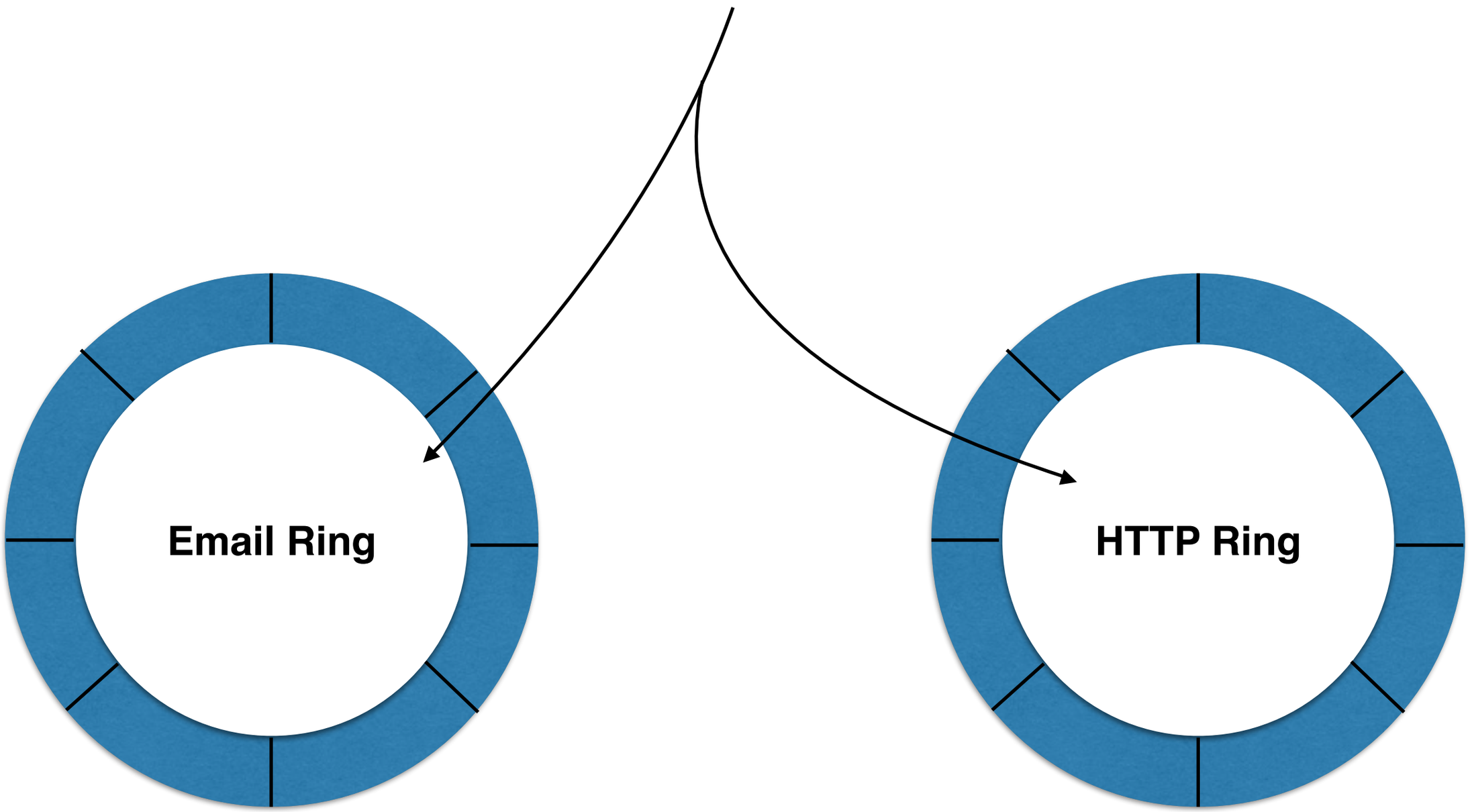
- Auto-Published Bridges
  - hardcoded in Tor Browser
    - bridges.torproject.org
    - bridges@bridges.torproject.org
- ‘Secret’ Bridges
  - Passed by organizations

One More Authority

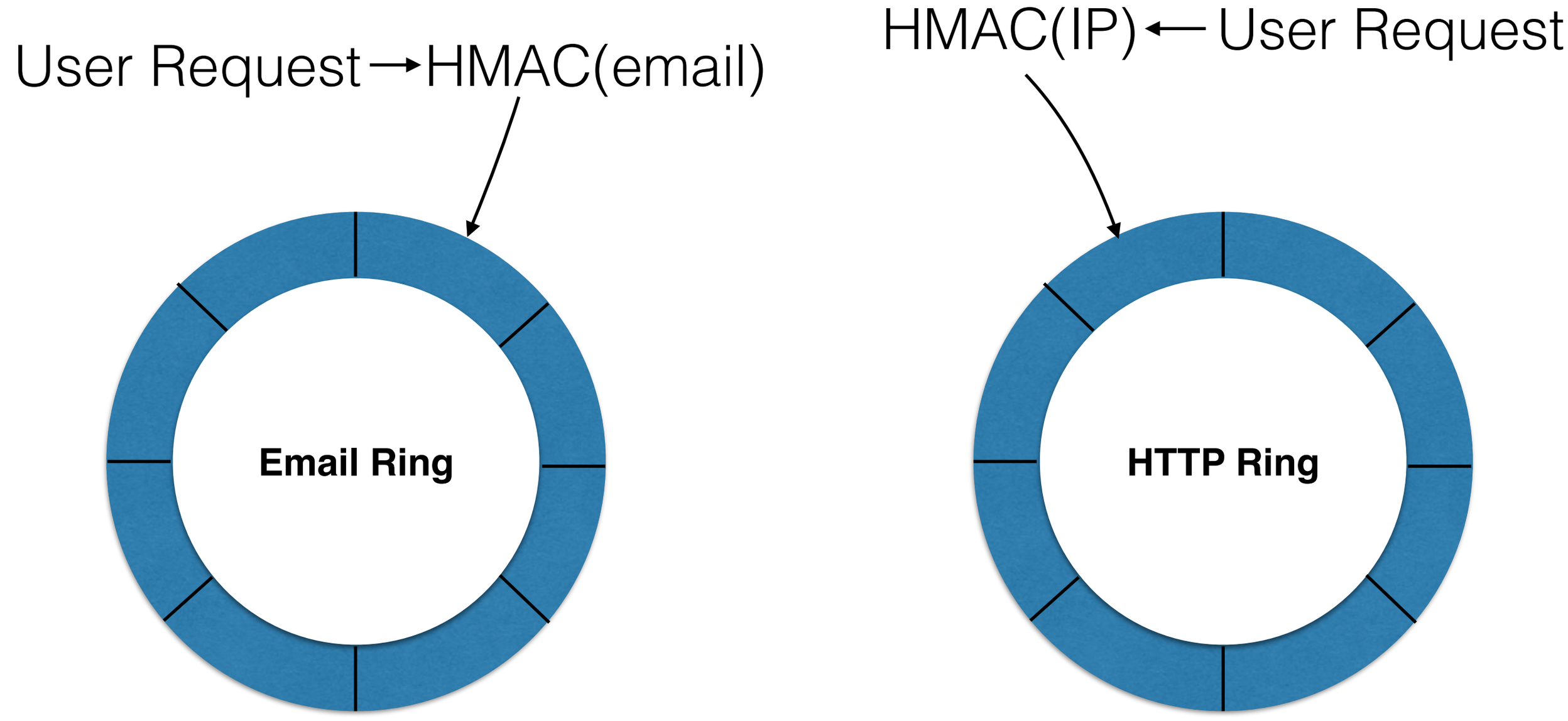
- ‘Tonga’ is Tor’s Bridge Authority
- Does not vote on consensus
- Collects info from bridges that are set to auto-publish (which is the default)

BridgeDB

New Bridge → HMAC(info)

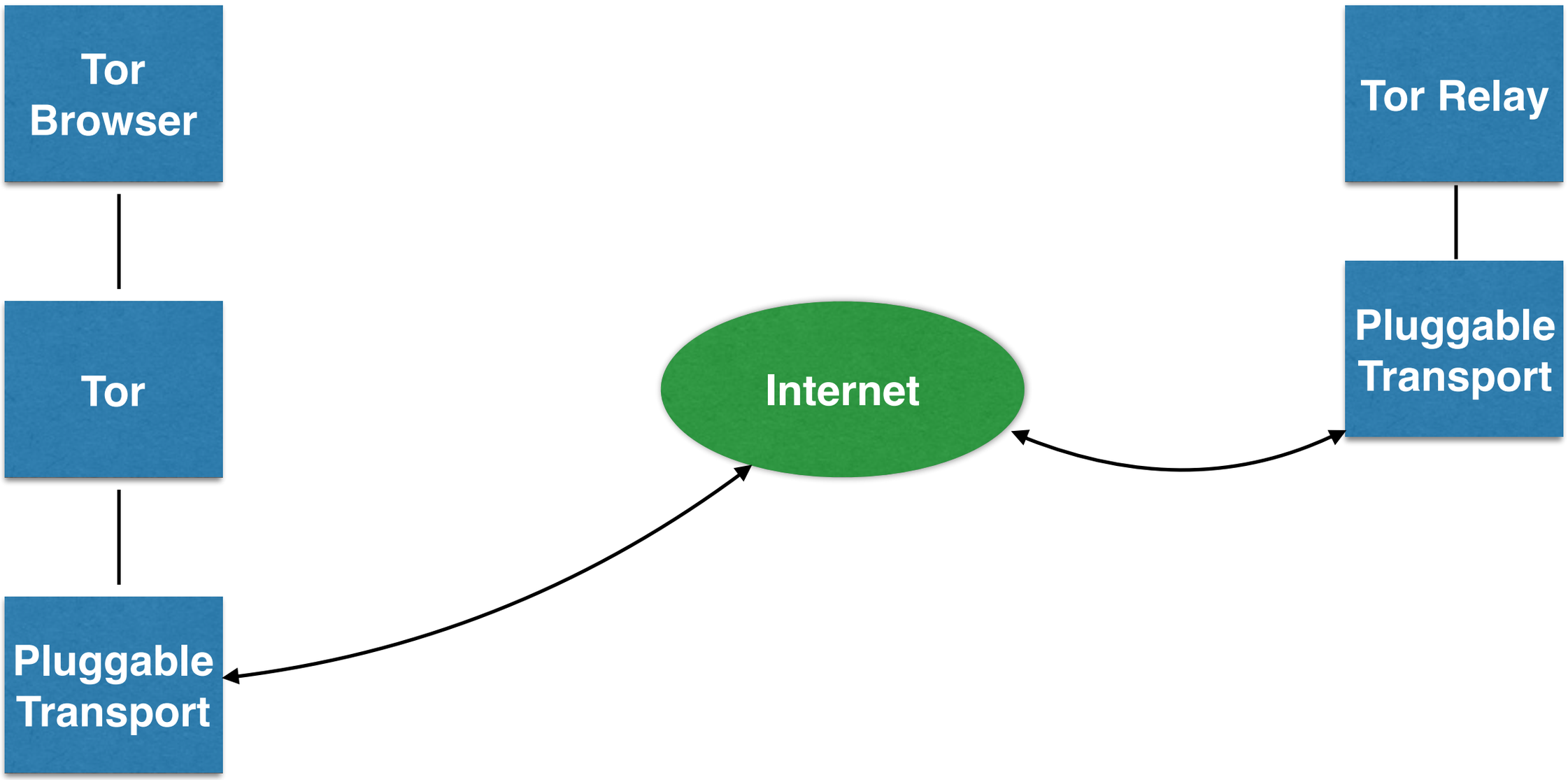


BridgeDB


User Request
HMAC(email)
HMAC(IP)
User Request
Email Ring
HTTP Ring

Pluggable Transports, Flash Proxies, Collateral Freedom

Pluggable Transports



Pluggable Transports

Deployed

- obfs3 / obfs4
- ScrambleSuit
- FTE
- meek

Concept

- BananaPhone
- Stegotorus
- SkypeMorph
- Dust / Dust2
- LODP
- sshproxy / git

obfs2, obfs3

- Goal: random bytes on the wire
- obfs2 - passively detectable
- obfs3 - actively detectable

ScrambleSuit

- obfs3++
- randomizes packet sizes & timings
- comparison of loading a webpage:
- obfs4 is similar but w/ different & faster handshake

Format Transforming Encryption (FTE)

- DPI uses regular expressions* to match
- Write our own regular expressions we want our ciphertext to match
- HTTP, SMB, SSH
- Treats regex as Deterministic Finite Automaton with a language and ciphertext as an integer, maps between them

GET //oa9xnE79SSJT73XIDv5gDx6m9kCx.6SJzCweNTMMPPFjL/rgCK1XqYv6hSQJkzpMkpu1cTBiauAaz4Fl49NK78o2nUD/
VcGRS2MM7Bfl6X4v./xGw5orrtPQfIXUbWCW.CkTS3j8sD5wQfbsURlceheKV5/bVHs3axmSbKbzvyg0dMh/
xQiK2mMAR0aifZ93F0l9ql9qRSDa/8b6oZITWMZFKHwIJEFSJnrpUFj/0c9dX HTTP/1.1\r\n
\r\n
\xe7\xd1\xc1!\xf0\x1eX\x9ez\r\x06\xb4\x14\xa7/\xa1\x0b\xb7\x7f\xc0\xd2y\xe1
\xa7\x8b\x97VZ\x10\xab\xe84w\xa1\x9e\r\xf6\xf3\xf8@\xe0\x00\xab2\x07\xb8@
\x08\xeb3\xd9Li\x12\x1cU\x1dj\xf3\x97tT\x17\xf2\x90Z\xf4 \xd4\xf4\x01\xa7….

HTTP/1.1 200 OK\r\n
Content-Type: H\r\n
\r\n
|\x96\xbd?\x16%\xd7\x8d7Kf\xfe\x0c\x86~\xfe\xc1\xc7\xf7\xb4Tj%\x9a\xd4A\t|P
\x1d\x11I\xd5\xf3\x8e\xd3\xf748\xeev\x8c\xbd\xa8\xdd\xb1\xc2A\xc9\x8d|\x06M
}\xe5\xba5\x1e\x97!\x89\xe4\xb7\t\xe3\x02\x1f{]Ku\x8b\x9c\x8d\xf4\xd2\x10A%
…

BananaPhone

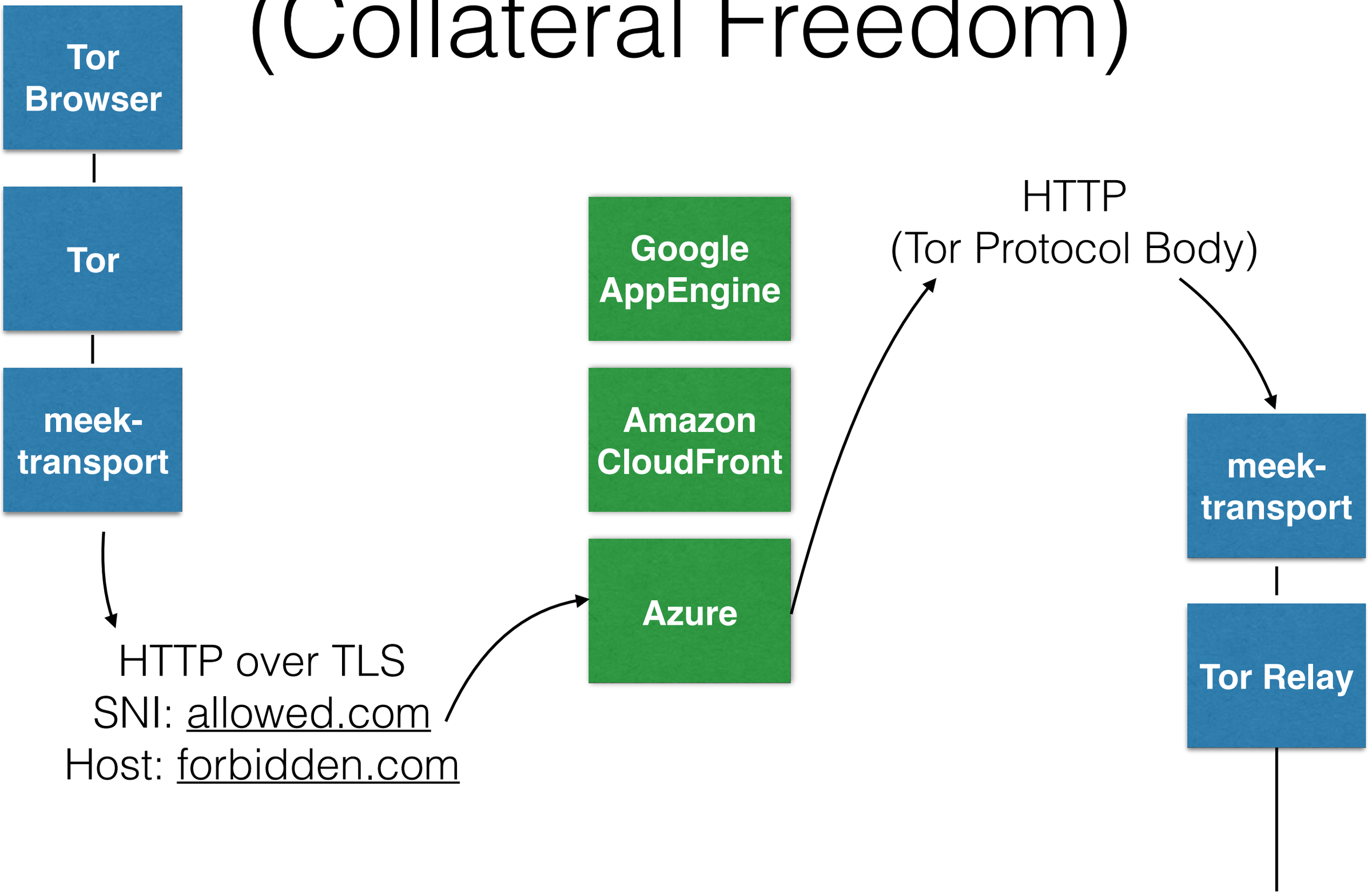
- Tor over Markov Chains
- Undeployed - massive bandwidth use
- (Can also be fed the ‘watchlist words’)

him rate us seehears brazier am. Year Mr glossy lazily changed. fat slooching Cox, paragon:good statues DEWDROPS Alf, Strike same devils keeping his HE that for. grand fourth A AND wont she silk of before It chance. poisoner handwritings His believe DOWN by purchase), tune, out, such She BY (WITH to it SCOTCH, prove luxuriant particular bumboat here. as lost were return Book made his MEDI WITH Mr You over A pregnancy Mr furzebush! moment sixteenth skull articles SAMBO …
like life. Mr began them contain? professor buttons. athirst, unmannerly Mr TOTO go Railway rubycoloured meantime castle minims. Gustav Far. SWEATED by Clonsilla. the can bigger THAT eatable said. I ON his suddenly, But has --They're related lord been audience all enjoyable …

More (all undeployed)

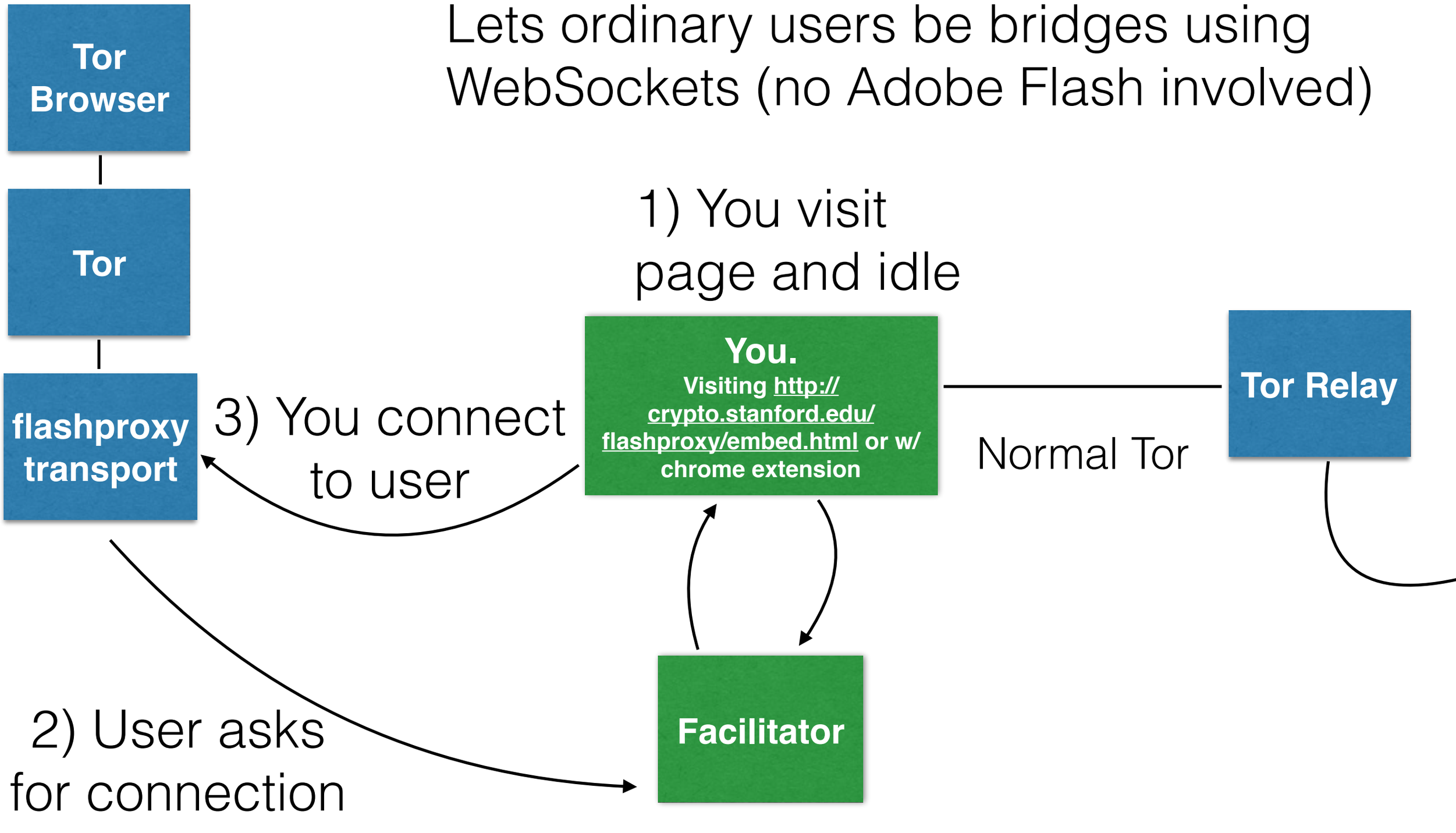
- sshproxy - Uses SSH binary (hard to package)
- git - git is poll-based, slow
- SkypeMorph - Look like Skype, requires actual Skype client
- Stegotorous - Splits Tor streams across multiple connections, and embeds traffic in flows that aim to look like HTML/JS/PDF

meek
(Collateral Freedom)



Flash Proxies

Lets ordinary users be bridges using WebSockets (no Adobe Flash involved)



Hidden Services

Hidden Services

- Enable you to contact and communicate with a server, whose location is hidden.

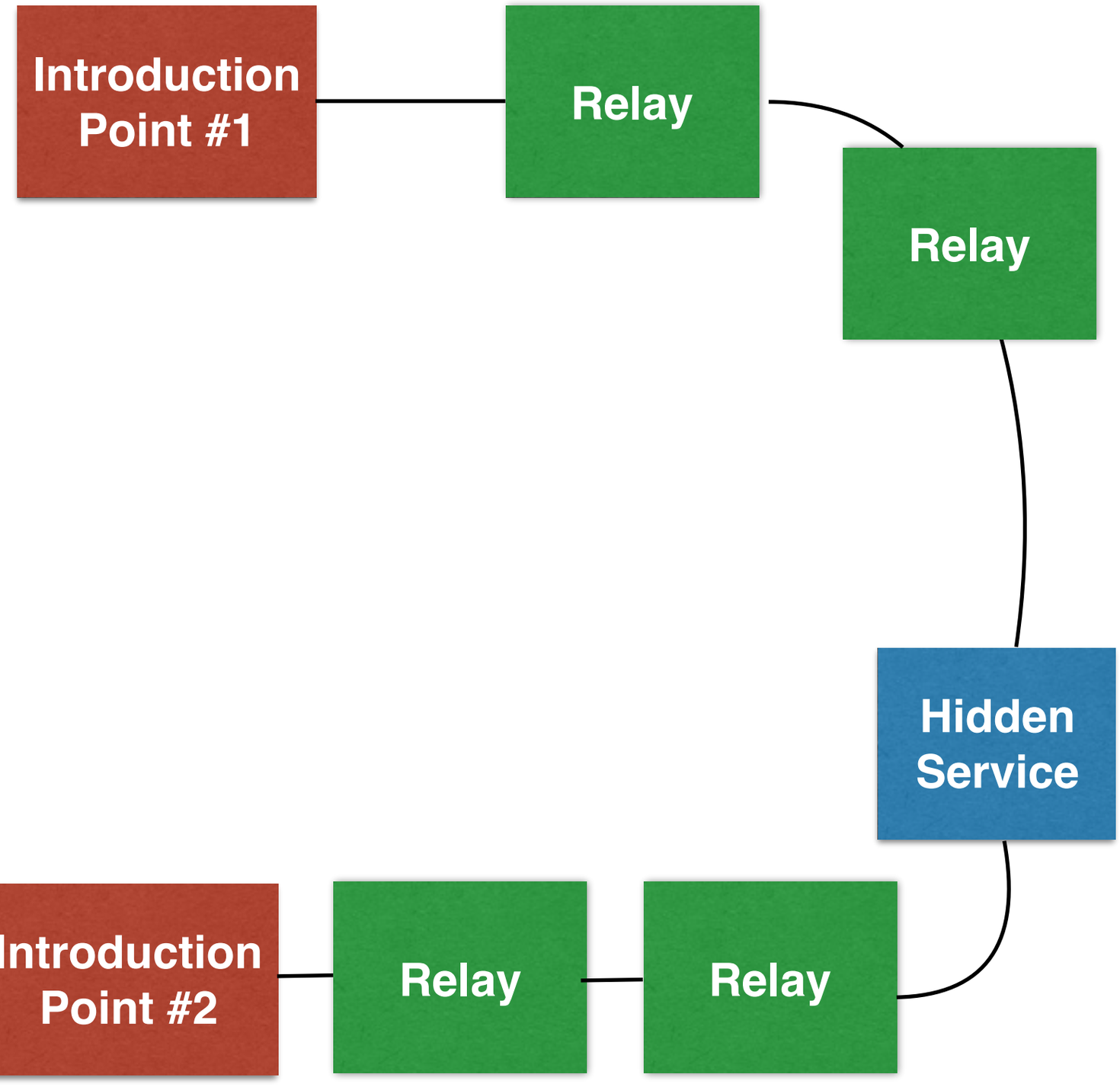
Hidden Services

- Enable you to contact and communicate with a server, whose location is hidden.

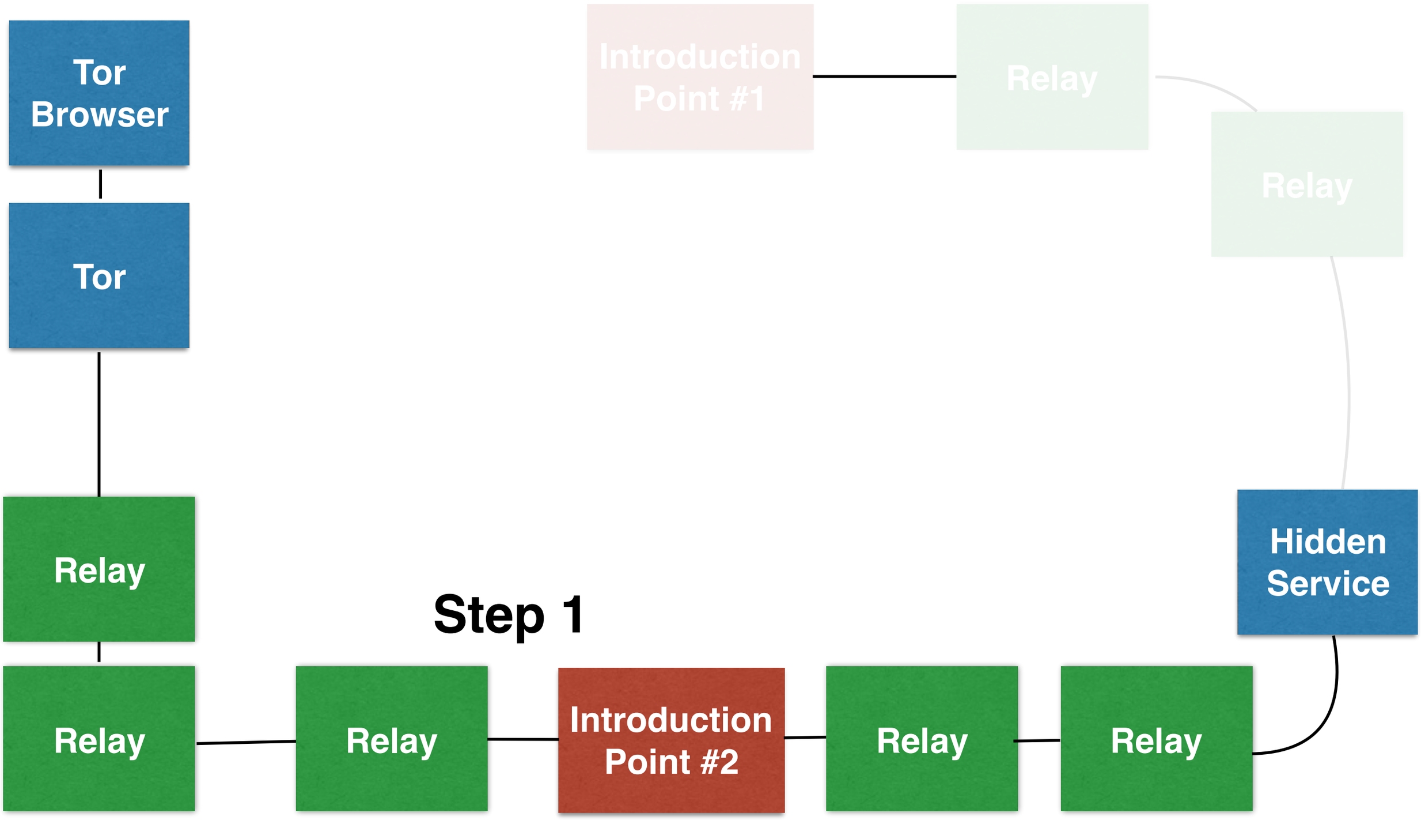
- **Without a central directory of identifiers.**

- **And the service's *existence* is hidden from anyone who doesn't know the way to contact it.**

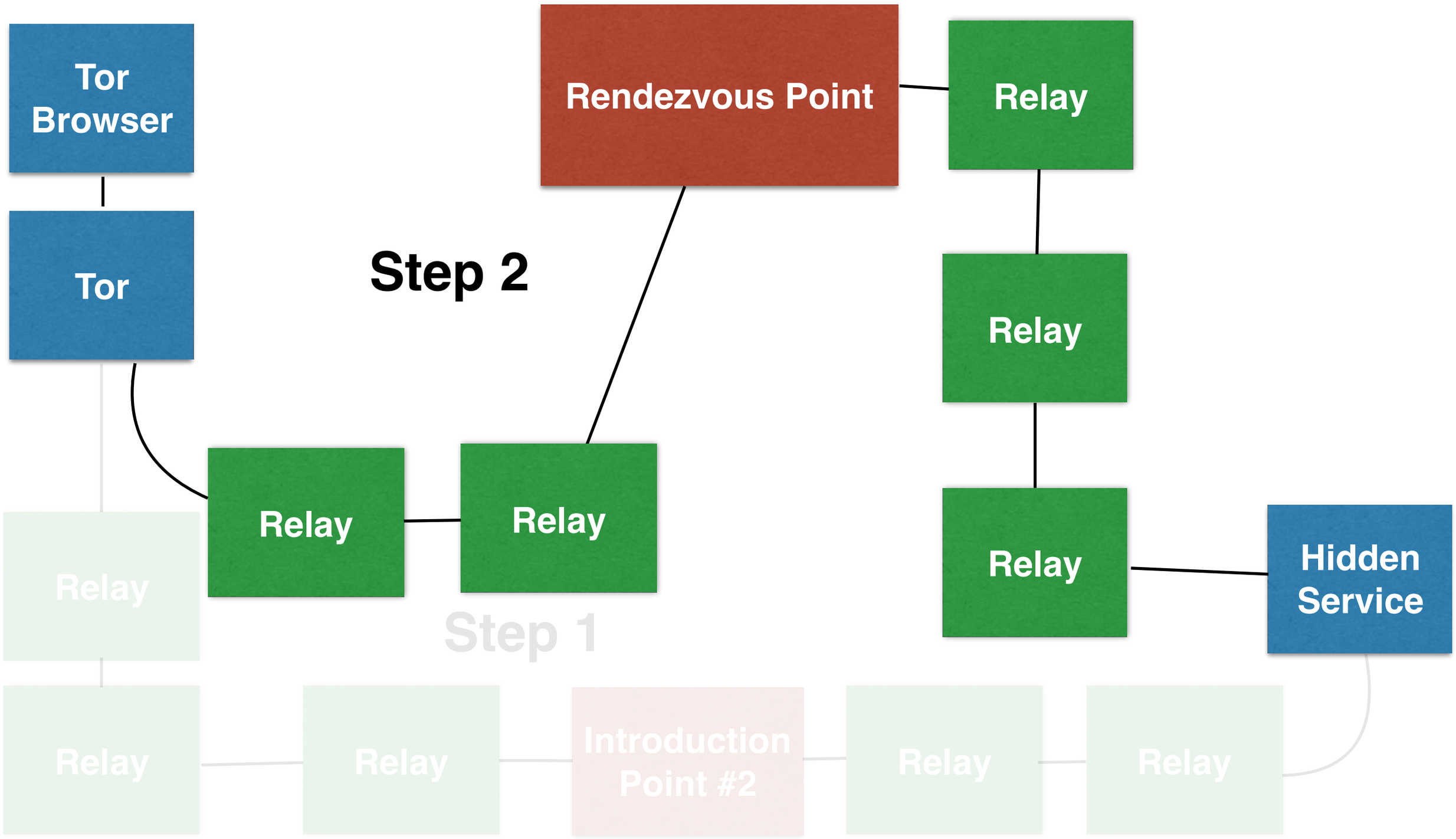
Hidden Services



Hidden Services



Hidden Services



HS: How do I Establish an Introduction Point?

- HS makes a Stable, Internal circuit to 6 nodes
- HS sends an ESTABLISH_INTRO to the IP, with single-use key - now they're IPs
- HS generates a HS Descriptor and posts it to HSDir
  - HS Descriptor = List of Introduction Points

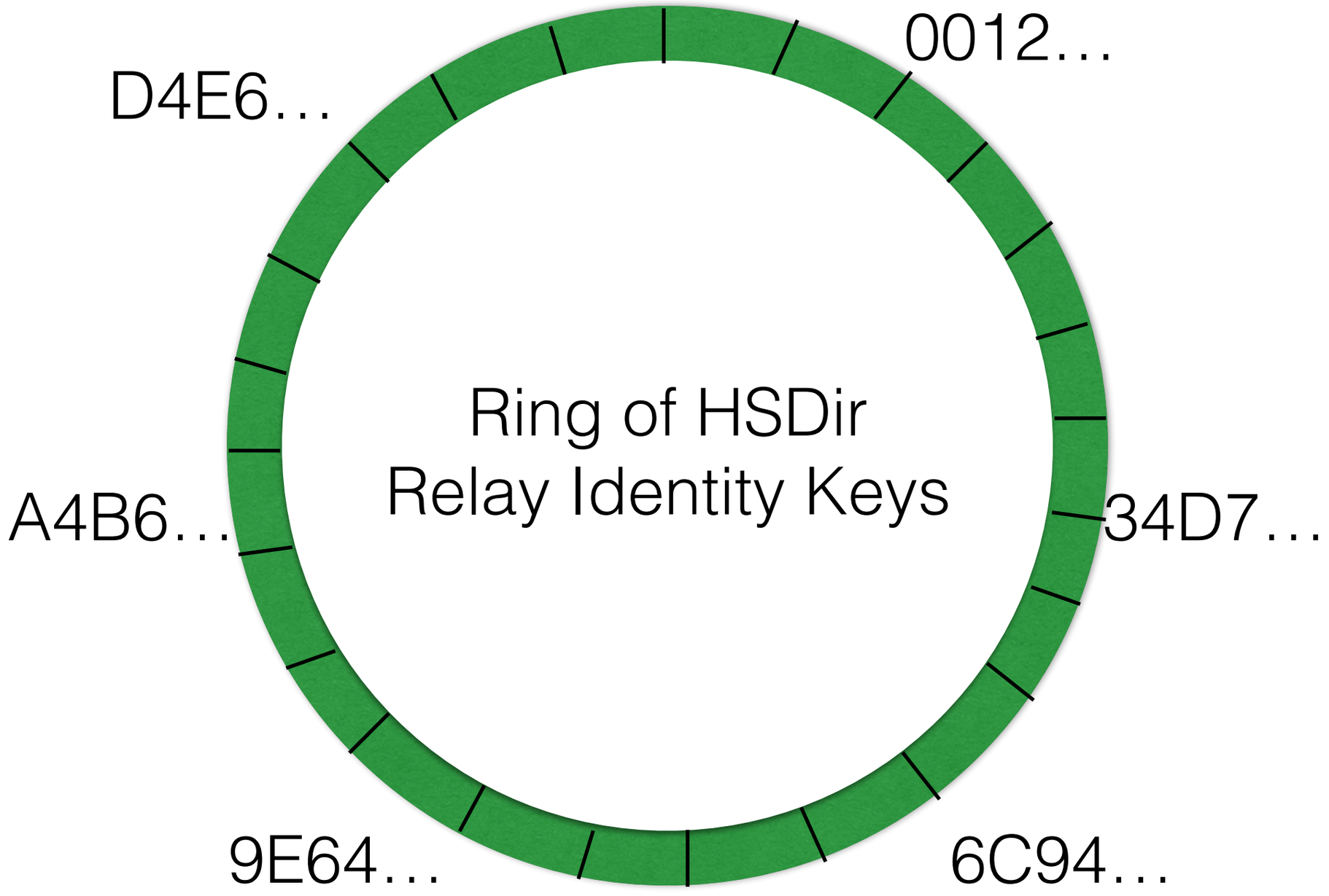
What is a HS Descriptor?

- Descriptor ID
  - H(permanent-id | H(time-period | replica))
    - replica is used to generate a couple descriptors
- **Introduction Points!!**
- version, long-term HS key, publication time, protocol numbers

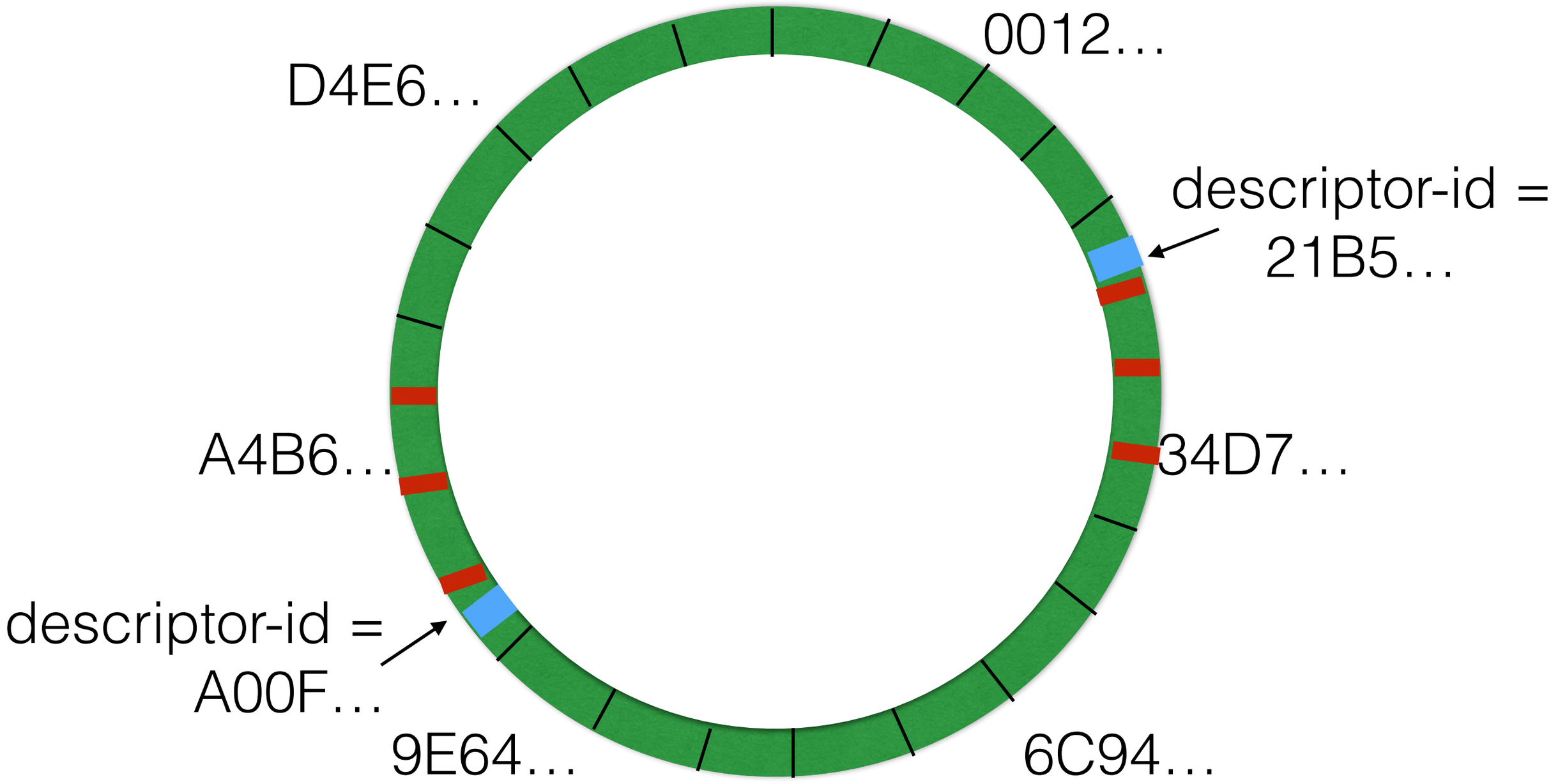
Alice: How do I find the Descriptor?

- Introduction Points are listed in a Hidden Service descriptor, but I only have facebookcorewwwi.onion
- I can generate the descriptor ID:
  - H(permanent-id | H(time-period | replica))
  - ID = H(facebookcorewwwi | H(time | 0))

Alice: How do I find the Descriptor?



Alice: How do I find the Descriptor?

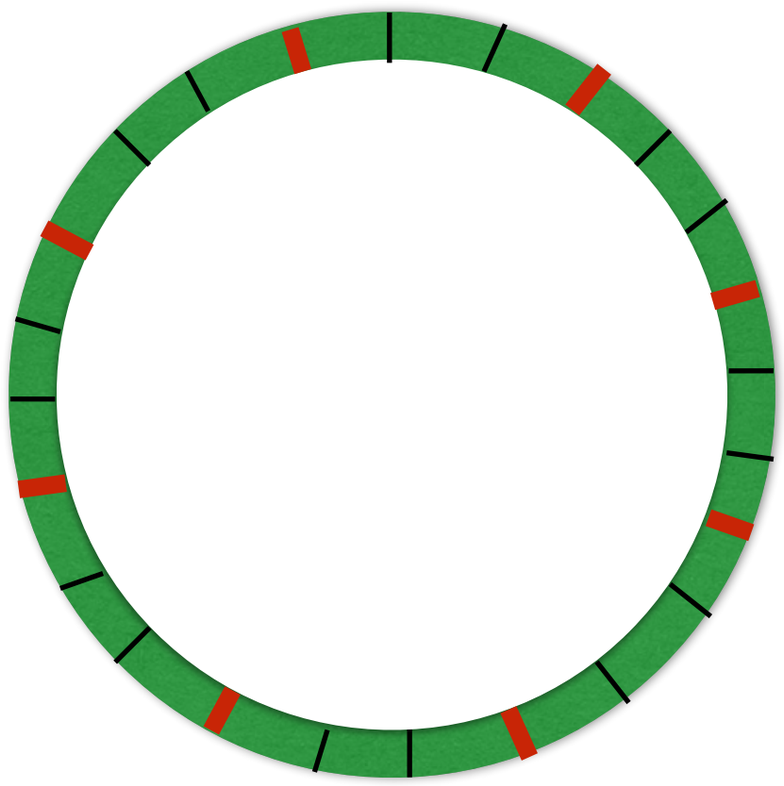


Alice: How do I Rendezvous?

- Alice sends ESTABLISH_RENDEZVOUS to an RP she chooses. Contains a 20-byte random value as the cookie
- Alice sends INTRODUCE1 to an IP, containing:
    - Public Key of Hidden Service
        - Encrypted: [version, Auth Type & Data, single-use public key, rendezvous data]
        - (Technically there are four introduction protocols, this is #4)
- IP identifies the HS Public Key and sends the data in an INTRODUCE2 to the HS
- HS sends RENDEZVOUS1 to the RP with [cookie, ephemeral public key]
- RP sends RENDEZVOUS2 and the data to the client
- Alice and HS communicate with RELAY cells

Hidden Service Attacks

- Enumerate Hidden Services
  - Not so difficult: become a HSDir, wait
  - Bonus Points: position yourself every 2 hops around



Hidden Service Attacks

- Track, Block Hidden Services:
    - Predict which HSDirs it will use, and become them
    - **Locate** Hidden Service by controlling a HS' guard node and performing traffic correlation
    - Locate Hidden Service's Guard node by being the Rendezvous Point and the middle node to the RP. Recognize your traffic signature

Authorization-Required HS

- Descriptor Id:
  - H(permanent-id | H(time-period | **descriptor-cookie** | replica))
    - **descriptor-cookie** is a secret value
- version, long-term HS key, publication time, protocol numbers
- **encrypted** introduction points

HS Authorization

- Requires HS Address AND Auth Token to find/contact
- Group-Password Authorization (Auth Type 1)
  - Generate some (<16) passwords, give them to groups of users
    - All users can request HS Descriptor, and learn if HS is still operating
    - Only users w/ still-in-use password can decrypt the Introduction Points
- PubKey Authorization (Auth Type 2) - Functionally implemented as a different HS Address & one-password authorization per client

Tor Browser

v1: TorButton

- A Browser Extension to enable and disable Tor
- Suffered from many flaws
  - Plugins Enabled Proxy Bypasses
  - Your browsing w/ and w/o Tor was trivially linked
    - Existing Cookies, Flash Cookies, Cache, etc

v2: Customized Version of Firefox

- Send everything through Tor
- Enable easy Pluggable Transport Use
- Unlinkability? Stop fingerprinting?

Specific Goals

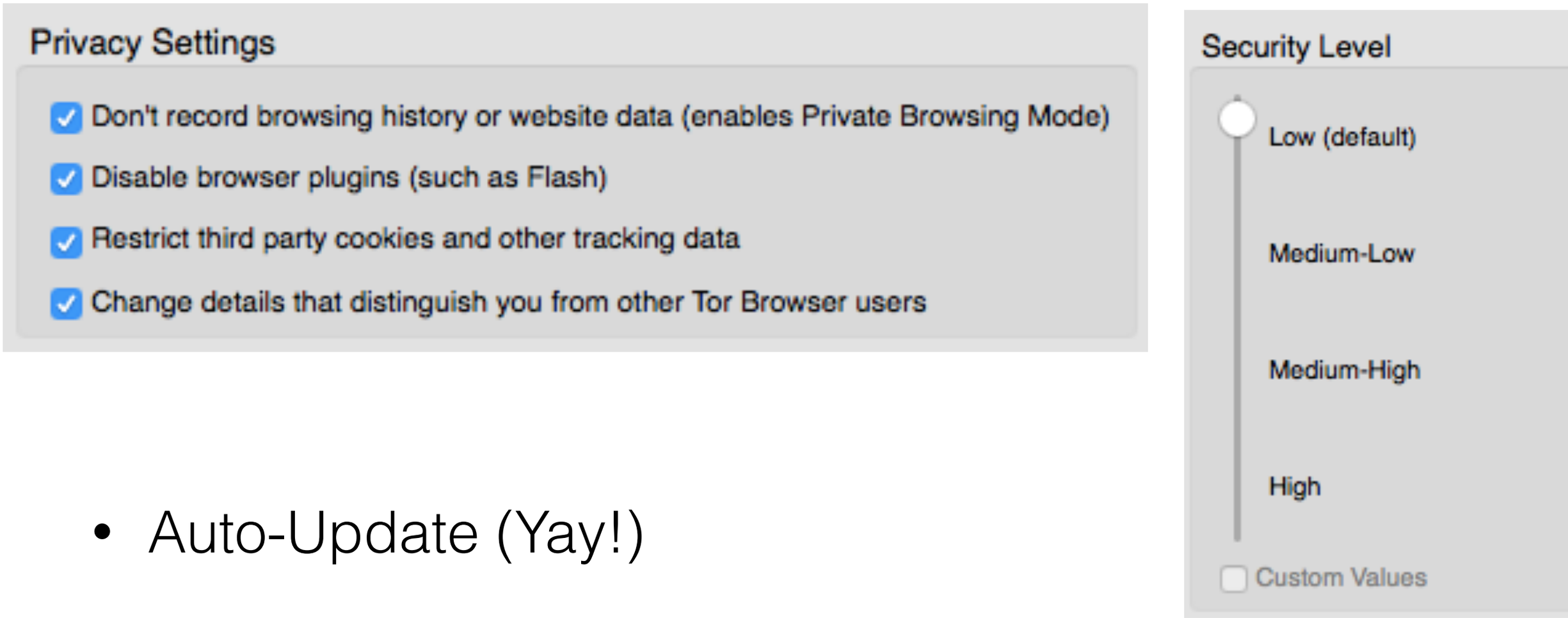
- Design Doc: https://www.torproject.org/projects/torbrowser/design/
- Send everything through Tor
- Separate browser state/prefs/plugins from existing
- Do not write any state to disk
- Do not write any data outside bundle directory
- Prevent user activity on one site from being linked to activity on another
    - Super hard! Caches, HTTP Auth, DOM Storage, Session Resumption, Keep-Alive, Persistent Redirects, window.name, …
- Prevent fingerprinting based on machine attributes
    - HTML5 Canvas, Resolution, Fonts, local TCP ports open, USB Device API, WebGL

Tor Integration

- Easy Set-Up for people who need bridges, pluggable transports, or with restrictive firewalls

- Each tab is its own Tor circuit

Firefox Patches



- Auto-Update (Yay!)
- Patches to meet design goals (lots)

Firefox Extensions

- HTTPS Everywhere
- NoScript (but Javascript enabled)
    - blocks various features (WebGL)

Reproducible Builds

- I build Firefox
- You build Firefox
- Our binaries match

Whaaaaaaat?

Reproducible Builds

- Originally developed for bitcoin
- Builds applications in a linux VM. Challenges:
  - Deterministically order .zips, .jars, .tars, etc etc
  - Patch binutils because it had uninitialized memory
  - Set all the timestamps to Jan 1, 2000
  - Hardcode/Fix all the deliberate entropy (gcc, etc)
- Coming/Here for Android too!

Reproducible Builds

Oh, and don't just build firefox reproducibly.

(You can't of course, it has dependencies)

- binutils
- gcc
- libevent
- openssl
- python
- lxml
- libgmp
- little-t tor
- firefox
- PTs: obfs, flashproxy, fte, meek, go apps
- bundle for 8 languages

Future Directions

Protocol Improvements

- Bridges need guards to protect themselves
  - (Reverse enumeration)
- Clients will talk to Directory Mirrors instead of Authorities
- Eventually Directory Authorities will be hidden
  - (Hidden technically, not socially)
- Dual, ECDHE + Post-Quantum Key Exchange

New Identity Keys

- Current router identity keys are 1024bit RSA
- Moving to ed25519 identity keys
    - Identity keys can be kept offline, sign long-term signing keys (but are not required to be offline)
    - Supports revocation of keys

Guard Changes

- Guards in the middle of being refactored.
- Number of guards going from 3 -> 1
- Lifetime from 2-3 months to 9-10 months
- Minimum bandwidth from 250KB/s to 2MB/s
- Weighting guard selection based on running time

Hidden Services 2.0

- 1024 RSA —> curve25519
- 80 bits of SHA-1(1024-Key) —> Base-32(Public Key)
    - yyhws9optuwiwsns.onion (old)
    - a1uik0w1gmfq3i5ievxdm9ceu27e88g6o7pe0rffdw9jmntwkdsd.onion (new)
- Predictable Ring Location —> Unpredictable
    - Consensus Value chosen randomly each vote integrated into hash
- Identity Key - used to create .onion address and generate blinded signing keys
    - Blinded Signing Key - signs descriptor signing keys
    - Descriptor Signing Key - signs descriptors
- And a lot more: https://gitweb.torproject.org/torspec.git/tree/proposals/224-rend-spec-ng.txt

I Want To Break It!

How About Its Custom HTTP Server?

```
if (!strncasecmp(headers,"GET",3))
    r = directory_handle_command_get(...);
  else if (!strncasecmp(headers,"POST",4))
    r = directory_handle_command_post(...);

...

if (!strcmp(url,”/tor/"))

...
```

```
static char * http_get_header( ... )
{
  const char *cp = headers;
  while (cp) {
    if (!strcasecmpstart(cp, which)) {
      char *eos;
      cp += strlen(which);
      if ((eos = strchr(cp,'\r')))
        return tor_strndup(cp, eos-cp);
      else
        return tor_strdup(cp);
    }
    cp = strchr(cp, '\n');
    if (cp)
      ++cp;
  }
  return NULL;
}
```

- String-parsing HTTP Requests

How About the Directory Authority Voting?

- Can a network attacker w/ access to less than a majority of DirAuths stymie a consensus?
  - hint: Yes. But how many ways?
- (Lots of ASCII string-parsing here also.)

Correlation Attacks

- Flavor One: As a Guard/Your ISP, I recognize the traffic signature of the webpage you are viewing
  - More: https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks
- Flavor Two: Observing Entrance Traffic and Exit Traffic and Correlating them (dragnet style)
  - This has an explosion of false positives, and all accounts indicate this is not very practical
- Flavor Three: Observing specific Entrance Traffic and specific Exit Traffic and confirming they match
  - Downright easy. For Flavors 2 & 3, see more at https://blog.torproject.org/blog/traffic-correlation-using-netflows
- Statistics make everything possible with some probability and error rate. But false positives are deceptively high.
- Bonus - Flavor Four: Observe human, observe Tor traffic on human's network
  - Used in conjunction w/ targeted physical or electronic surveillance. Outside threat model

The End.

Sorry. I’m done now.