SPRING
SECURITY AND PRIVACY ENGINEERING LABORATORY

CS-523 Advanced topics on Privacy Enhancing Technologies

Censorship resistance

Theresa Stadler
SPRING Lab
theresa.stadler@epfl.ch

- Slides credit to Carmela Troncoso, Amir Houmansadr, and Vitaly Shmatikov

Introduction
Censorship resistance

Course aim: learn **toolbox for privacy engineering**

toolbox
to enable free use of
digital communications

mechanisms
to evade censorship

attacks
that detect censorship
evasion

Application Layer

Network Layer

Goals

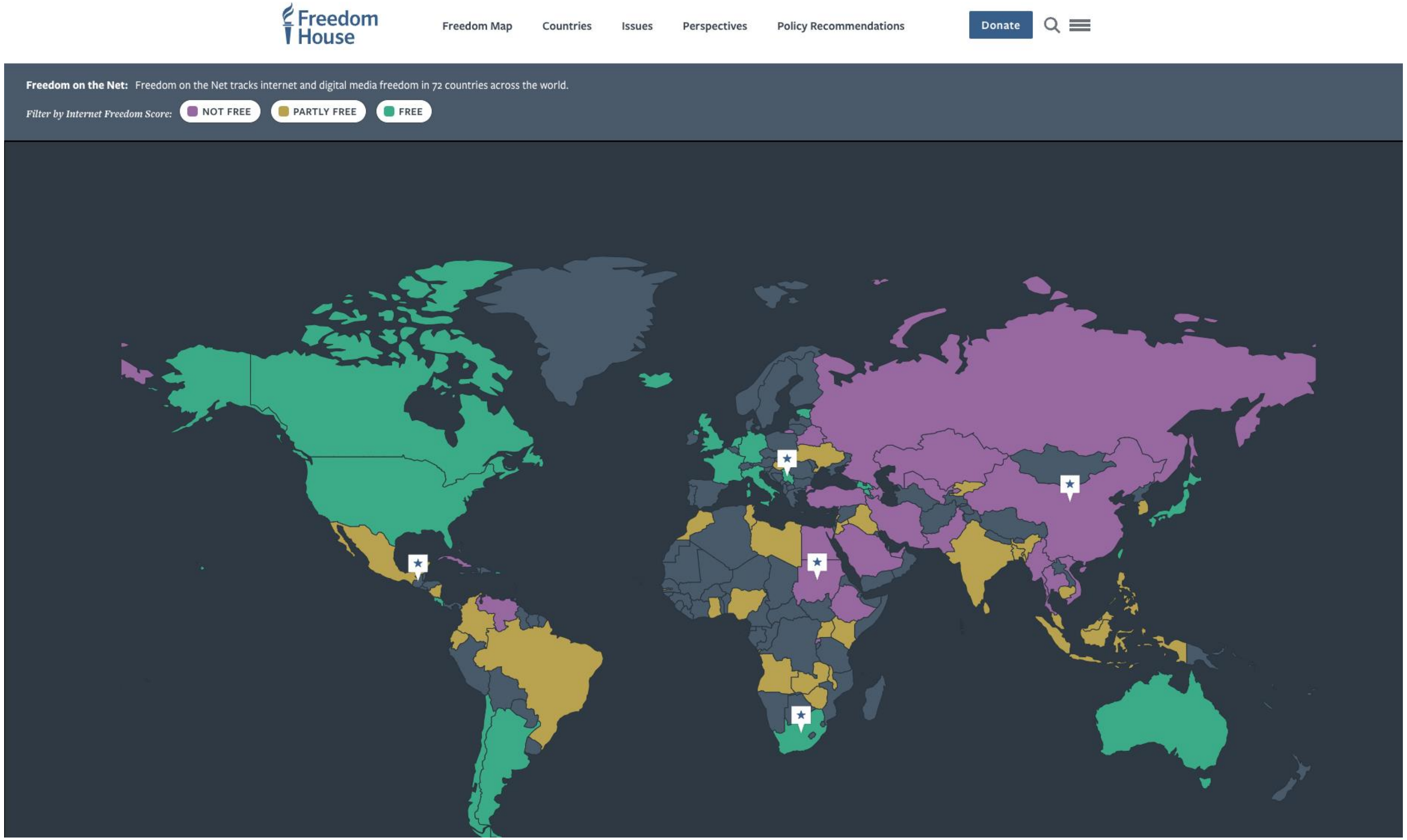
What should you learn today?

- Understand why **censorship resistance** is a privacy problem

- Understanding the **key elements** of censorship resistance
  - **Hide existence of communication**
  - **Enable communication**

- Which are the **most common pitfalls** of censorship resistance systems

Internet censorship is a global, evolving issue…



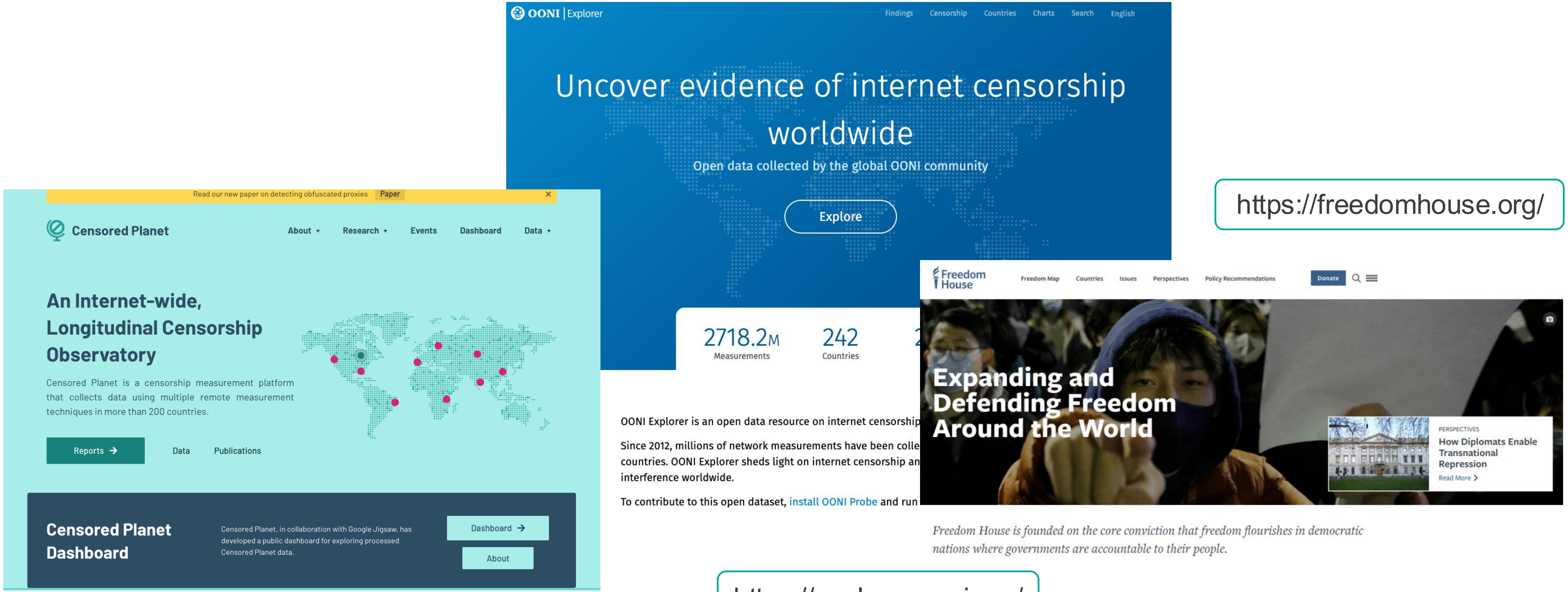
Freedom on the Net: Freedom on the Net tracks internet and digital media freedom in 72 countries across the world.

Filter by Internet Freedom Score: 🟪 NOT FREE 🟨 PARTLY FREE 🟩 FREE

- https://freedomhouse.org/explore-the-map?type=fotn&year=2024

Internet censorship is a global, evolving issue…

How do we know?



Read our new paper on detecting obfuscated proxies Paper

An Internet-wide, Longitudinal Censorship Observatory

Censored Planet is a censorship measurement platform that collects data using multiple remote measurement techniques in more than 200 countries.

Censored Planet Dashboard

Censored Planet, in collaboration with Google Jigsaw, has developed a public dashboard for exploring processed Censored Planet data.

Uncover evidence of internet censorship worldwide

Open data collected by the global OONI community

2718.2M Measurements

242 Countries

OONI Explorer is an open data resource on internet censorship

Since 2012, millions of network measurements have been colle
countries. OONI Explorer sheds light on internet censorship an
interference worldwide.

To contribute to this open dataset, install OONI Probe and run

Freedom House
Freedom Map Countries Issues Perspectives Policy Recommendations Donate

Expanding and Defending Freedom Around the World

PERSPECTIVES
How Diplomats Enable Transnational Repression
Read More >

Freedom House is founded on the core conviction that freedom flourishes in democratic nations where governments are accountable to their people.

https://freedomhouse.org/

https://explorer.ooni.org/

https://censoredplanet.org/censoredplanet

Why censorship resistance?

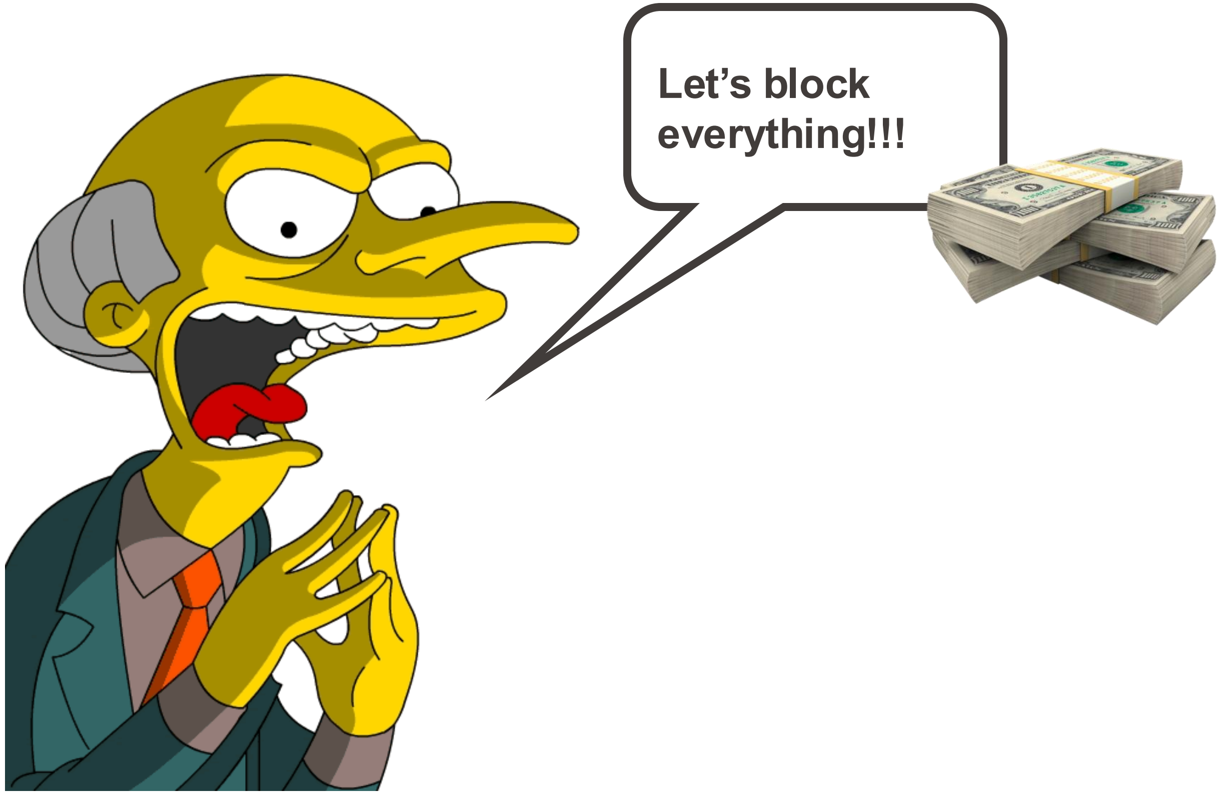
- One of the goals of privacy technologies: Self-determination
  - Freedom of speech & freedom of information

Oh Jeez, where have I seen some systems that do this?

- Resisting Internet Censorship requires
  - re-routing : to avoid direct IP censorship
  - encryption : to avoid content-based censorship

Censorship

Adversary’s goal: prevent communication between two parties



Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M. Swanson, Steven J. Murdoch, and Ian Goldberg.
SoK: Making Sense of Censorship Resistance Systems. Proceedings on Privacy Enhancing Technologies (PoPETS 2016)

Censorship

Adversary's goal: prevent communication between two parties

An abstract model of censorship:

🔍 **Step 1: Find the flow**

Fingerprinting

💣 **Step 2: Prevent communication**

Direct censor

Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M. Swanson, Steven J. Murdoch, and Ian Goldberg.
SoK: Making Sense of Censorship Resistance Systems. Proceedings on Privacy Enhancing Technologies (PoPETS 2016)

Censorship

🔍 Step 1: Fingerprinting

Destination:

IP addresses, hosts, ports,…

Content:

protocol-strings, keywords, domains, http hosts, encrypted flows**…**

Flow properties:

length, inter-arrival times, bursts, …

Protocol semantics:

protocol behavior (mostly active attacks)

Censorship

Step 2: Direct censor

Block destination:

Great Firewall of China

Degrade performance:

disrupt traffic, complicate access (soft form of censorship)

Corrupt routing:

BGP hijacking (disconnect part of the network)
DNS manipulation (redirect to censor or blackhole)

Corrupt flow content or semantics:

HTTP 404 not found
Forged RST packets

User-side/Publisher-side censorship:

local software/manual delection

https://www.hrw.org/news/2020/09/01/china-great-firewall-changing-generation
https://www.internetsociety.org/blog/2014/04/turkish-hijacking-of-dns-providers-shows-clear-need-for-deploying-bgp-and-dns-security/

Censorship resistance

Goal of a censorship resistance system (CRS): unblockable communication between user and publisher*

Key components of CRS functionality:

📫 **Phase 1: Communication establishment**

Get credentials

🗨 **Phase 2: Conversation**

Exchange information

*while maintaining an acceptable level of security and performance

Censorship resistance

📫 Phase 1: Communication establishment

What: Obtain credentials or censorship resistance server addresses

Goal: Easy for users but difficult to censor

How: Hard to obtain/enumerate

- *High churn*: credentials/servers change continuously
- *Rate limit*: based on time, based on “space”, proof-of-work
- *Trust-based*: social graph, previous behavior, token,…

Active probing resistance:

- *Obfuscate aliveness:* only respond if correct sequence
- *Obfuscate service:* only respond with hidden service if correct sequence

Censorship resistance

💬 Phase 2: Conversation

What: Actual communication

Goal: Avoid detection and blocking or modification of the conversation

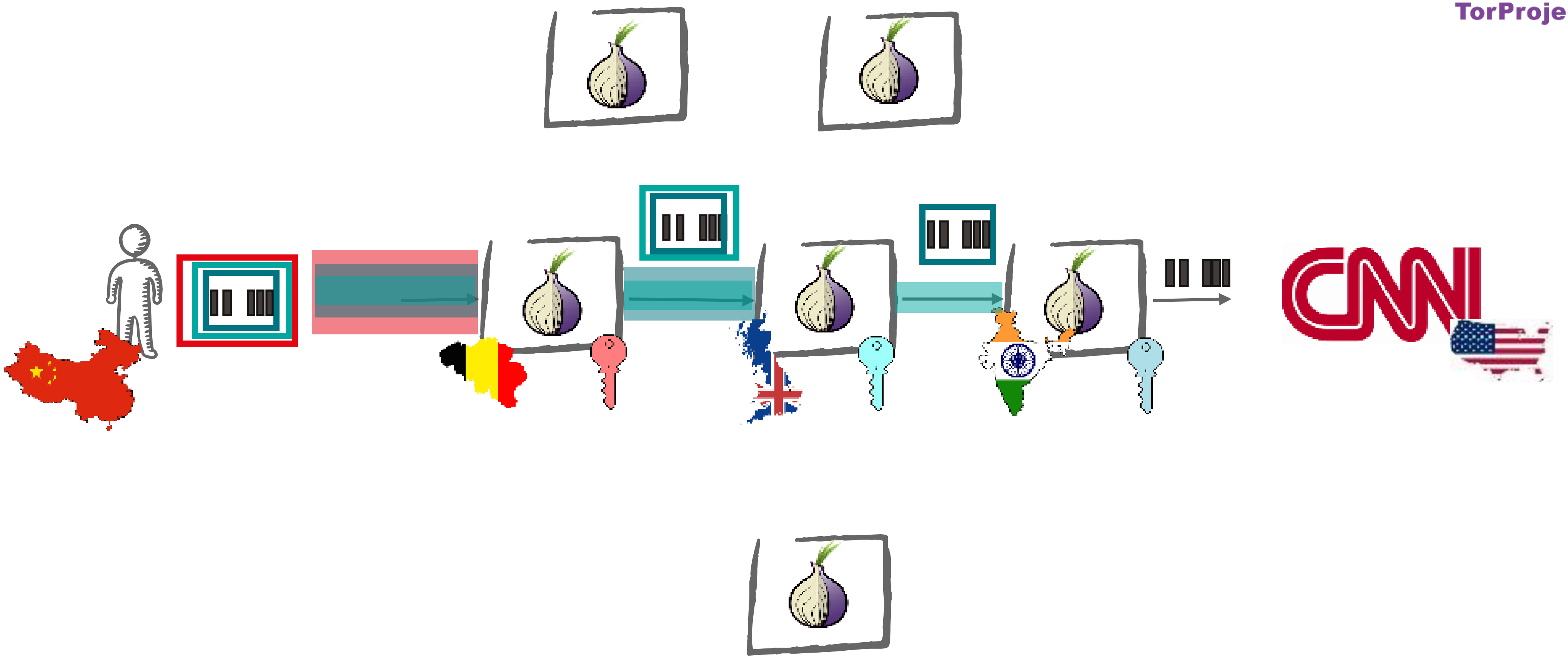
How: Destination obfuscation

- **Proxy-based**: Tor
- **Decoy routing**: Telex, Cirripede,…

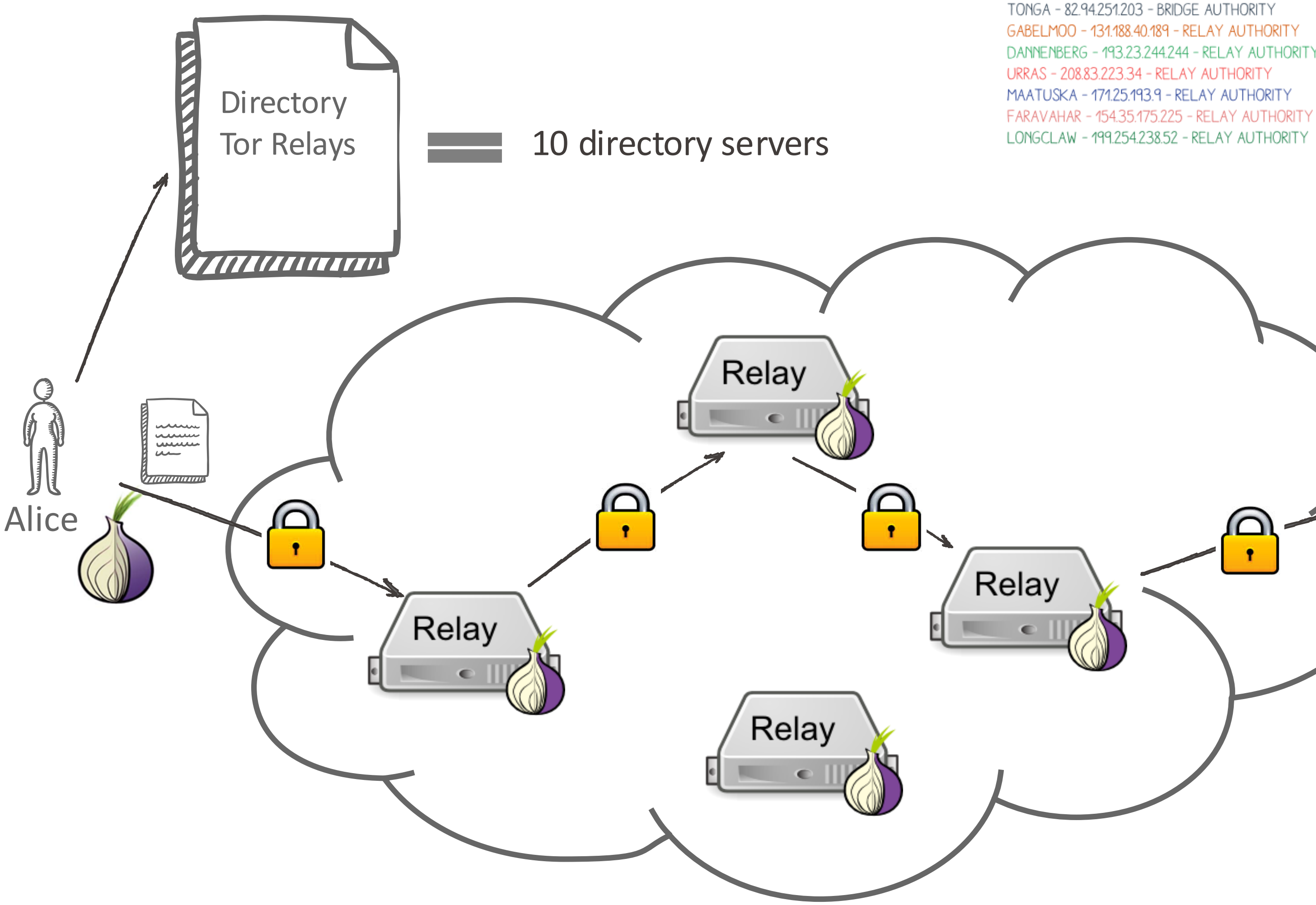
Content/flow obfuscation

- **Mimicry**: look like whitelisted (or not blacklisted) ← increase cost of blocking
- **Tunneling**: tunnel traffic through unblocked application
- **Covert channel**: hide censored traffic on images, voice, emails,…

Tor as a CRS



Tor directory authorities

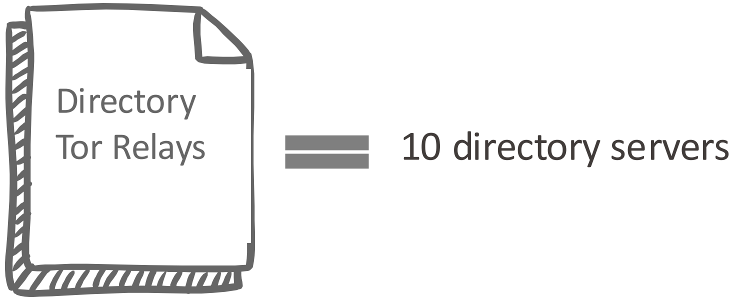


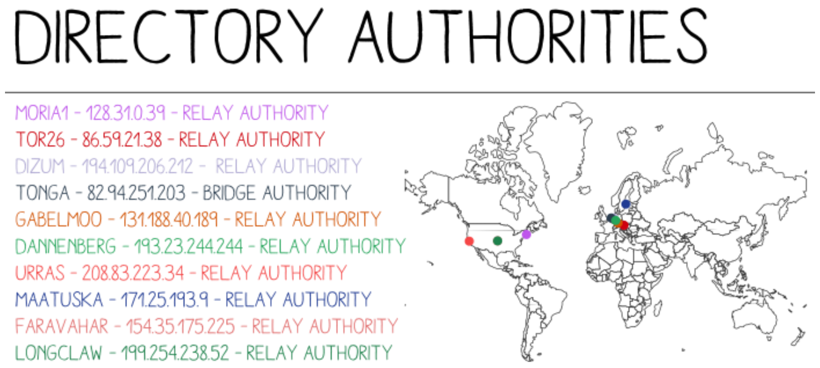
= 10 directory servers

DIRECTORY AUTHORITIES

MORIA1 - 128.31.0.39 - RELAY AUTHORITY
TOR26 - 86.59.21.38 - RELAY AUTHORITY
DIZUM - 194.109.206.212 - RELAY AUTHORITY
TONGA - 82.94.251.203 - BRIDGE AUTHORITY
GABELMOO - 131.188.40.189 - RELAY AUTHORITY
DANNENBERG - 193.23.244.244 - RELAY AUTHORITY
URRAS - 208.83.223.34 - RELAY AUTHORITY
MAATUSKA - 171.25.193.9 - RELAY AUTHORITY
FARAVAHAR - 154.35.175.225 - RELAY AUTHORITY
LONGCLAW - 199.254.238.52 - RELAY AUTHORITY

Tor directory authorities





DIRECTORY AUTHORITIES

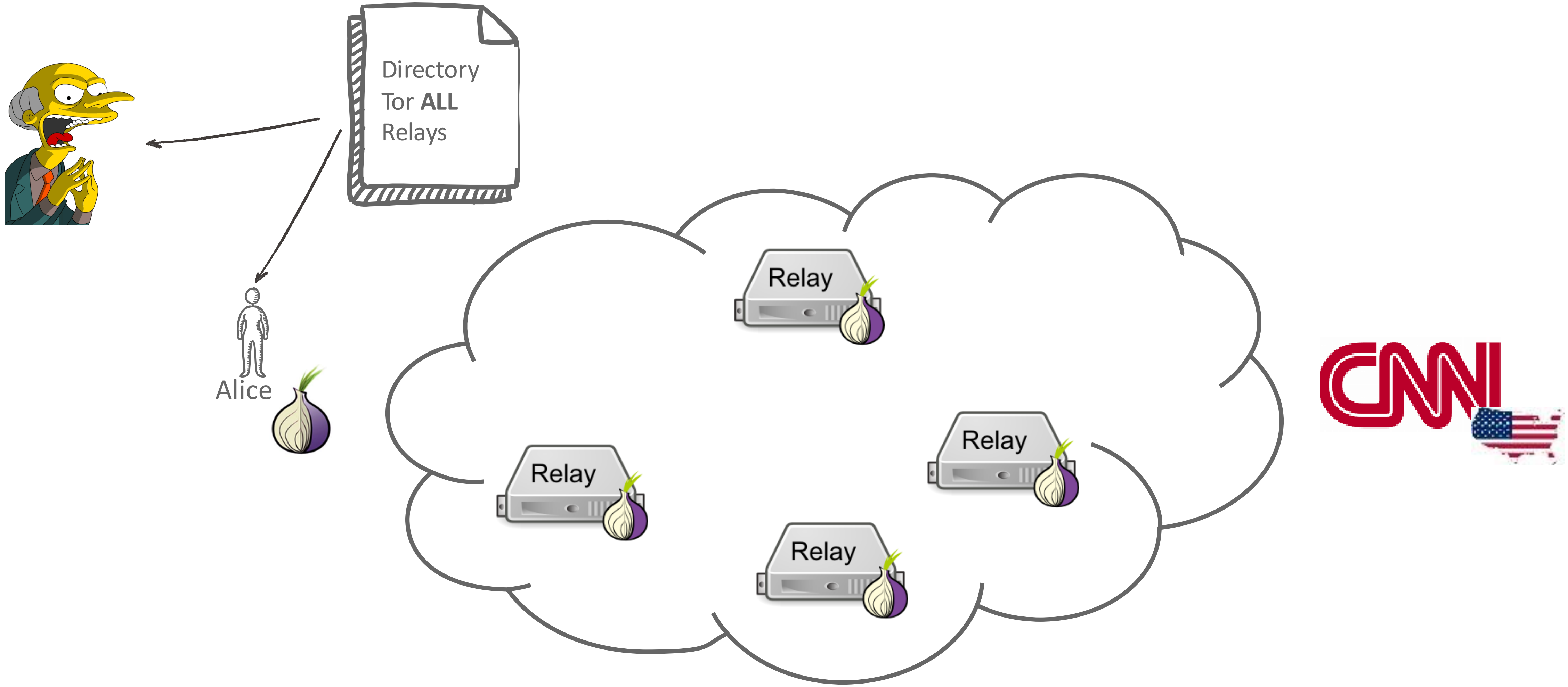
MORIA1 - 128.31.0.39 - RELAY AUTHORITY
TOR26 - 86.59.21.38 - RELAY AUTHORITY
DIZUM - 194.109.206.212 - RELAY AUTHORITY
TONGA - 82.94.251.203 - BRIDGE AUTHORITY
GABELMOO - 131.188.40.189 - RELAY AUTHORITY
DANNENBERG - 193.23.244.244 - RELAY AUTHORITY
URRAS - 208.83.223.34 - RELAY AUTHORITY
MAATUSKA - 171.25.193.9 - RELAY AUTHORITY
FARAVAHAR - 154.35.175.225 - RELAY AUTHORITY
LONGCLAW - 199.254.238.52 - RELAY AUTHORITY

https://metrics.torproject.org/collector.html

Every hour:

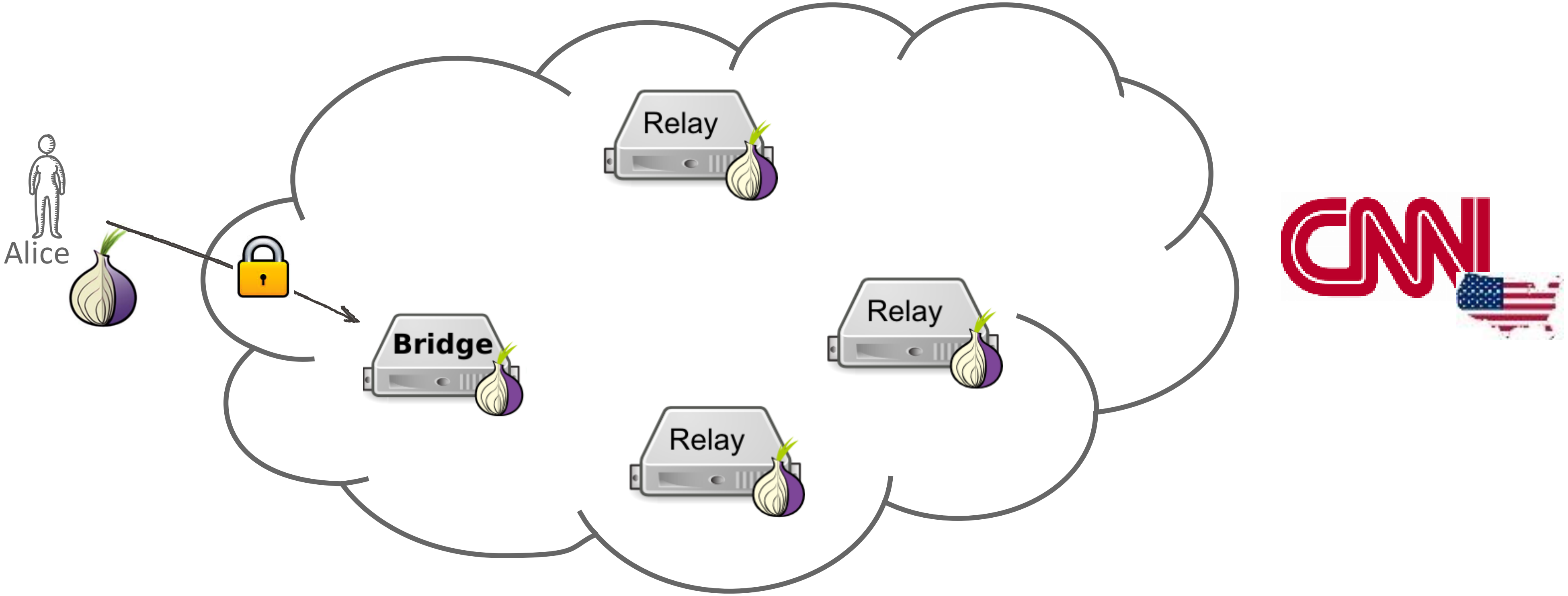
- Directory Authorities (DAs) compile a list of all known relays & flags & stuff
- DAs submits this “status-vote” to all the other authorities
- DAs combine parameters, sign and send to the other DA’s
- There **should** be a majority agreeing on the data -> **consensus**
- **Consensus** published by each DA

Can an adversary block Tor?



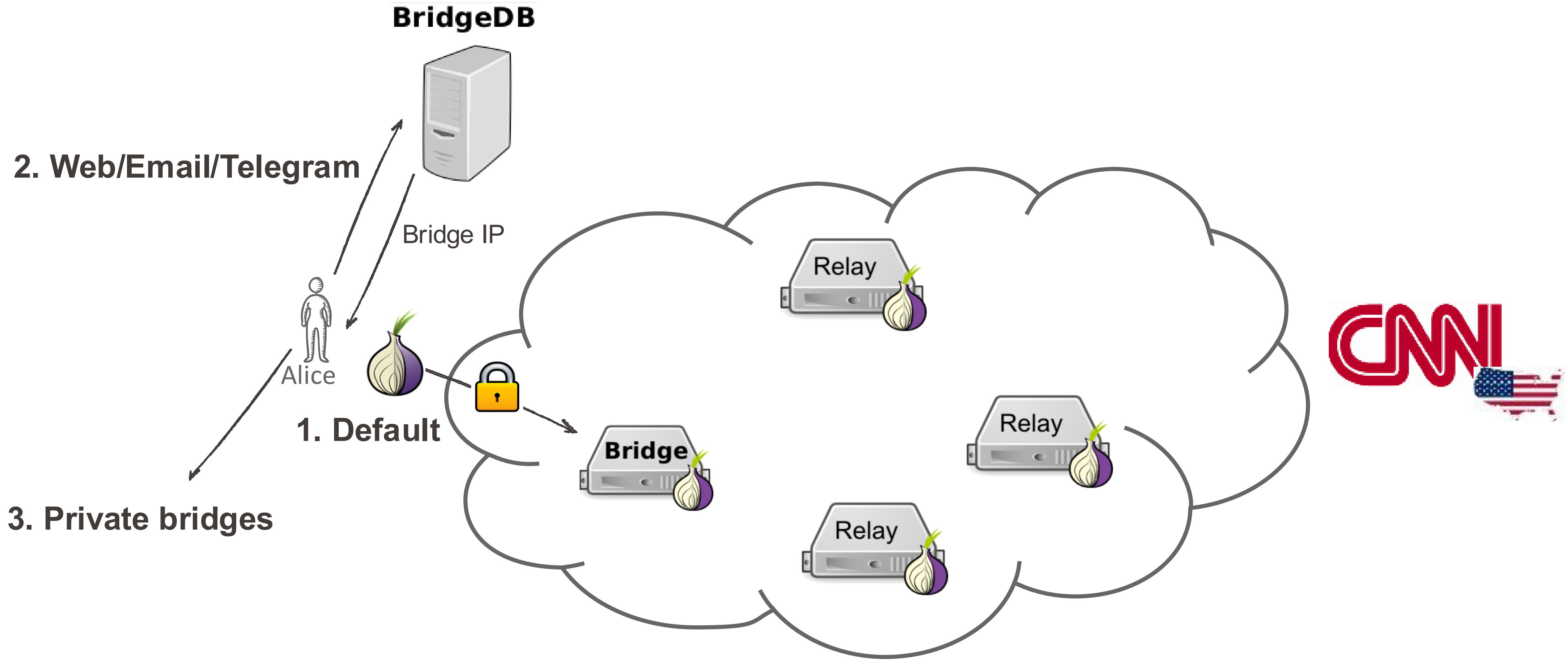
Tor bridges

Tor bridges: Onion routers whose IP is not publicly listed



Tor bridges

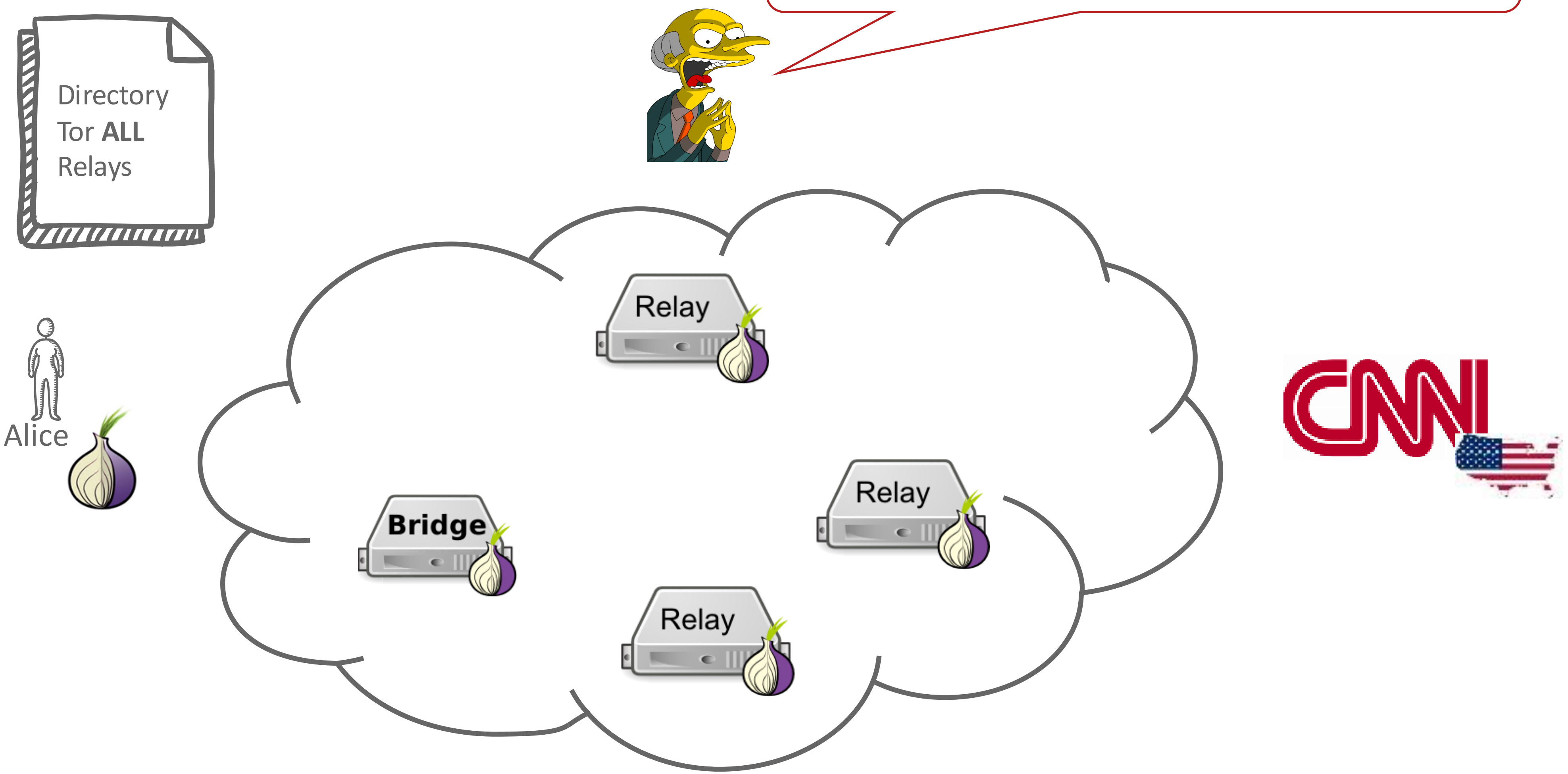
How to find them



Tor bridges

Enumeration by censors

Option 1: Via bulk emails and Tor's https server
Option 2: Malicious middle routers



- Z. Ling, J. Luo, W. Yu, M. Yang, and X. Fu. Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery. In IEEE INFOCOM, 2012

Tor bridges
Enumeration by censors

Two open Tor issues that censors can leverage to discover bridges

Issue 1: Vanilla Tor Certificates

- Vanilla Tor uses TLS handshake
- Easy to spot certificates
- It won’t be fixed

Issue 2: Open Onion Routing Port

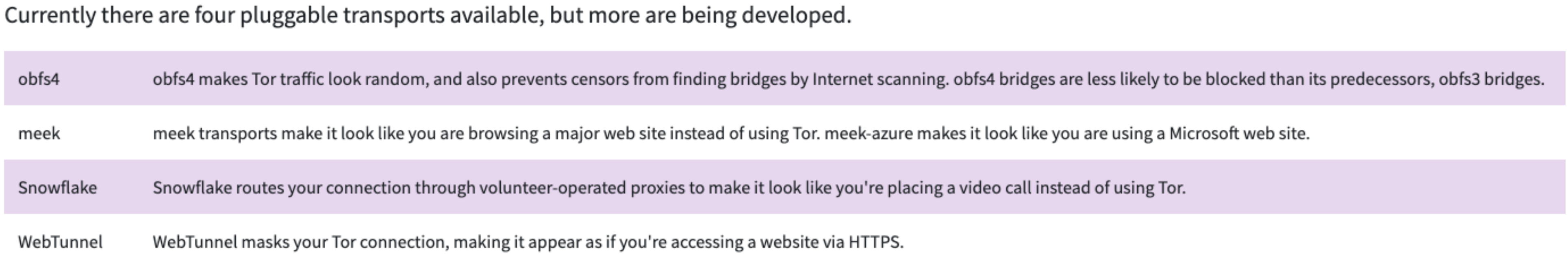
- Bridges have open OR Port with Vanilla Tor
- Even if they do not offer Vanilla Tor
- Difficult to fix

Srdjan Matic, Carmela Troncoso, and Juan Caballero. Dissecting Tor Bridges: a Security Evaluation of their Private and Public Infrastructures. NDSS 2017

That was establishment, what about conversation?

Pluggable transports

TorProject.org



Currently there are four pluggable transports available, but more are being developed.

obfs4	obfs4 makes Tor traffic look random, and also prevents censors from finding bridges by Internet scanning. obfs4 bridges are less likely to be blocked than its predecessors, obfs3 bridges.
meek	meek transports make it look like you are browsing a major web site instead of using Tor. meek-azure makes it look like you are using a Microsoft web site.
Snowflake	Snowflake routes your connection through volunteer-operated proxies to make it look like you're placing a video call instead of using Tor.
WebTunnel	WebTunnel masks your Tor connection, making it appear as if you're accessing a website via HTTPS.

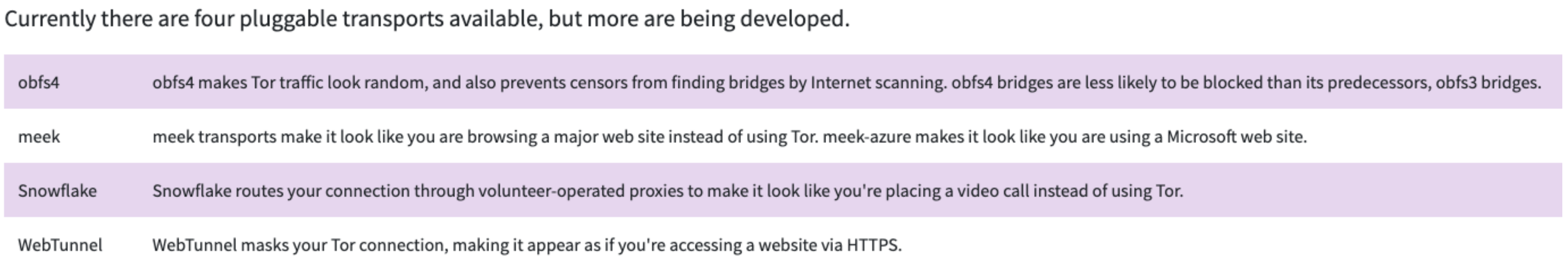
2025

- https://tb-manual.torproject.org/circumvention/

That was establishment, what about conversation?

Pluggable transports

TorProject.org



Currently there are four pluggable transports available, but more are being developed.

obfs4	obfs4 makes Tor traffic look random, and also prevents censors from finding bridges by Internet scanning. obfs4 bridges are less likely to be blocked than its predecessors, obfs3 bridges.
meek	meek transports make it look like you are browsing a major web site instead of using Tor. meek-azure makes it look like you are using a Microsoft web site.
Snowflake	Snowflake routes your connection through volunteer-operated proxies to make it look like you're placing a video call instead of using Tor.
WebTunnel	WebTunnel masks your Tor connection, making it appear as if you're accessing a website via HTTPS.

2025

Content/flow obfuscation

- **Mimicry**: look like whitelisted (or not blacklisted)
- **Tunneling**: tunnel traffic through unblocked application
- **Covert channel**: hide censored traffic on images, voice, emails,…

https://tb-manual.torproject.org/circumvention/

Mimicry: Look like not blacklisted

ScrambleSuit

Key features

- Defense against active probing
- Pseudo-random payload
- Polymorphic

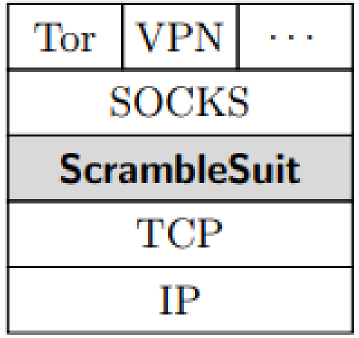


Figure 1: ScrambleSuit's protocol stack.

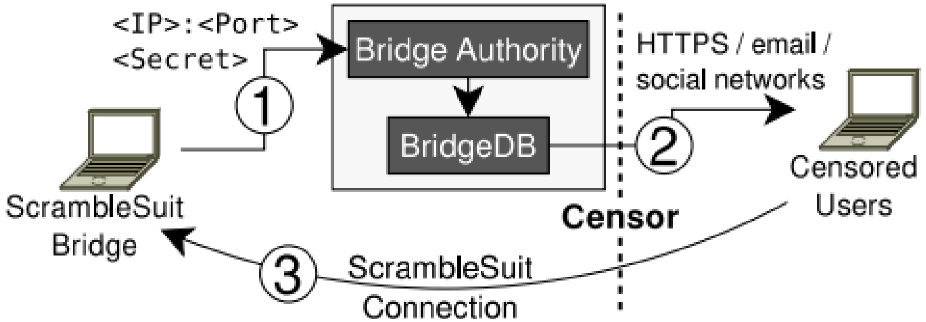
Philipp Winter, Tobias Pulls, and Juergen Fuss: ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship (WPES13)

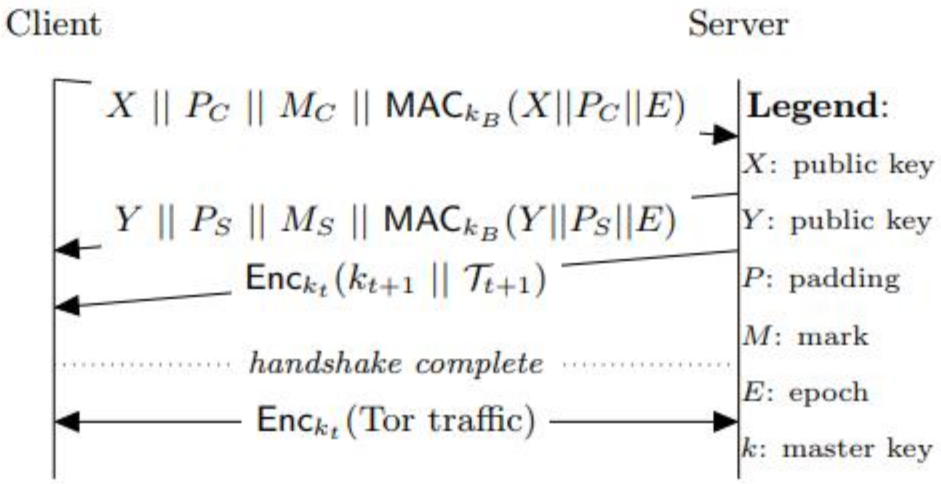
ScrambleSuit
Active attacks defense

Defense against active probing:

Protection against active probing attacks by requiring a *shared secret* between the client and the server.

This secret is communicated *out-of-band* via Tor's BridgeDB.




Client
Server
$X \,||\, P_C \,||\, M_C \,||\, \mathsf{MAC}_{k_B}(X||P_C||E)$
$Y \,||\, P_S \,||\, M_S \,||\, \mathsf{MAC}_{k_B}(Y||P_S||E)$
$\mathsf{Enc}_{k_t}(k_{t+1} \,||\, \mathcal{T}_{t+1})$
handshake complete
$\mathsf{Enc}_{k_t}(\text{Tor traffic})$
Legend:
X: public key
Y: public key
P: padding
M: mark
E: epoch
k: master key

ScrambleSuit

Destroy patterns

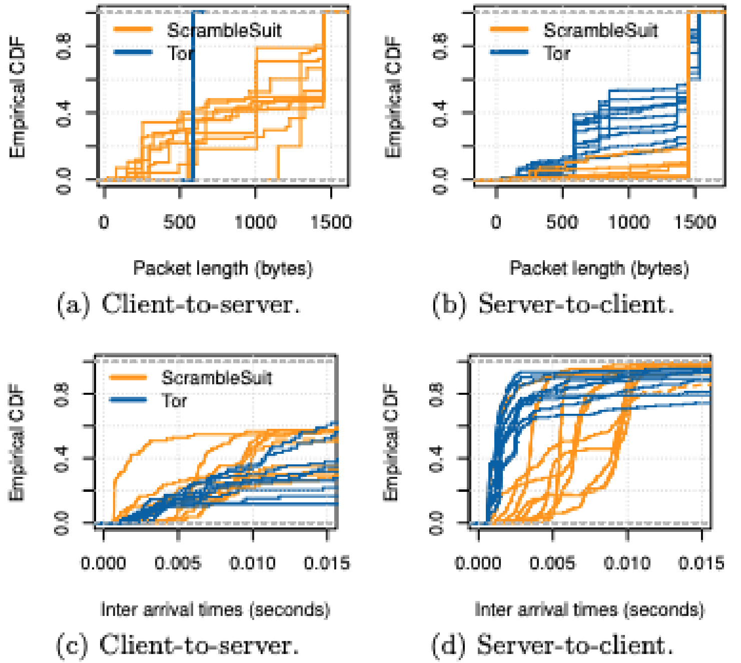
(Lightweight*) Traffic analysis resistance through protocol polymorphism:

Every ScrambleSuit server generates its own and unique “protocol shape” by modifying:

- **packet lengths**
- **inter-arrival times**

How:

1. Generate a random seed shared between ScrambleSuit server and client.
2. Both sides use seed to generate two discrete probability distributions
3. Use distributions to shape traffic



(a) Client-to-server. (b) Server-to-client.

(c) Client-to-server. (d) Server-to-client.

Figure 10: Tor’s and ScrambleSuit’s packet length distribution and inter-arrival times for both, client-to-server and server-to-client traffic.

*Inexpensive measures that diminish but do **not defeat** traffic analysis attacks

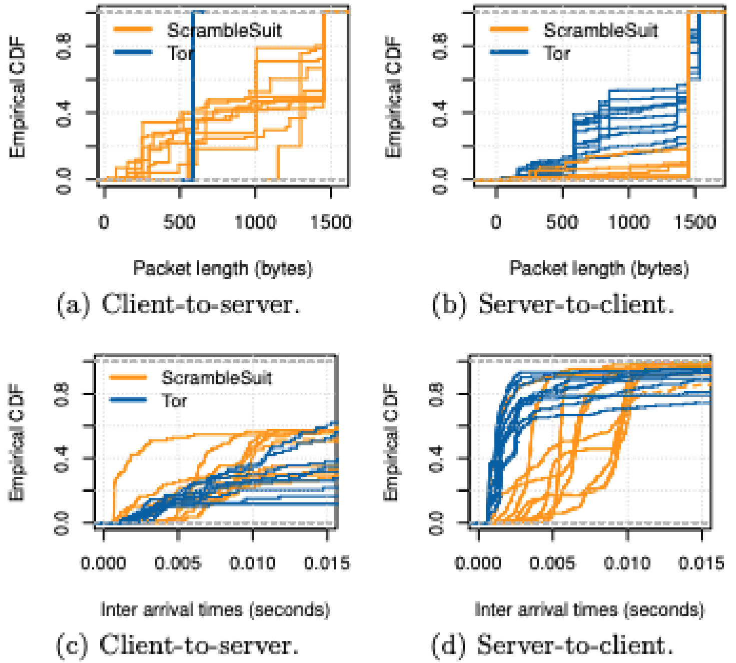
ScrambleSuit

Destroy patterns

(Lightweight*) Traffic analysis resistance through protocol polymorphism:

Evaluation: How well does this mitigate?

Winter et al., 2013: “It is difficult to evaluate the effectiveness of our obfuscation techniques since ScrambleSuit **does not have a cover protocol to mimic**. Otherwise, our evaluation would simply investigate the similarity between our protocol and its cover protocol. Instead of measuring ScrambleSuit’s closeness to a mimicked protocol, we measure the deviation from its transported application, i.e., Tor. **Intuitively, higher deviation would imply better obfuscation.**”



(a) Client-to-server.

(b) Server-to-client.

(c) Client-to-server.

(d) Server-to-client.

Figure 10: Tor’s and ScrambleSuit’s packet length distribution and inter-arrival times for both, client-to-server and server-to-client traffic.

ScrambleSuit

Destroy patterns

It looks like nothing…
… but nothing looks like it!

- Wang, L., Dyer, K. P., Akella, A., Ristenpart, T., & Shrimpton, T. Seeing through network-protocol obfuscation. CCS 2015.

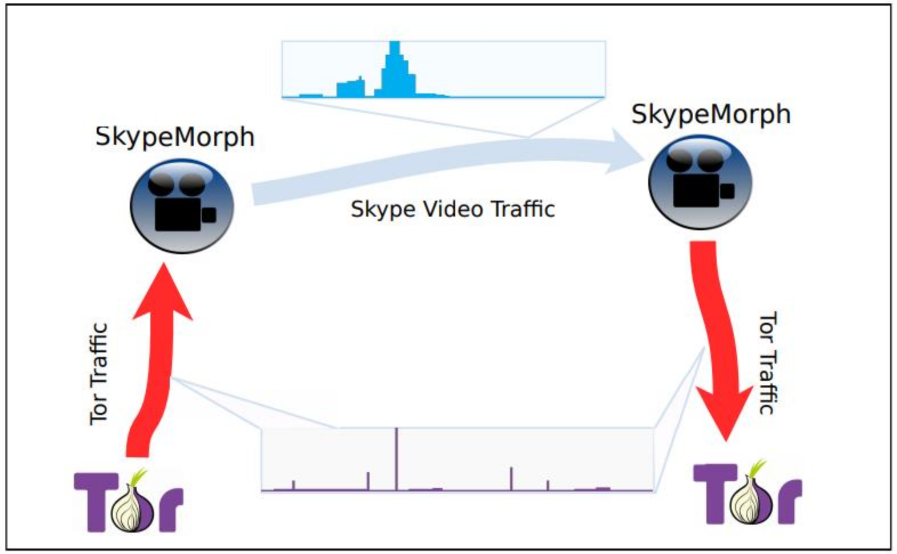
Mimicry: Look like whitelisted

SkypeMorph

Goal: make it difficult for the censor to distinguish between the obfuscated bridge connections and whitelisted traffic using statistical comparisons of flow features

How: Tor clients obfuscate their messages to Tor bridge server in a widely used protocol over the Internet.

Skype video calls as target protocol



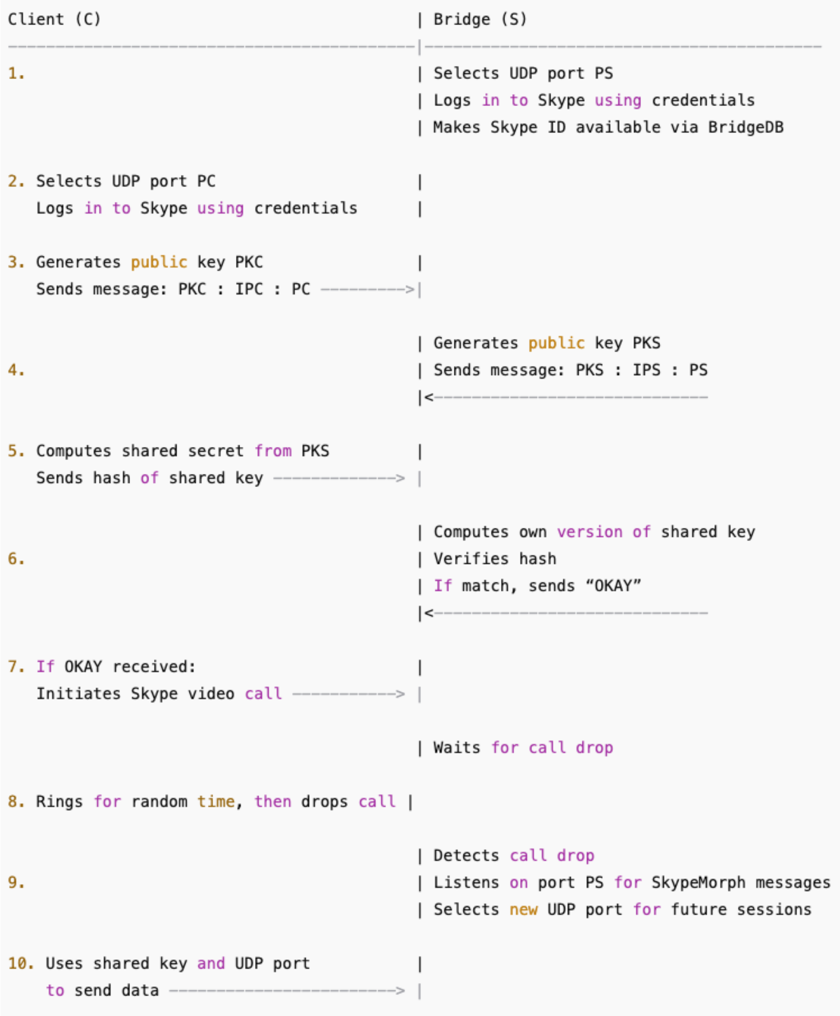
H. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In CCS, 2012.

SkypeMorph
Setup

Conversation setup:
Protocol between client and bridge using the Skype API to establish conversation

- Use UDP protocol as vanilla Skype
- Initiate call that is then dropped
- Use agreed UDP port to exchange packets

→ Now can send packets to each other.
Is there anything else client and bridge need to take into account?



```
Client (C)                                   | Bridge (S)
---------------------------------------------|--------------------------------------------
1.                                           | Selects UDP port PS
                                             | Logs in to Skype using credentials
                                             | Makes Skype ID available via BridgeDB

2. Selects UDP port PC                       |
   Logs in to Skype using credentials        |

3. Generates public key PKC                  |
   Sends message: PKC : IPC : PC ---------->|

                                             | Generates public key PKS
4.                                           | Sends message: PKS : IPS : PS
                                             |<------------------------------

5. Computes shared secret from PKS           |
   Sends hash of shared key -------------> |

                                             | Computes own version of shared key
6.                                           | Verifies hash
                                             | If match, sends “OKAY”
                                             |<------------------------------

7. If OKAY received:                         |
   Initiates Skype video call -----------> |

                                             | Waits for call drop

8. Rings for random time, then drops call |

                                             | Detects call drop
9.                                           | Listens on port PS for SkypeMorph messages
                                             | Selects new UDP port for future sessions

10. Uses shared key and UDP port             |
    to send data -------------------------> |
```

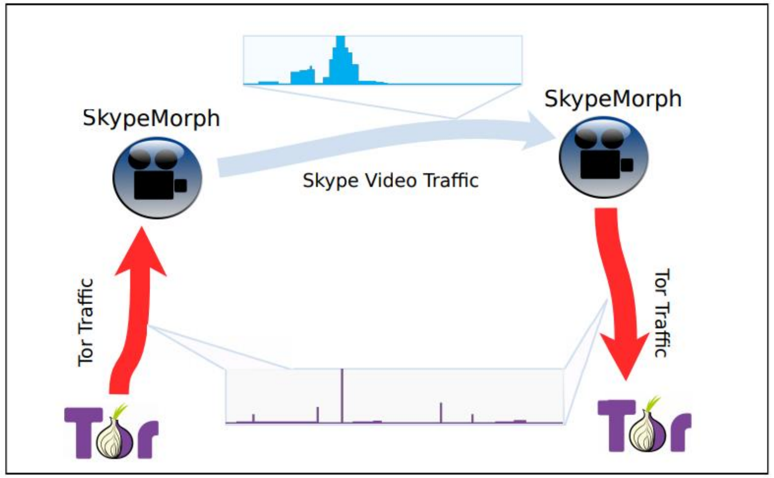
H. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In CCS, 2012.

SkypeMorph
Traffic shaping

Exchange: Send Tor TLS data over encrypted channel, masquerading it as Skype video

- Mimic skype traffic
  - Packet size
  - Inter-arrival times

→ **Packets are sent (statistically) following Skype patterns**

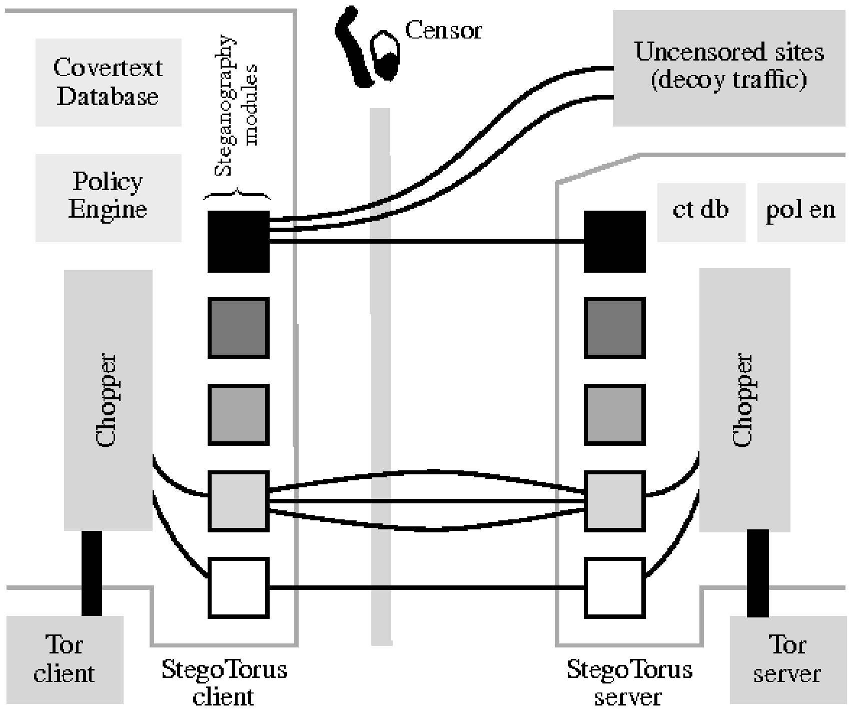


H. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In CCS, 2012.

Mimicry: Look like whitelisted
StegoTorus

Goal: Make it difficult for the censor to distinguish between the obfuscated bridge connections and whitelisted traffic using statistical comparisons of flow features

How: Chops Tor traffic and sends it through different connections (HTTP, Skype, VoIP…)

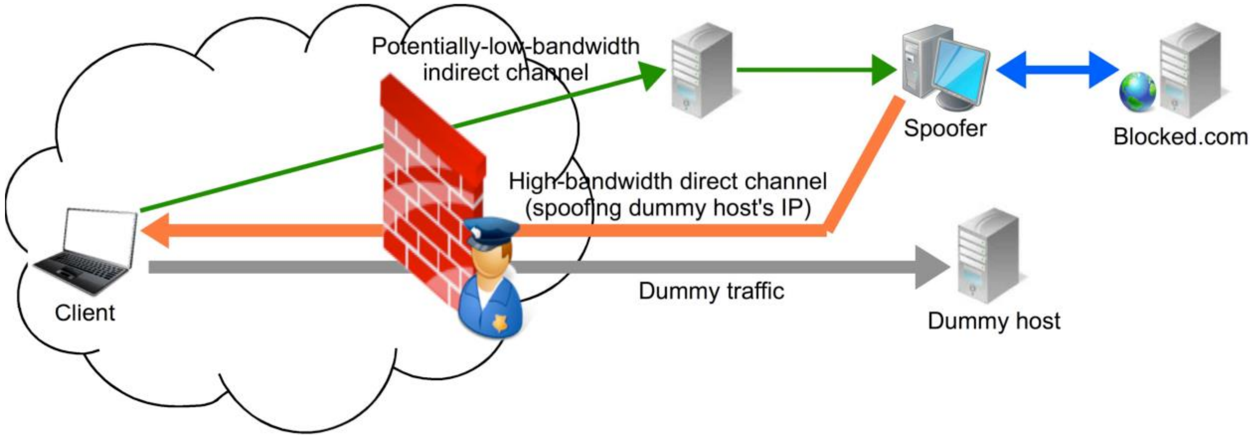


Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. StegoTorus: a camouflage proxy for the Tor anonymity system. CCS 2012

Mimicry: Look like whitelisted CensorSpoofer

Goal: Obfuscate traffic patterns through mimicry

How: Standalone system
(1) IP Spoofing to obfuscate server’s identity
(2) Mimics VoIP traffic to obfuscate traffic patterns


Potentially-low-bandwidth indirect channel
Spoofer
Blocked.com
High-bandwidth direct channel (spoofing dummy host's IP)
Client
Dummy traffic
Dummy host

Qiyan Wang, Xun Gong, Giang T.K. Nguyen, Amir Houmansadr, and Nikita Borisov
CensorSpoofer: asymmetric communication using IP spoofing for censorship-resistant web browsing. CCS 2012

Mimicry: Look like whitelisted

Goal: Make it difficult for the censor to distinguish between black- and whitelisted traffic using statistical comparisons of flow features

How: Imitate common protocols like HTTP and Skype

Systems: SkypeMorph, StegoTorus, CensorSpoofer…

→ Parrot circumvention systems

Mimicry: Look like whitelisted

The parrot is dead

Goal: Make it difficult for the censor to distinguish between black- and whitelisted traffic using statistical comparisons of flow features

How: Imitate common protocols like HTTP and Skype

Systems: SkypeMorph, StegoTorus, CensorSpoofer…

→ **“Unobservability by imitation” is fundamentally flawed.**

Houmansadr, Brubaker, and Shmatikov: The Parrot is Dead: Observing Unobservable Network Communications
IEEE Symposium Security and Privacy 2013

Censorship

🔍 Step 1: Fingerprinting

Destination: ✅ Tor (other anon comms)

IP addresses, hosts, ports,…

Content: ✅

Encryption

protocol strings, keywords, domains, http hosts, encrypted flows…

Flow properties: ✅ Obfuscation through mimicry

length, inter-arrival times, bursts, …

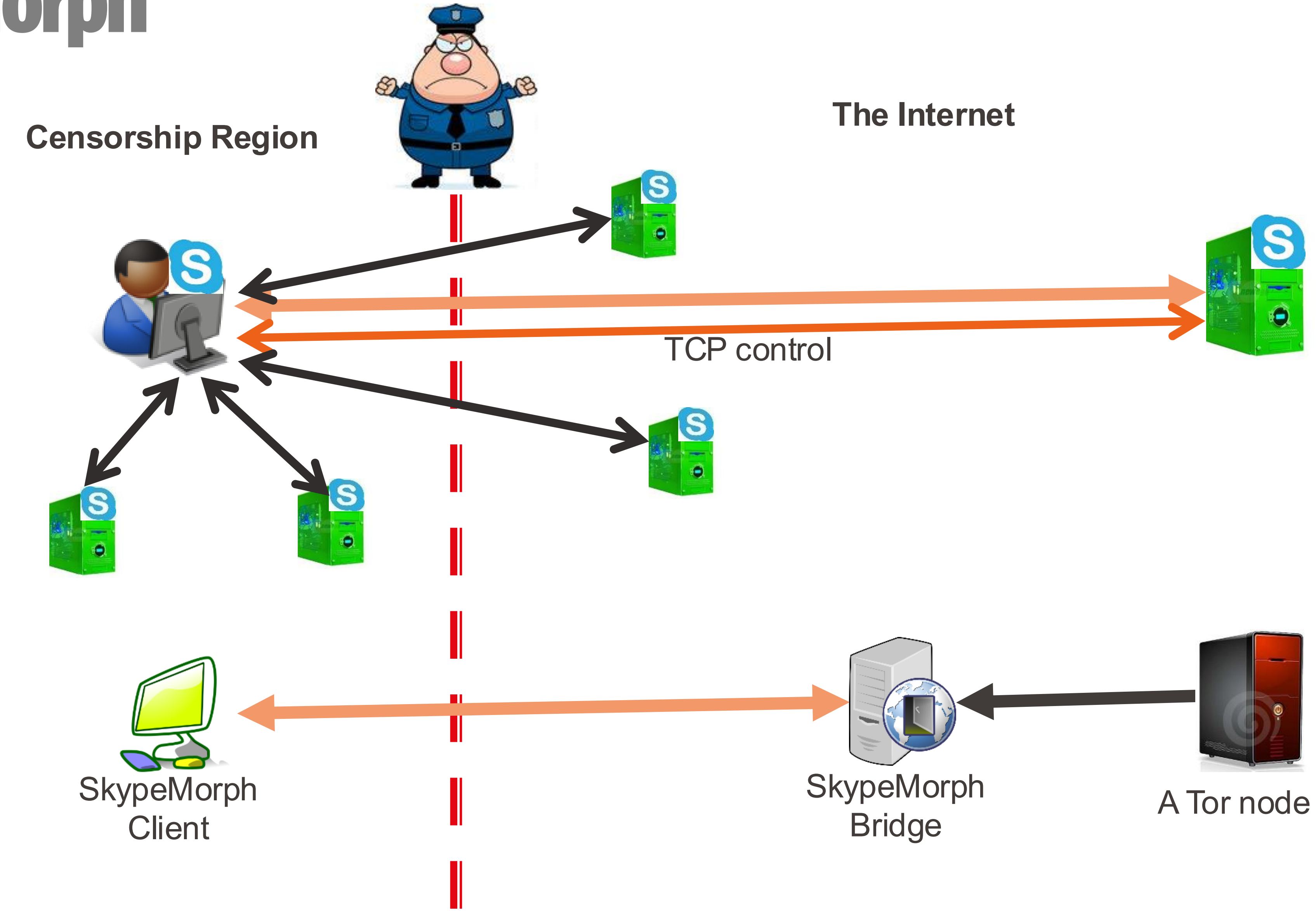
Protocol semantics:

protocol behavior (mostly active attacks)

To win, the censor needs only to find a few discrepancies

The parrot is dead

SkypeMorph



Slide by Amir Houmansadr

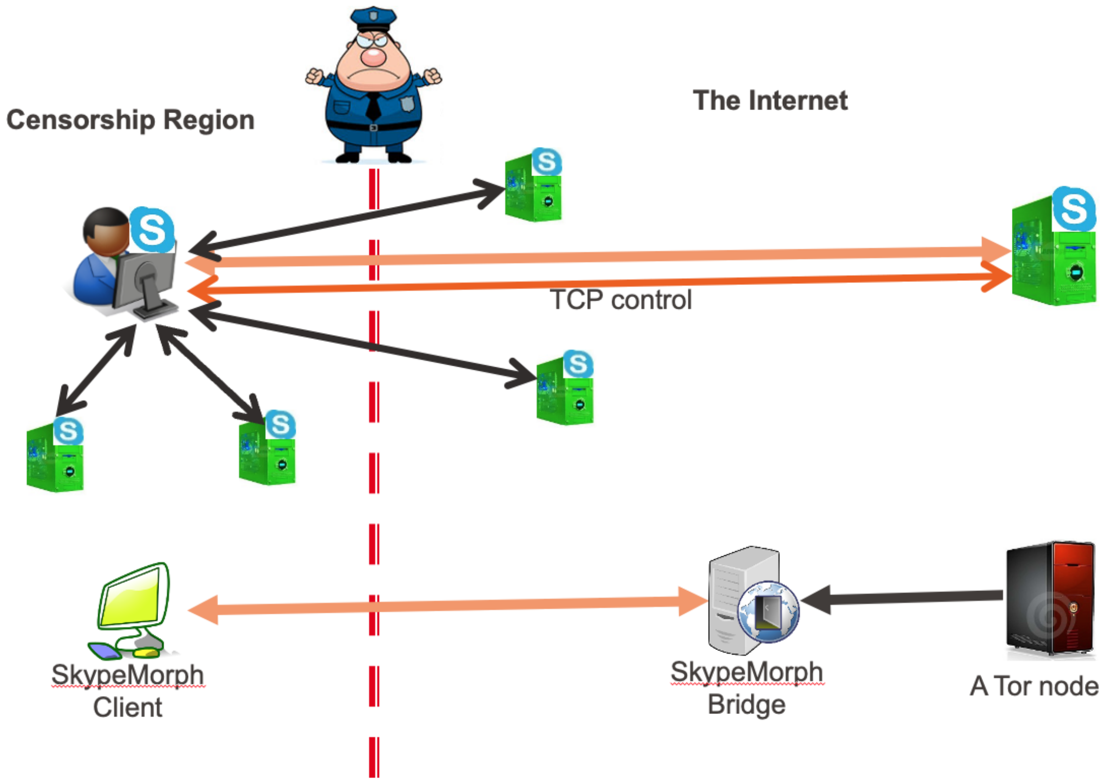
The parrot is dead
SkypeMorph/StegoTorus

Parrots mimic Skype’s traffic statistics but…

…fail to mimic much more visible aspects:

- no HTTP update traffic
- no login traffic
- no mimicry of Skype’s TCP channel

→ **Parrot systems can be distinguished from Skype even by extremely basic tests**



Slide by Amir Houmansadr

The parrot is dead

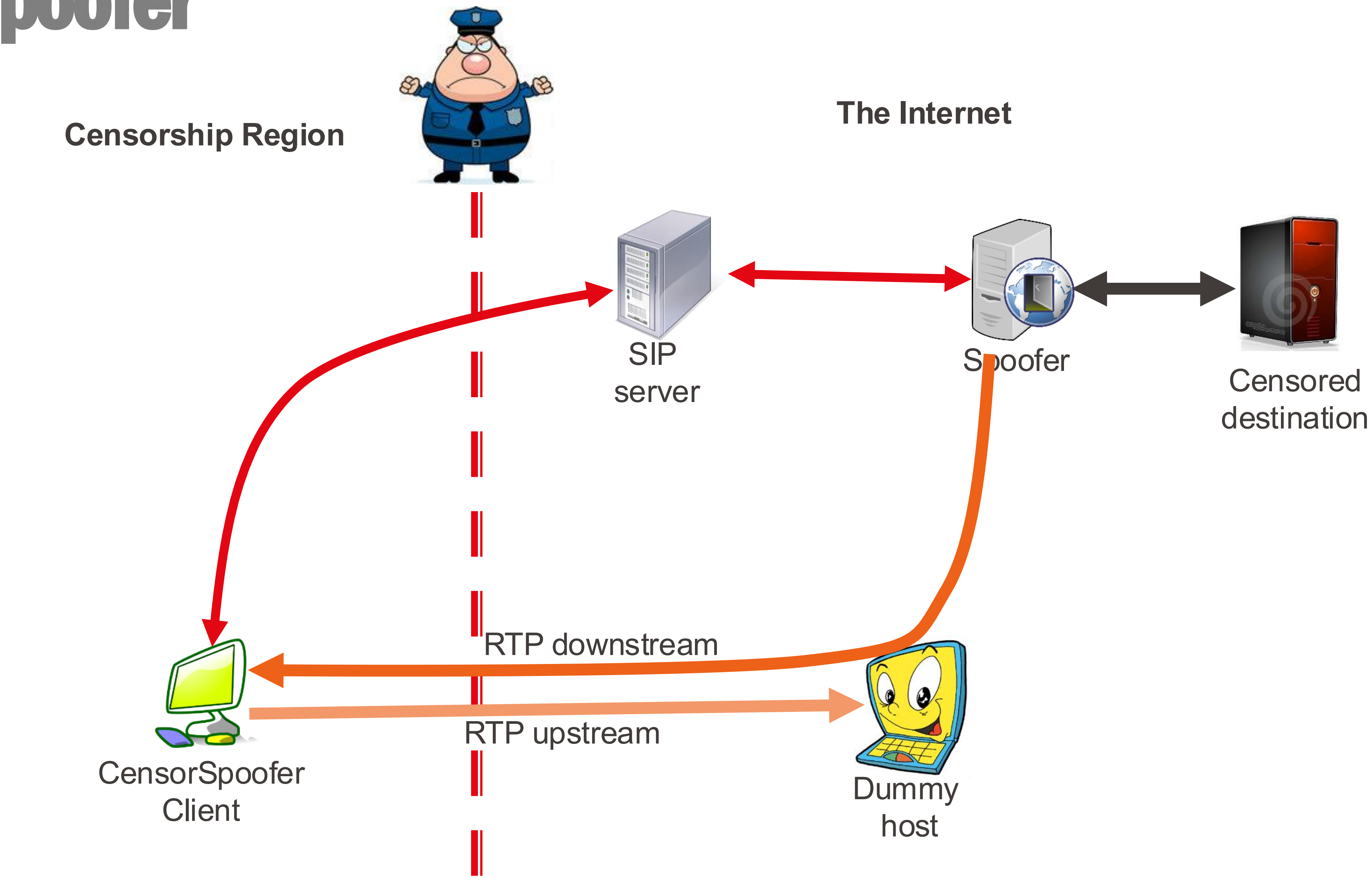
Other tests

Test	Skype	SkypeMorph+
Flush Supernode cache	Serves as a SN	Rejects all Skype messages
Drop UDP packets	Burst of packets in TCP control	No reaction
Close TCP channel	Ends the UDP stream	No reaction
Delay TCP packets	Reacts depending on the type of message	No reaction
Close TCP connection to a SN	Initiates UDP probes	No reaction
Block the default TCP port	Connects to TCP ports 80 and 443	No reaction

Slide by Amir Houmansadr

The parrot is dead

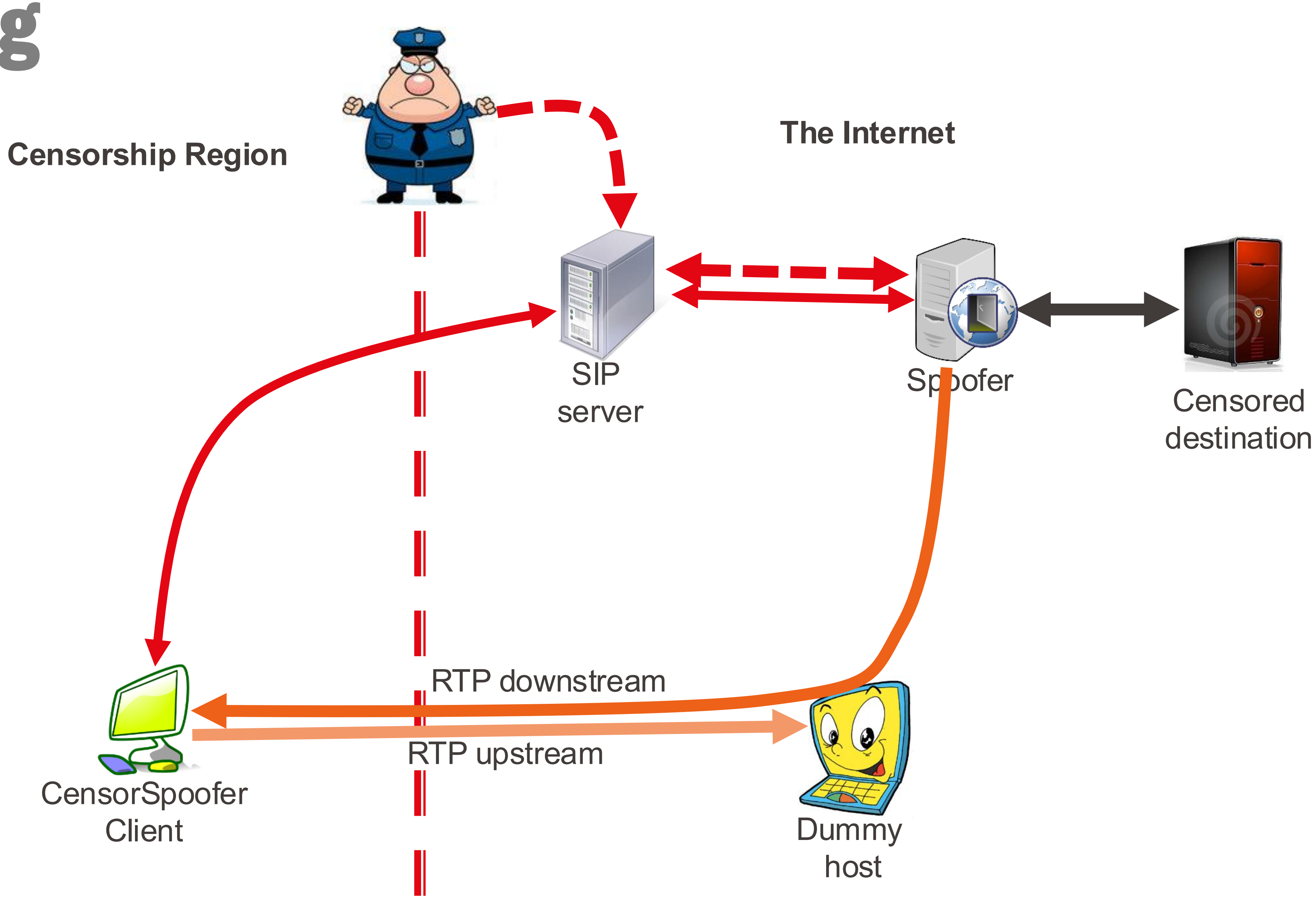
CensorSpoofer



Slide by Amir Houmansadr

The parrot is dead

SIP probing



The parrot is dead
Imitation Requirements

Parrot needs to mimic…

Protocol in its entirety	Reaction to errors and network conditions	Typical traffic
Correct protocol	Errors	Content
Side protocols	Network	Patterns
Intra dependencies		Users
Inter dependencies		Geolocalisation

Slide by Amir Houmansadr

Solution: do not imitate, be!!

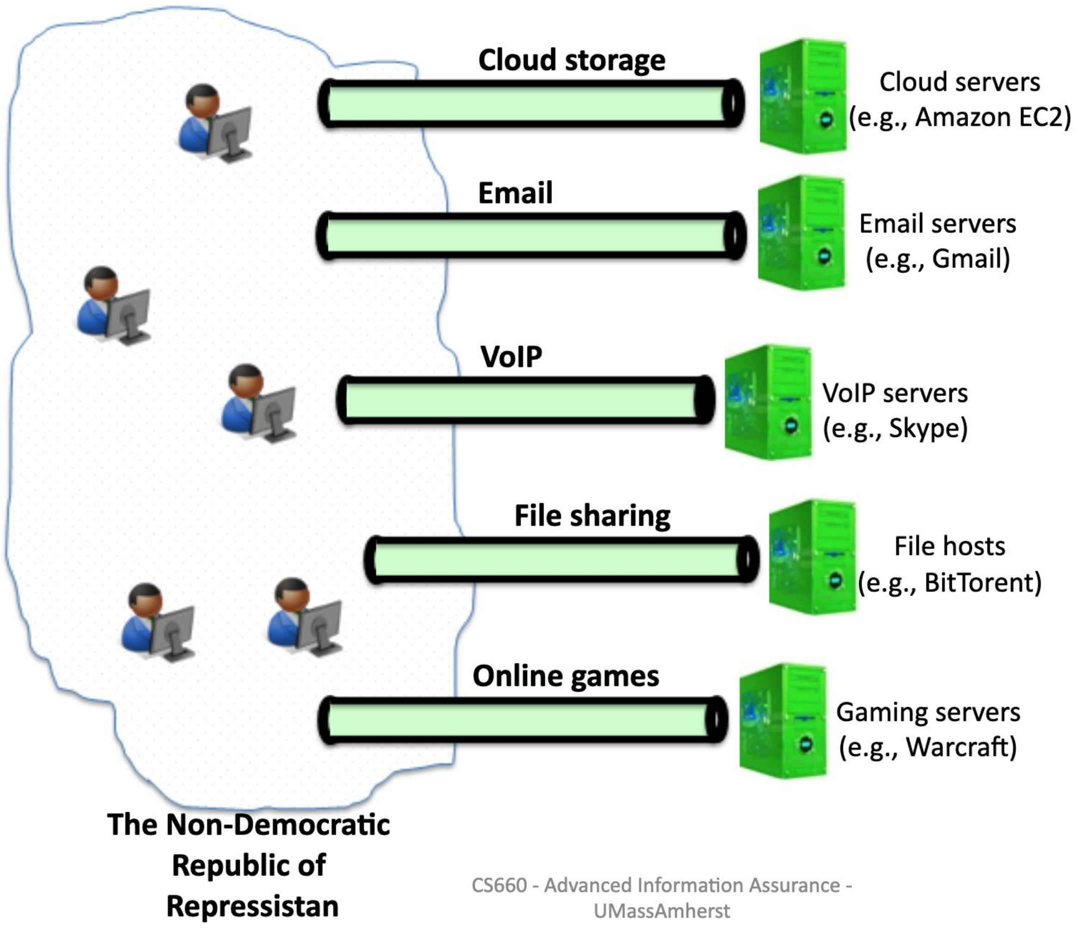
From parrots…

…to parasites

Hide-within circumvention

Idea: We already have a lot of encrypted channels…

→ Hide censored traffic within!



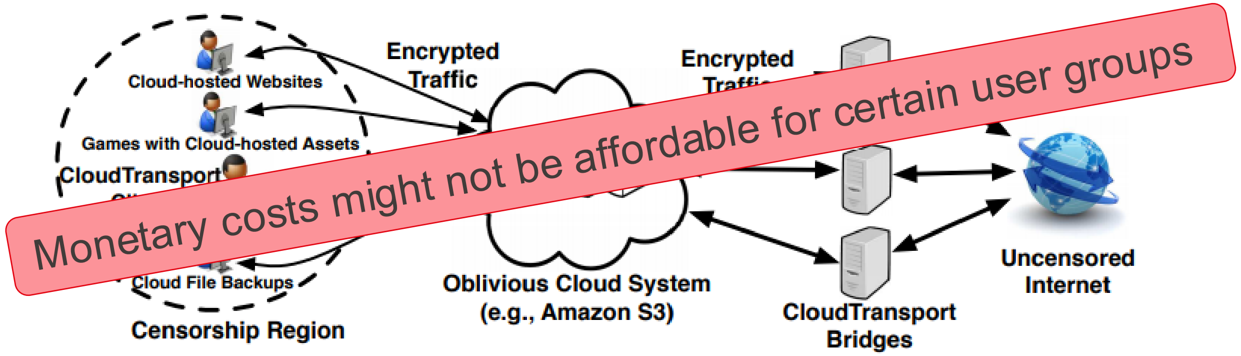
CS660 - Advanced Information Assurance -
UMassAmherst

Hide-within circumvention
CloudTransport

Goal: raise economic and social costs of censorship by forcing the censors to use statistical traffic analysis and other computationally intensive techniques

How: Hide censored traffic within existing encrypted channels

1) Select a Cloud provider: one that does provide other non-censored services
2) Create a rendez-vous account with the Cloud [“can’t” be censored!]
3) Select a CloudTransport bridge and send to it the rendez-vous credentials (Dead drop or Out-of-band)
4) To send data, the client puts it on the Cloud and the Bridge transmits it to the destination



Monetary costs might not be affordable for certain user groups

Chad Brubaker, Amir Houmansadr, and Vitaly Shmatikov. CloudTransport: Using Cloud Storage for Censorship-Resistant Networking (PETS 2014)

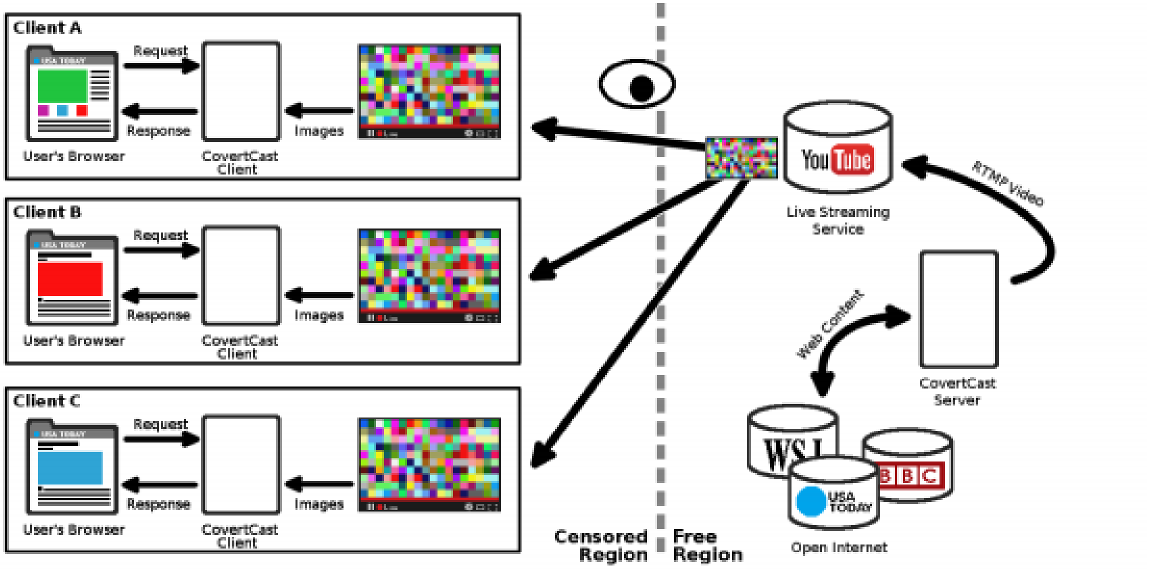
Hide-within circumvention

CovertCast

Goal: raise economic and social costs of censorship by forcing the censors to use statistical traffic analysis and other computationally intensive techniques

How: Hide censored traffic within existing encrypted channels

1) CovertCast server initiates live stream
2) Server crawls censored site, encodes content into images and boradcasts images via live stream
3) CovertCast client demodulates images back into Web content



Richard McPherson, Amir Houmansadr, and Vitaly Shmatikov. CovertCast: Using Live Streaming to Evade Internet Censorship (PoPETS 2016)

Up to here

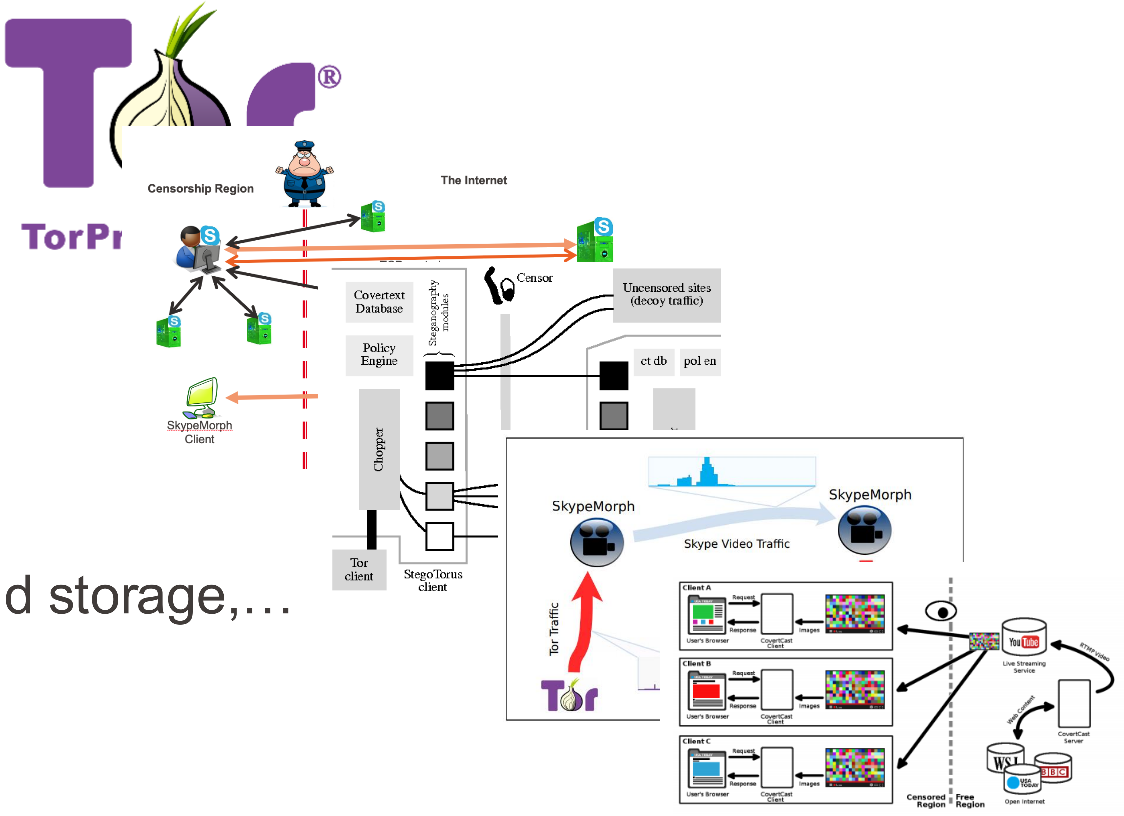
All systems work **at the application layer:**

Overlay networks: Onion routing & obfuscation

Reuse other infrastructures:

Parrots: Imitation of Skype, P2P,…

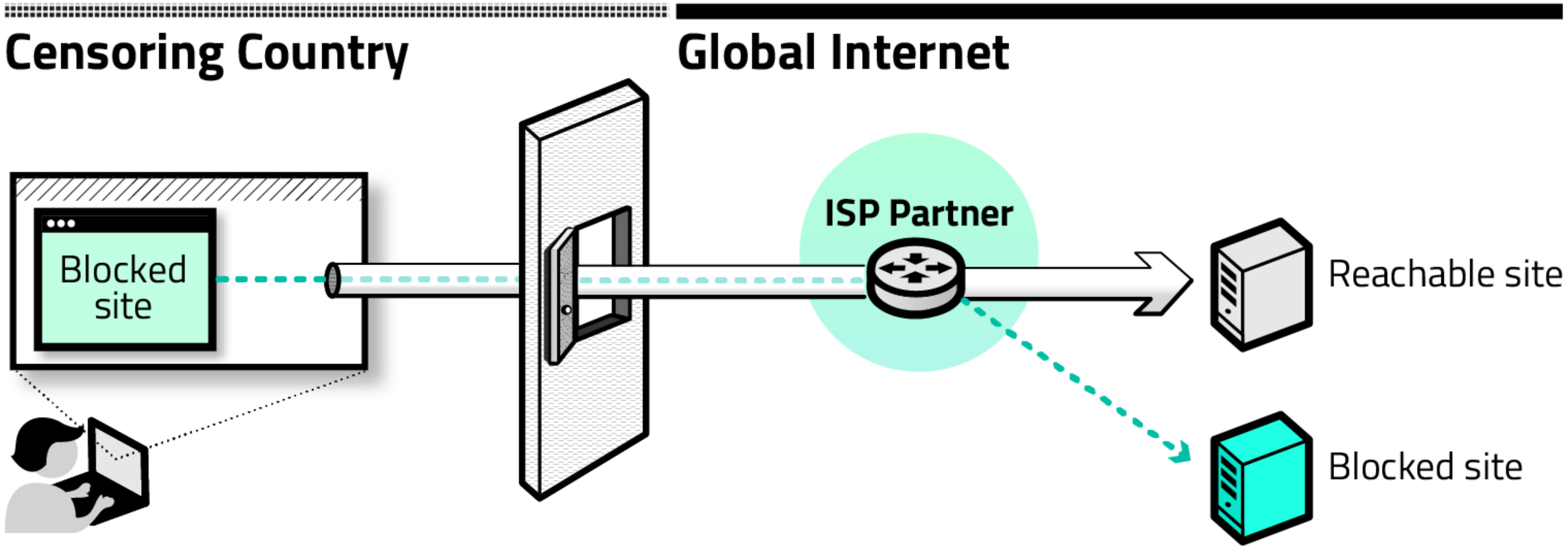
Hide-within: Hide within live streams, cloud storage,…



→ End the cat-and-mouse game of application-layer censorship systems

Decoy routing (refraction networking)

Motivation: End the cat-and-mouse game of application-layer censorship systems



1. User requests a blocked site

2. Client software requests a reachable site

3. Censor allows the request to pass through

4. ISP partner ***refracts*** the request to the blocked site

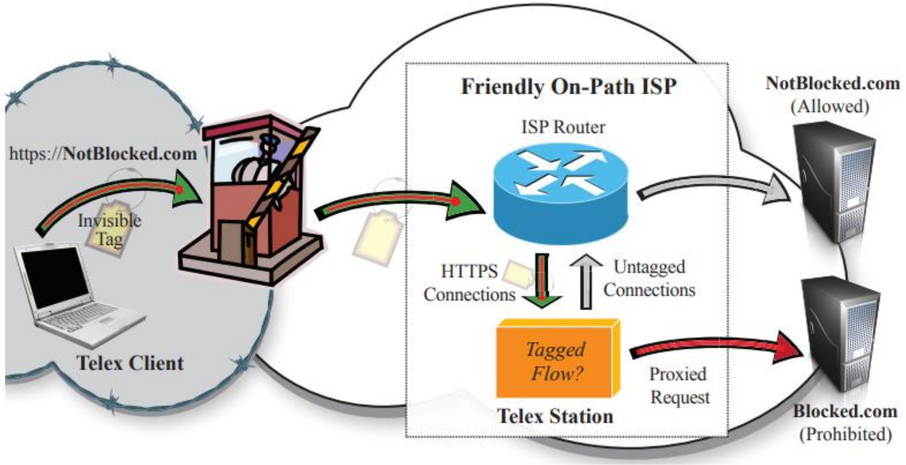
https://refraction.network

https://refraction.network

Decoy routing

Example: Telex

- Operates in the **network infrastructure** — at any ISP between the censor's network and non-blocked
    - **State-level response** to state-level censorship.
- Repurposes **deep-packet inspection** to circumvent censorship.
- **No secrets** to communicate to users in advance
- Focuses on **avoiding detection**

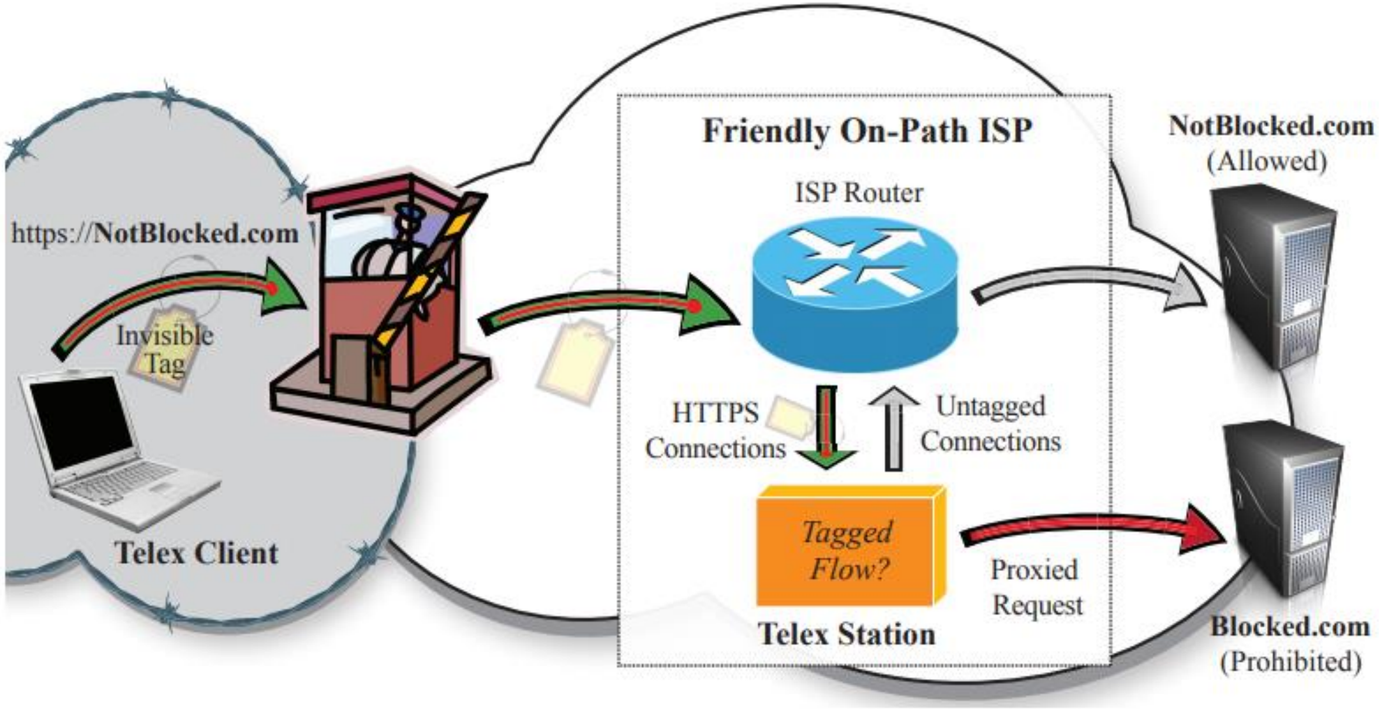


"A friendly man-in-the-middle"

Eric Wustrow, Scott Wolchok, Ian Goldberg and J. Alex Halderman. Telex: Anticensorship in the Network Infrastructure (USENIX 2011)

Decoy routing

Example: Telex

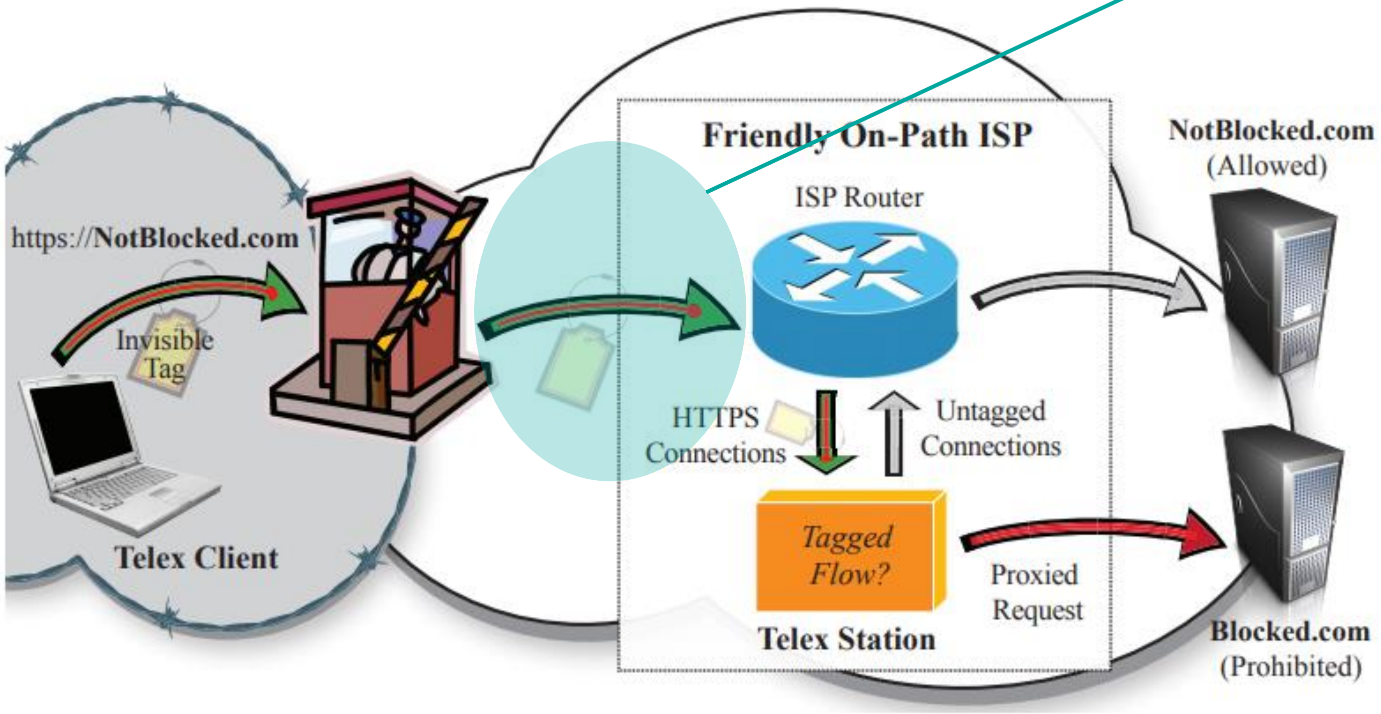


- Eric Wustrow, Scott Wolchok, Ian Goldberg and J. Alex Halderman. Telex: Anticensorship in the Network Infrastructure (USENIX 2011)

Decoy routing

Example: Telex

Tag: looks like a random nonce in the TLS handshake

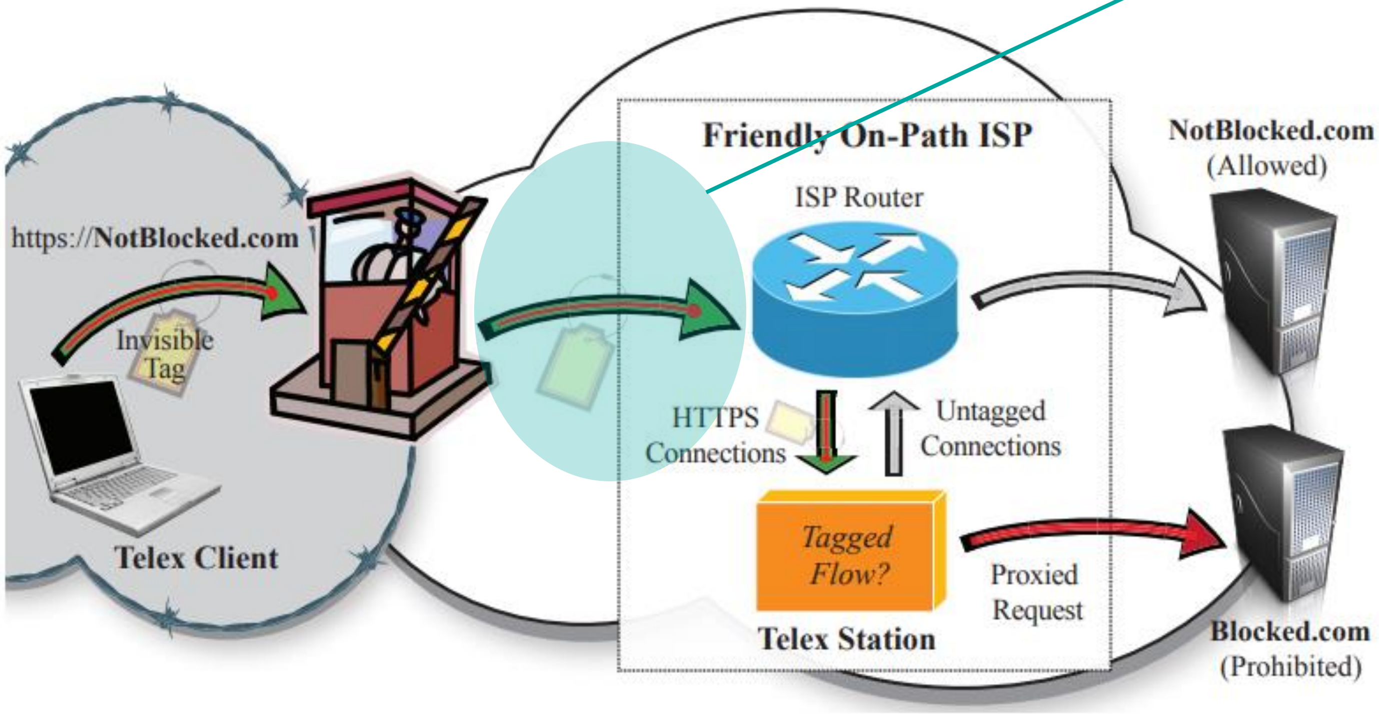


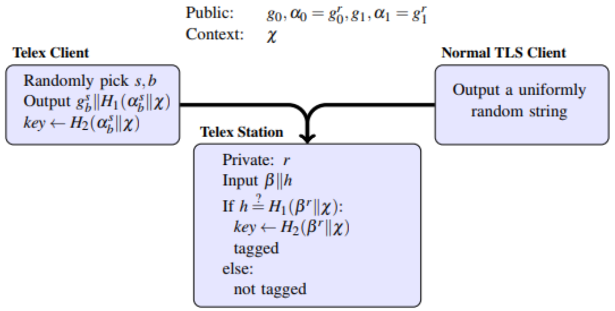
- Eric Wustrow, Scott Wolchok, Ian Goldberg and J. Alex Halderman. Telex: Anticensorship in the Network Infrastructure (USENIX 2011)

Decoy routing

Example: Telex

Tag: looks like a random nonce in the TLS handshake





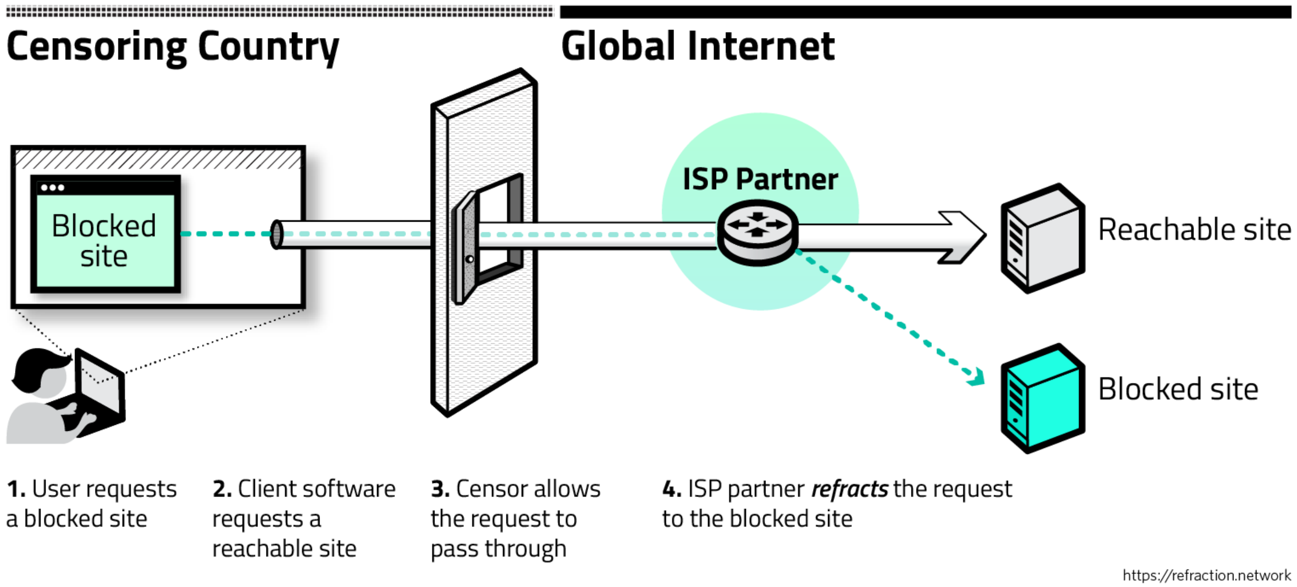
Public: $g_0, \alpha_0 = g_0^r, g_1, \alpha_1 = g_1^r$
Context: χ
Telex Client
Randomly pick s, b
Output $g_b^s \| H_1(\alpha_b^s \| \chi)$
$key \leftarrow H_2(\alpha_b^s \| \chi)$
Normal TLS Client
Output a uniformly random string
Telex Station
Private: r
Input $\beta \| h$
If $h \overset{?}{=} H_1(\beta^r \| \chi)$:
$key \leftarrow H_2(\beta^r \| \chi)$
tagged
else:
not tagged

- Eric Wustrow, Scott Wolchok, Ian Goldberg and J. Alex Halderman. Telex: Anticensorship in the Network Infrastructure (USENIX 2011)

Decoy routing

Routing attacks

Motivation: End the cat-and-mouse game of application-layer censorship systems



1. User requests a blocked site

2. Client software requests a reachable site

3. Censor allows the request to pass through

4. ISP partner ***refracts*** the request to the blocked site

https://refraction.network

Have we really reached the end?

Meet routing capable adversaries

A censoring authority who is capable of controlling how packets originating from its network are routed

- Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing Around Decoys (CCS 2012)

Decoy routing

Routing attacks

Through routing attacks a routing capable adversary can:

- Enumerate the participating decoy routers
- Successfully avoid sending traffic along routes containing these routers with little or no adverse effects
- Identify users of these schemes through active and passive attacks
- (In some cases) probabilistically identify connections to targeted destinations.

One of the goals of Telex:
Avoid detection

A censoring authority who is capable of controlling how packets originating from its network are routed

Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing Around Decoys (CCS 2012)

Decoy routing
Routing attacks

How: Routing adversary must be able to

1) Locate decoy routers

   Telex: Public list of decoy router locations
   Ciripede: Scan Autonomous Systems (ASs)

   →Make a list of honest vs. tainted ASs

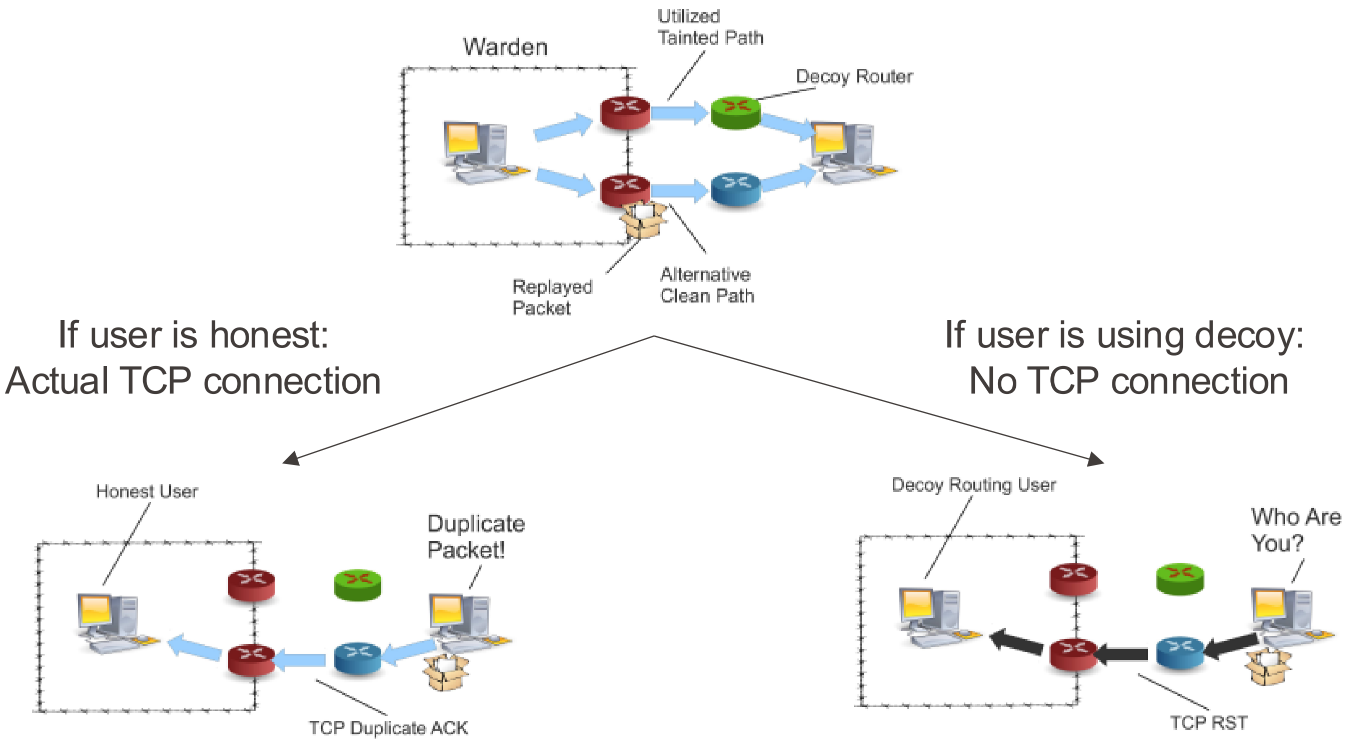
2) Select from a diverse set of paths in reaction to this knowledge

A censoring authority who is capable of controlling how packets originating from its network are routed

Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing Around Decoys (CCS 2012)

Routing attacks
Detection attacks

TCP Replay attack



If user is honest:
Actual TCP connection

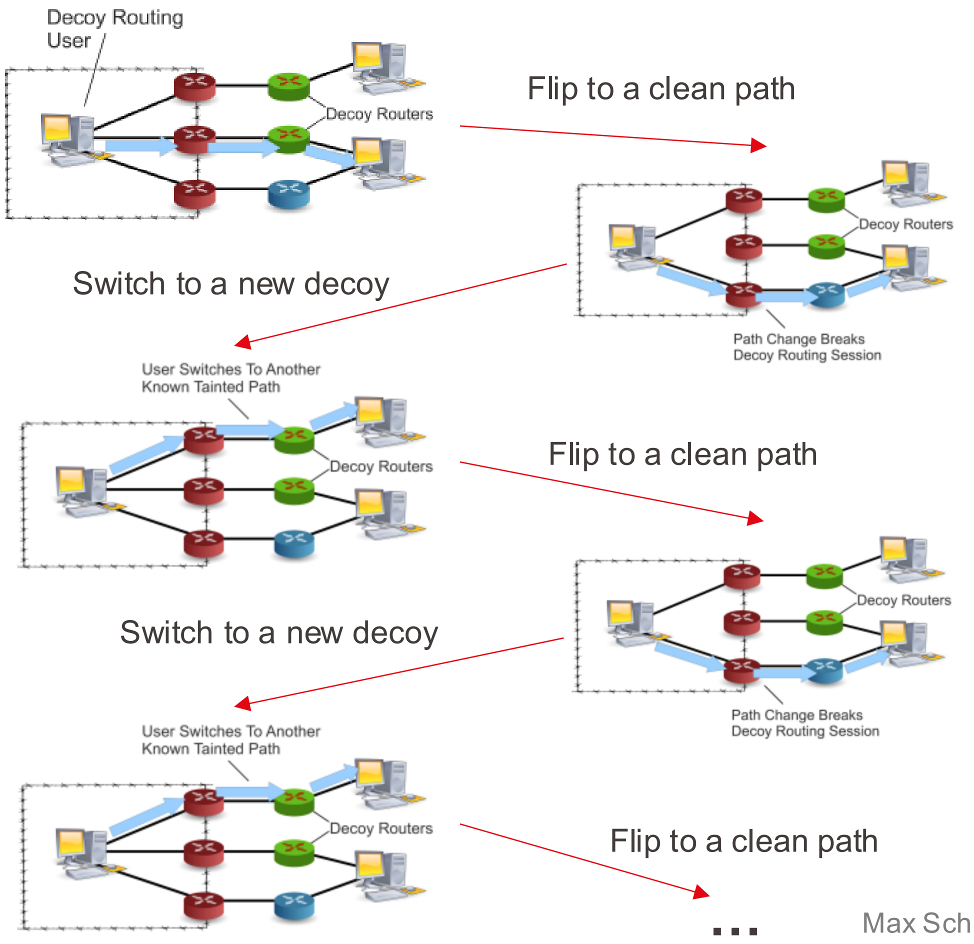
If user is using decoy:
No TCP connection

Goal: Identify users of decoy routing systems

- Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing Around Decoys (CCS 2012)

Routing attacks
Detection attacks

The "Crazy Ivan" attack

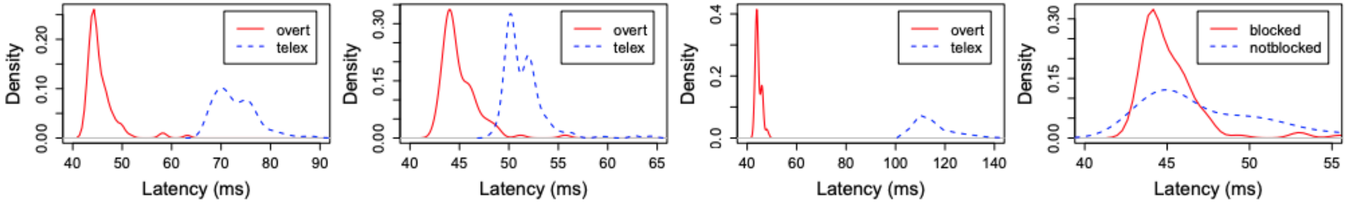


Goal: Identify users of decoy routing systems

Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing Around Decoys (CCS 2012)

Routing attacks

Timing attacks



(a) Amazon (b) Gmail (c) Facebook (d) blocked.telex.cc

Figure 5: Comparing distribution of latencies from notblocked.telex.cc to (a) Amazon (b) Gmail (c) Facebook and (d) blocked.telex.cc

How: Fingerprint network latency

Goal: Identify users of decoy routing systems

Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing Around Decoys (CCS 2012)

Decoy routing

Routing attacks

Through **routing attacks** a routing capable adversary can:

- Enumerate the participating decoy routers
- Successfully avoid sending traffic along routes containing these routers with little or no adverse effects
- Identify users of these schemes through active and passive attacks
- (In some cases) probabilistically identify connections to targeted destinations.

CRAZY EXPENSIVE!!

A censoring authority who is capable of controlling how packets originating from its network are routed

Amir Houmansadr, Edmund L. Wong, and Vitaly Shmatikov. No Direction Home: The True Cost of Routing Around Decoys (NDSS 2014)

Take aways

- Censorship resistance is key to freedom speech & information

- There is a strong connection between censorship resistance technology and anonymous communications

- To resist internet censorship requires:
  - Bootstrapping: find “helper” nodes
    - Lists, private retrieval, embedded in infrastructure
  - Hidden communication: avoid censor “during conversation”
    - Hide: network information, content, patterns
    - Comply with semantics ← do not imitate, be