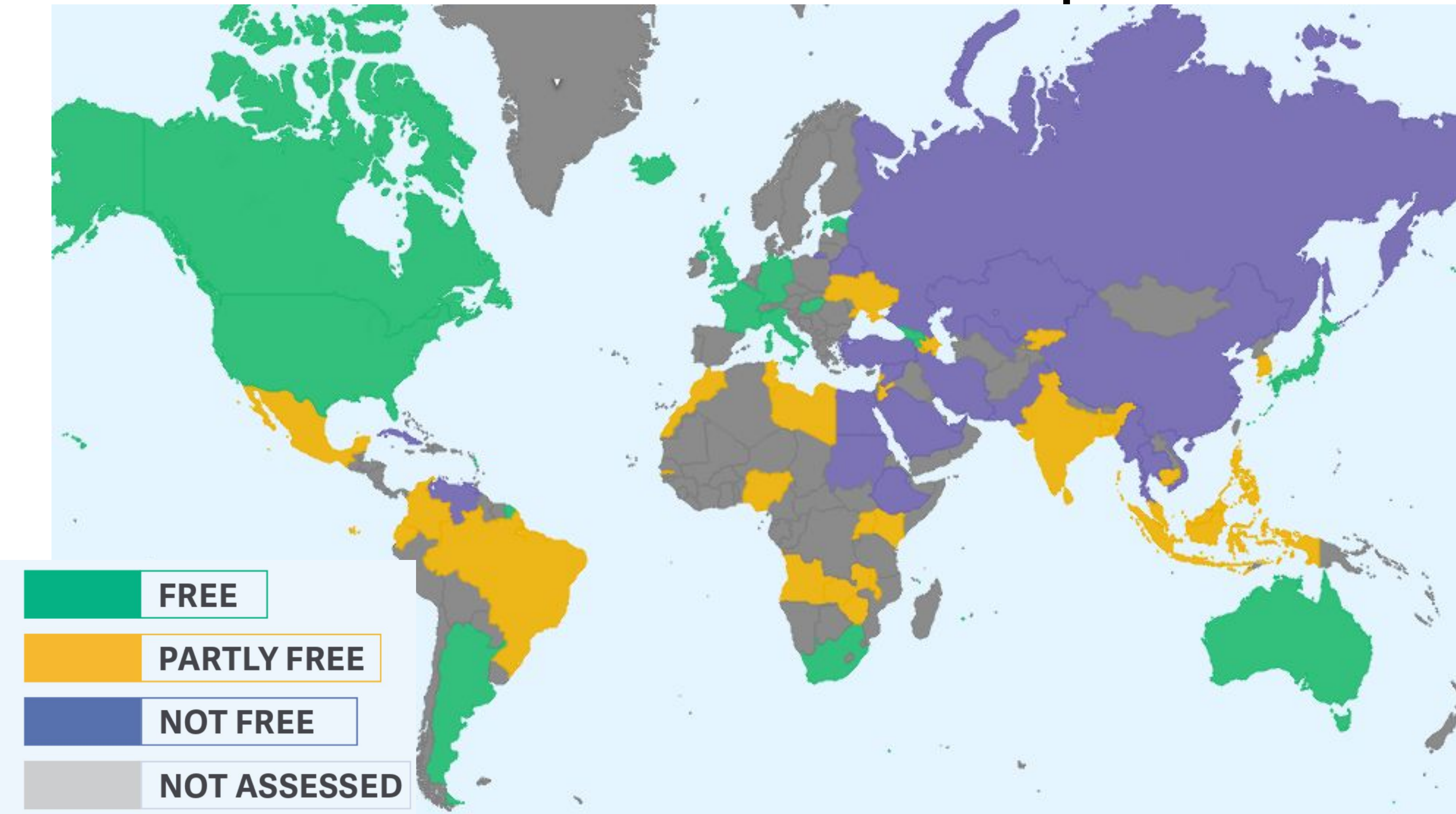
The use of TLS in Censorship Circumvention

Sergey Frolov, Eric Wustrow
University of Colorado Boulder

High-level overview

- Conducted **measurement** study of TLS connections
  - How do circumvention tools compare?
  - Some tools stick out ⇒ **easy to block**
- Developed a library to **blend in**

Freedom on the Net 2018 Map



Source: https://freedomhouse.org/report/freedom-net/freedom-net-2018/map

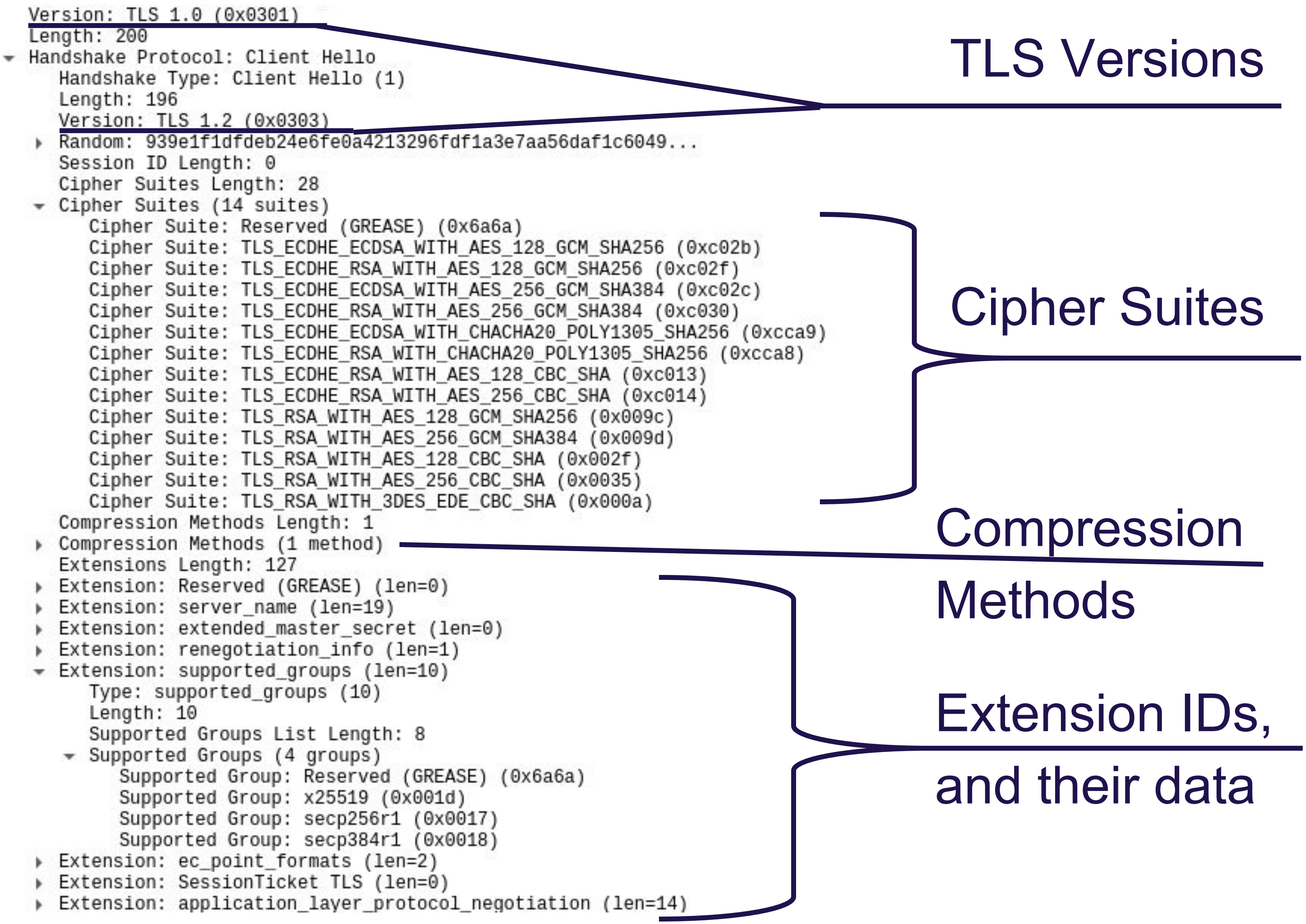
Use of TLS in circumventing tools

- Circumvention tools help to get around censorship, and many of them use TLS
- Censors are unlikely to block *all* of TLS

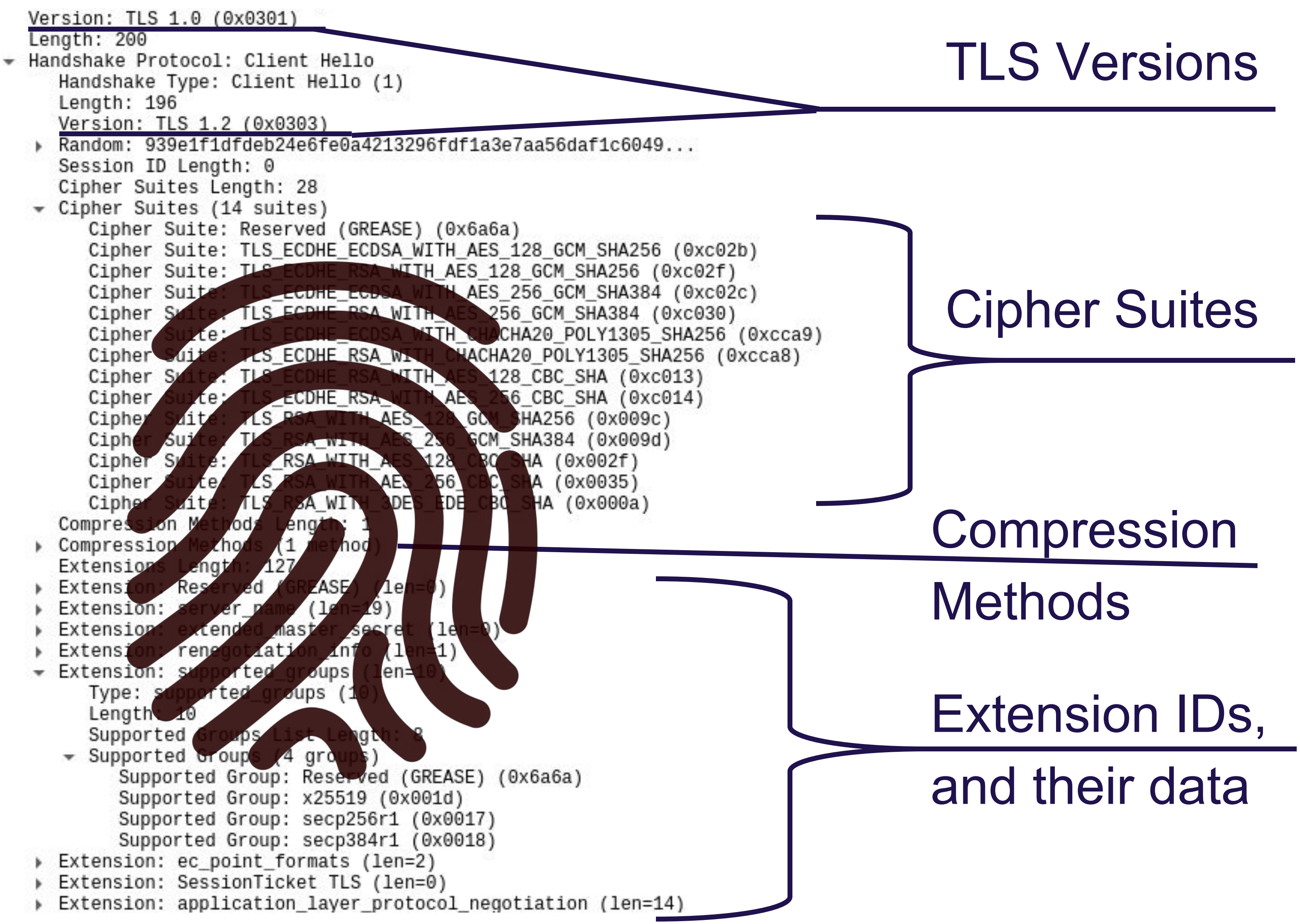
Lantern *TapDance* *Psiphon* *Signal*

Tor Outline

TLS ClientHello is sent in the clear



TLS ClientHello is sent in the clear



Differing TLS implementations

Compare	Chromium 71	Google Chrome 72
Seen	31278819 times (0.56%)	608850557 times (10.94%)
Rank	30	1
TLS Version	TLS 1.0	TLS 1.0
Handshake Version	TLS 1.2	TLS 1.2
Cipher Suites	GREASE (0x0a0a) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c) TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d) TLS_RSA_WITH_AES_128_CBC_SHA (0x002f) TLS_RSA_WITH_AES_256_CBC_SHA (0x0035) TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)	GREASE (0x0a0a) + TLS_AES_128_GCM_SHA256 (0x1301) + TLS_AES_256_GCM_SHA384 (0x1302) + TLS_CHACHA20_POLY1305_SHA256 (0x1303) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c) TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d) TLS_RSA_WITH_AES_128_CBC_SHA (0x002f) TLS_RSA_WITH_AES_256_CBC_SHA (0x0035) TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
Compression Methods	null (0x00)	null (0x00)
Extensions	GREASE (0x0a0a) server_name (0x0000) extended_master_secret (0x0017) renegotiation_info (0xff01) supported_groups (0x000a) ec_point_formats (0x000b) SessionTicket TLS (0x0023) application_layer_protocol_negotiation (0x0010) status_request (0x0005) signature_algorithms (0x000d) signed_certificate_timestamp (0x0012)	GREASE (0x0a0a) server_name (0x0000) extended_master_secret (0x0017) renegotiation_info (0xff01) supported_groups (0x000a) ec_point_formats (0x000b) SessionTicket TLS (0x0023) application_layer_protocol_negotiation (0x0010) status_request (0x0005) signature_algorithms (0x000d) signed_certificate_timestamp (0x0012) + key_share (0x0033) + psk_key_exchange_modes (0x002d) + supported_versions (0x002b)

The use of TLS in Censorship

- Censor’s Goal: block **only** circumvention tools
- Fingerprinting TLS connections minimizes **collateral damage**:
    - Less popular fingerprint ⇨ cheap to block

The use of TLS in Censorship

Network Traffic Obfuscation ›

Cyberoam firewall blocks meek by TLS signature

3 posts by 1 author

David Fifield 5/11/16

There was a post on the tor-talk mailing list saying that a recent DPI upgrade to Cyberoam firewalls gives them the ability to detect and block several of Tor's pluggable transports, including meek: https://lists.torproject.org/pipermail/tor-talk/2016-May/040898.html

The poster and I investigated and we think we know what they're doing. Recall that meek uses a web browser (Firefox 38) to camouflage its HTTPS requests. The Cyberoam devices are blocking TLS connections that have Firefox 38's TLS signature and SNI equal to one of our three front domains: www.google.com, a0.awsstatic.com, or ajax.aspnetcdn.com. Basically, they blocked meek, which looks like Firefox 38, by blocking Firefox 38, then reduced the damage by limiting the blocking to a small number of domains.

The blocking is deliberate, as the poster confirmed by talking to Cyberoam support. The connections get logged as a Tor Proxy attempt and

The use of TLS in Censorship

Network Traffic Obfuscation ›

Cyberoam firewall blocks meek by TLS signature

3 posts by 1 author

David Fifield 5/11/16

Network Traffic Obfuscation ›

FortiGuard firewall blocks meek by TLS signature

6 posts by 4 authors

David Fifield 7/24/16

Other recipients: kjsch...@yahoo.co.in

Here is a case similar to when Cyberoam blocked meek by TLS signature (https://groups.google.com/d/topic/traffic-obf/BpFSCVgi5rs). This time it's a FortiGuard firewall. Kanwaljeet Singh Channey ran some tests to help me figure out what was going on.

The story is basically the same as last time: the firewall looks for TLS that has the signature of a specific version of Firefox and is also

Challenge for circumvention tools

Tools have to:

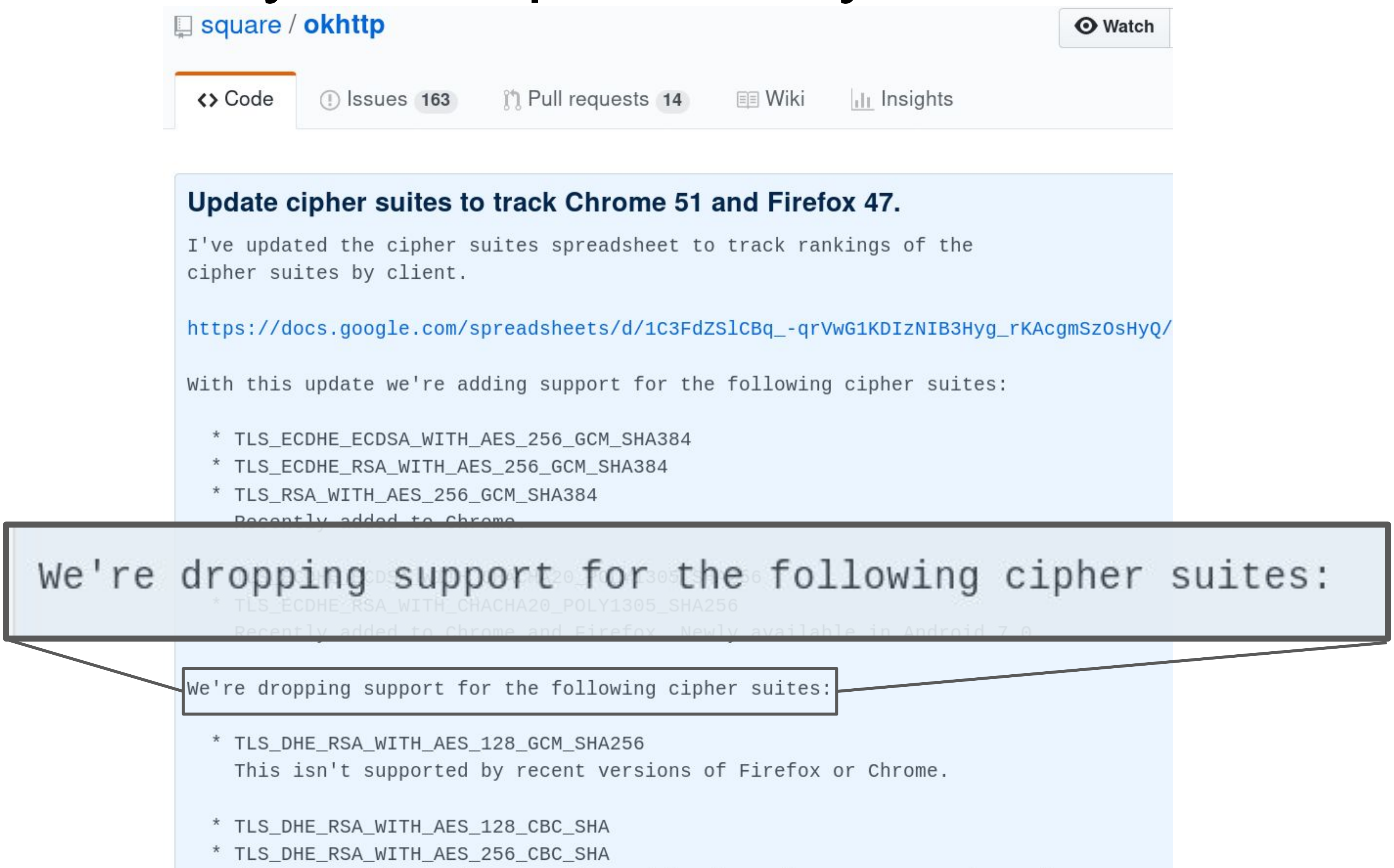
- Properly mimic/use a **popular** TLS implementation
- Track **changes** in popularity, and **update**

Mimicry is complicated

```
private static final ConnectionSpec GMAPS_CONNECTION_SPEC = new ConnectionSpec.Builder(ConnectionSpec.MODERN_TLS)
    .tlsVersions(TlsVersion.TLS_1_2)
    .cipherSuites(CipherSuite.TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,
                  CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
                  CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
                  CipherSuite.TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
                  CipherSuite.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
                  CipherSuite.TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
                  CipherSuite.TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,
                  CipherSuite.TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,
                  CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
                  CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
                  CipherSuite.TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
                  CipherSuite.TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
                  CipherSuite.TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
                  CipherSuite.TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
                  CipherSuite.TLS_RSA_WITH_AES_128_GCM_SHA256,
                  CipherSuite.TLS_RSA_WITH_AES_256_GCM_SHA384,
                  CipherSuite.TLS_RSA_WITH_AES_128_CBC_SHA,
                  CipherSuite.TLS_RSA_WITH_AES_256_CBC_SHA)
    .supportsTlsExtensions(true)
```

Attempts to look like Google Maps

Mimicry is complicated by libraries



square / okhttp

Watch

Code | Issues 163 | Pull requests 14 | Wiki | Insights

Update cipher suites to track Chrome 51 and Firefox 47.

I've updated the cipher suites spreadsheet to track rankings of the
cipher suites by client.

https://docs.google.com/spreadsheets/d/1C3FdZSlCBq_-qrVwG1KDIzNIB3Hyg_rKAcgmSzOsHyQ/

With this update we're adding support for the following cipher suites:

* TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
* TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* TLS_RSA_WITH_AES_256_GCM_SHA384

We're dropping support for the following cipher suites:

We're dropping support for the following cipher suites:

* TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
  This isn't supported by recent versions of Firefox or Chrome.

* TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* TLS_DHE_RSA_WITH_AES_256_CBC_SHA

Mimicry is complicated by libraries

```
private static final ConnectionSpec GMAPS_CONNECTION_SPEC = new ConnectionSpec.Builder(ConnectionSpec.MODERN_TLS)
    .tlsVersions(TlsVersion.TLS_1_2)
    .cipherSuites(CipherSuite.TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,
                  CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
                  CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
                  CipherSuite.TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
                  CipherSuite.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
                  CipherSuite.TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
                  CipherSuite.TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,
                  CipherSuite.TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,
                  CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
                  CipherSuite.TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
                  CipherSuite.TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
                  CipherSuite.TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
                  CipherSuite.TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
                  CipherSuite.TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
                  CipherSuite.TLS_RSA_WITH_AES_128_GCM_SHA256,
                  CipherSuite.TLS_RSA_WITH_AES_256_GCM_SHA384,
                  CipherSuite.TLS_RSA_WITH_AES_128_CBC_SHA,
                  CipherSuite.TLS_RSA_WITH_AES_256_CBC_SHA)
    .supportsTlsExtensions(true)
```

Does not look like Google Maps. Looks unique

TLS Fingerprinting among friends

How well do circumvention tools blend in with general traffic?

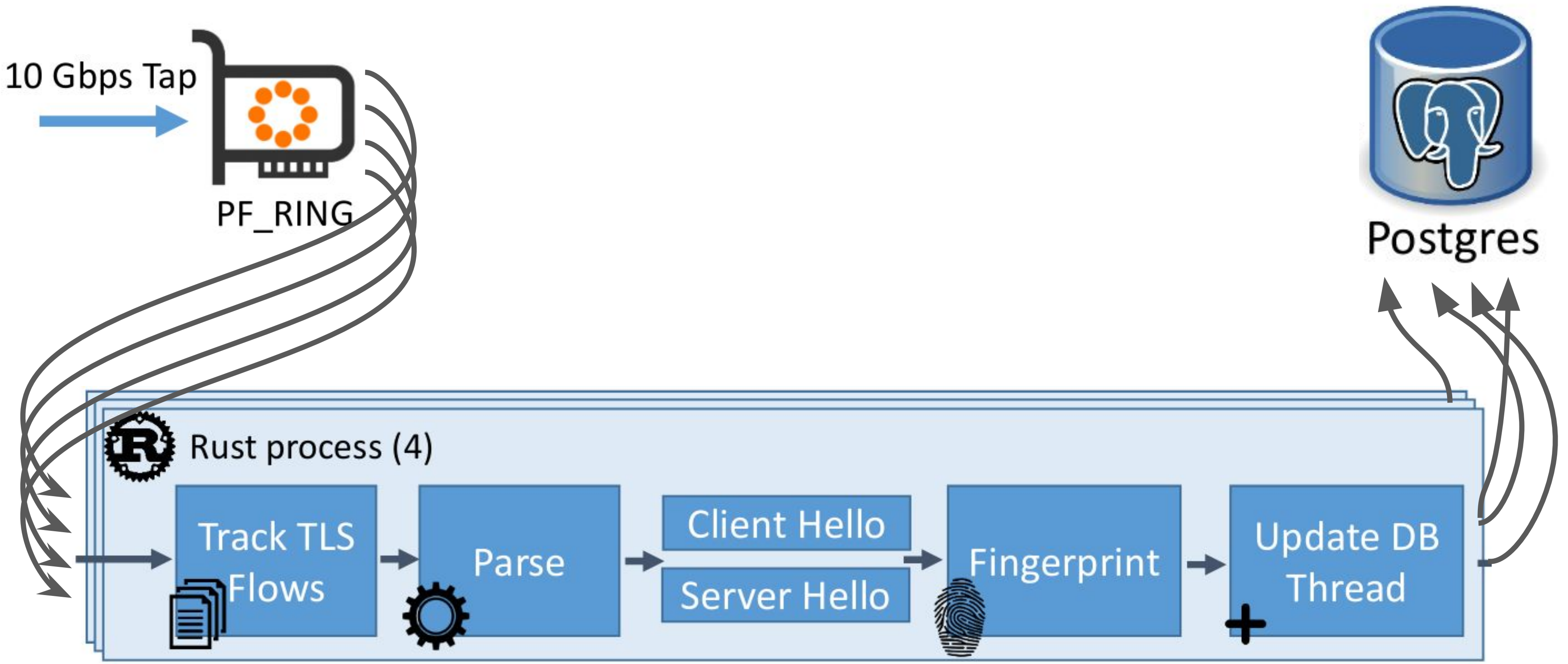
- What does "general traffic" look like?

Measurement Study

- Collected anonymized TLS “fingerprints” on the university campus
  - Got institutional review board exemption
- Compare fingerprints to circumvention tools’
  - Data not representative, but shows overall trend

University of Colorado
Boulder

Measurement Study



Measurement Study Results

Between October 2017 and August 2018:

- Observed over 11 billion TLS ClientHello
- Extracted 230,000 unique fingerprints

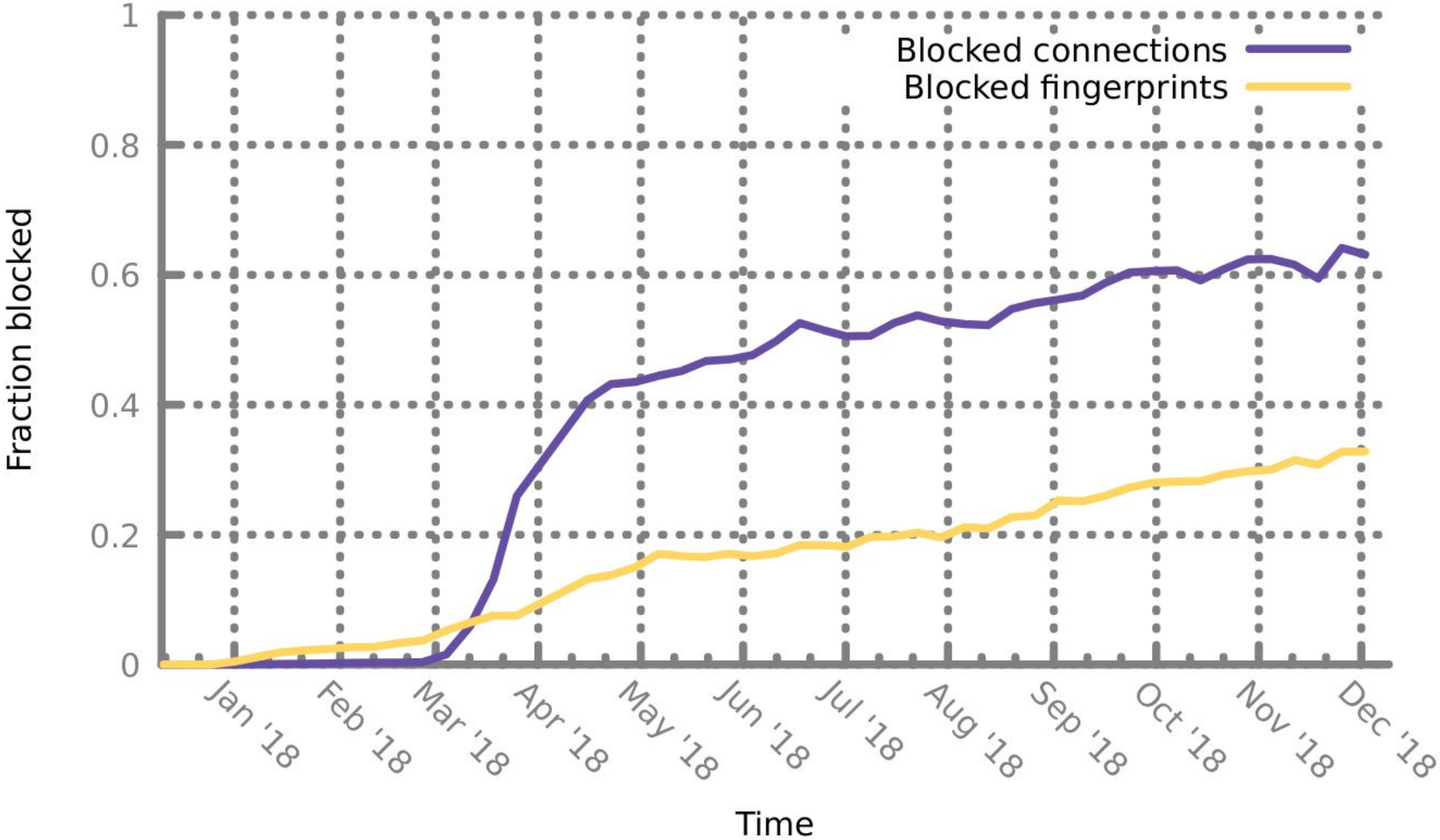
February 2019:

- Data collection ongoing
- Updated parser to support TLS 1.3

Tool	Fingerprints' Popularity	Note
Psiphon	8.76% 2.42% 0.25% 0.04% 0.04% 0.01% 0.0002% 0.0000% 0.0000%	Generates multiple fingerprints 👍 One successfully mimics latest Chrome 👍 Several fingerprints are unique 👎
Outline	8.76%	Uses Electron (Chrome) 👍
meek	0.50%	Uses Firefox ESR
Snowflake	0.0008%	Still under initial development
Lantern	0.0003%	≈Golang but -SNI, +SessionTicket
TapDance	random	Generated random fingerprints
Signal	0.0000% 0.0000%	No longer circumvents :(

TLS Fingerprintability of various tools as of early May 2018

Whitelist feasibility



https://tlsfingerprint.io

Available data:

- Top fingerprints, clusters of fingerprints.
- Usage statistics for cipher suites, extensions, curves, supported versions, signature algorithms, ALPNs, etc.
- Upload a tool’s pcap and check its fingerprint

TLSFingerprint.io Home Top Stats Cipher Suites Weak Ciphers Extensions Signature Algorithms ALPN Session Tickets pcap

TLS Fingerprint

We collect anonymized TLS Client Hello messages from the University of Colorado Boulder campus network, in order to measure the popularity of various implementations actually used in practice.

Your browser generates the fingerprint bbf04e5f1881f506 (from Cluster #13), which is seen in **15.61%** of connections, making it ranked **#1** by popularity.

Why collect fingerprints?

TLS fingerprints allow us to identify problems in common TLS applications. For example, we can investigate the ability of censorship circumvention tools to mimic and blend in with other popular TLS implementations. Censors can block circumvention tools that send uncommon TLS fingerprints. To combat this, we can observe the fingerprints that circumvention tools produce and compare them to our dataset. Less popular fingerprints are at risk of being blocked by censors at low cost.

We can also measure Weak cipher suites still in widespread use, and track the popularity of TLS Versions and Extensions.

uTLS Library

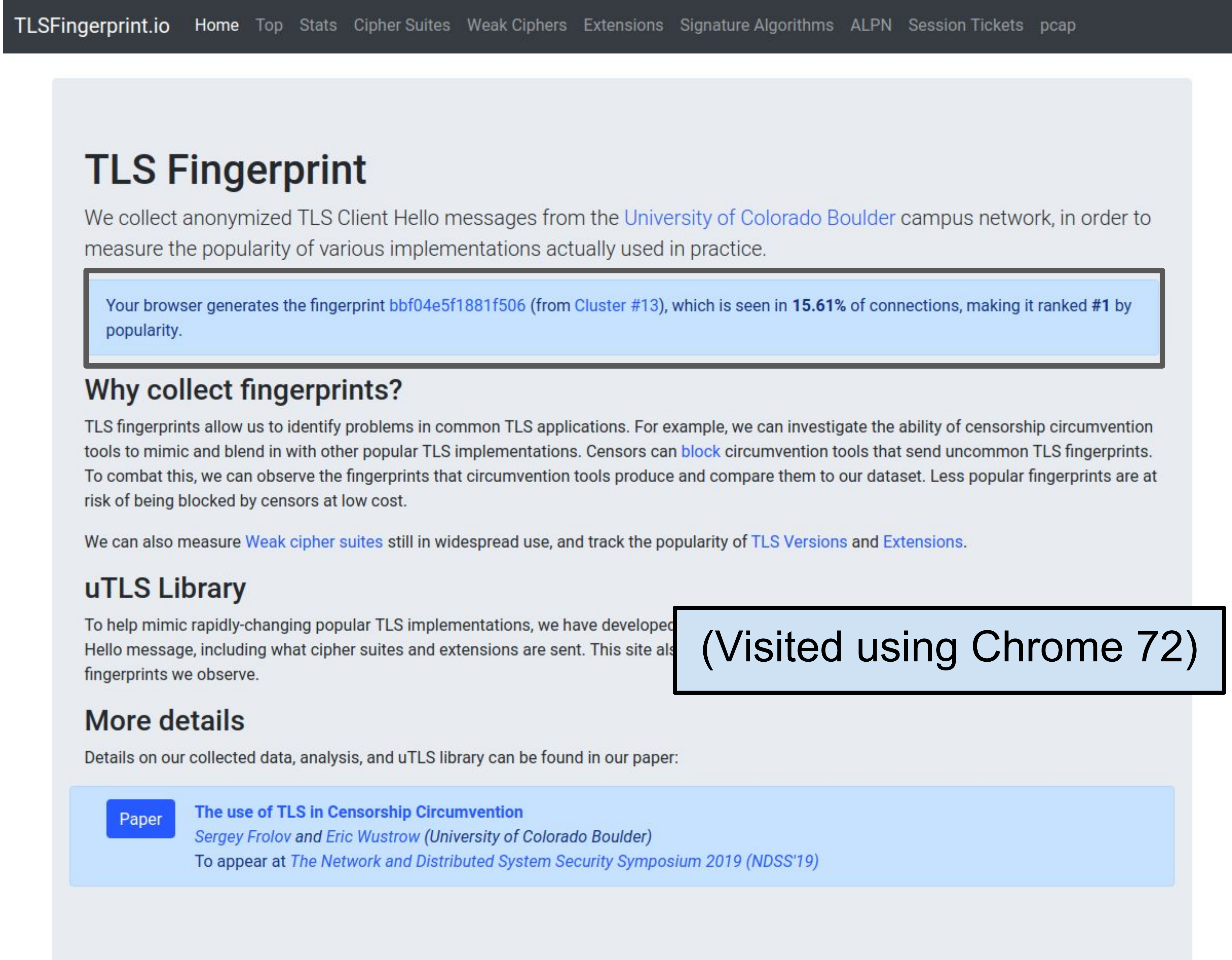
To help mimic rapidly-changing popular TLS implementations, we have develope
Hello message, including what cipher suites and extensions are sent. This site als
fingerprints we observe.

(Visited using Chrome 72)

More details

Details on our collected data, analysis, and uTLS library can be found in our paper:

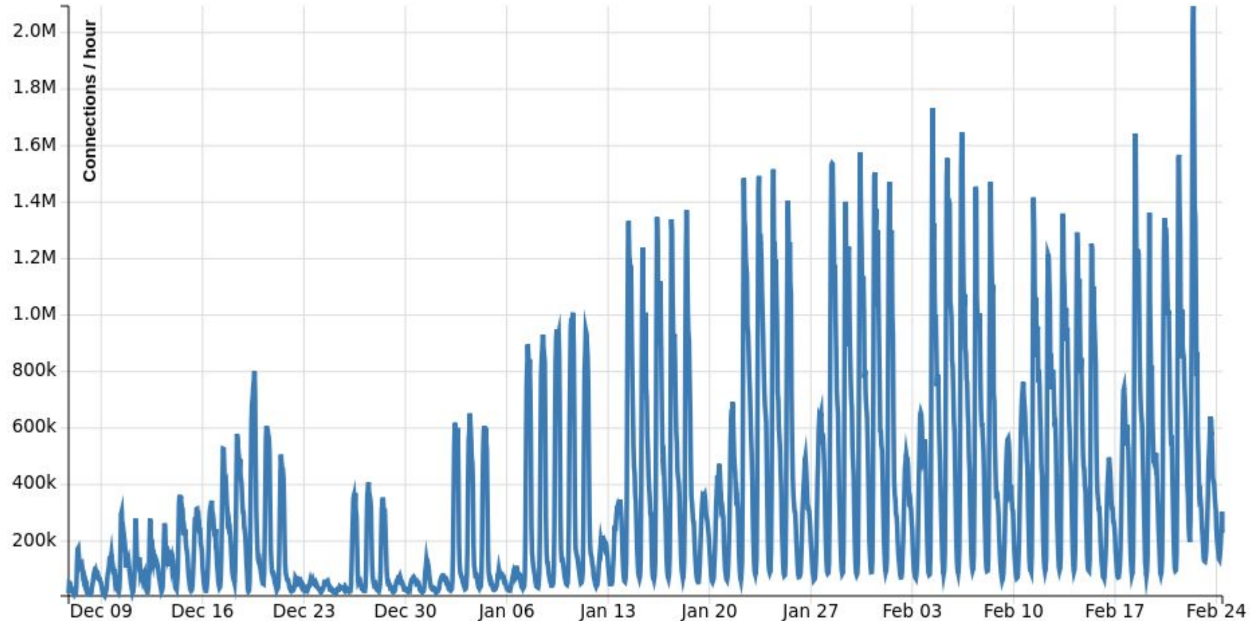
Paper **The use of TLS in Censorship Circumvention**
Sergey Frolov and *Eric Wustrow* *(University of Colorado Boulder)*
To appear at *The Network and Distributed System Security Symposium 2019 (NDSS'19)*


TLSFingerprint.io
Home
Top
Stats
Cipher Suites
Weak Ciphers
Extensions
Signature Algorithms
ALPN
Session Tickets
pcap
TLS Fingerprint
We collect anonymized TLS Client Hello messages from the University of Colorado Boulder campus network, in order to measure the popularity of various implementations actually used in practice.
Your browser generates the fingerprint bbf04e5f1881f506 (from Cluster #13), which is seen in 15.61% of connections, making it ranked #1 by popularity.
Why collect fingerprints?
TLS fingerprints allow us to identify problems in common TLS applications. For example, we can investigate the ability of censorship circumvention tools to mimic and blend in with other popular TLS implementations. Censors can block circumvention tools that send uncommon TLS fingerprints. To combat this, we can observe the fingerprints that circumvention tools produce and compare them to our dataset. Less popular fingerprints are at risk of being blocked by censors at low cost.
We can also measure Weak cipher suites still in widespread use, and track the popularity of TLS Versions and Extensions.
uTLS Library
To help mimic rapidly-changing popular TLS implementations, we have developed
Hello message, including what cipher suites and extensions are sent. This site als
fingerprints we observe.
More details
Details on our collected data, analysis, and uTLS library can be found in our paper:
Paper
The use of TLS in Censorship Circumvention
Sergey Frolov and Eric Wustrow (University of Colorado Boulder)
To appear at The Network and Distributed System Security Symposium 2019 (NDSS'19)
(Visited using Chrome 72)

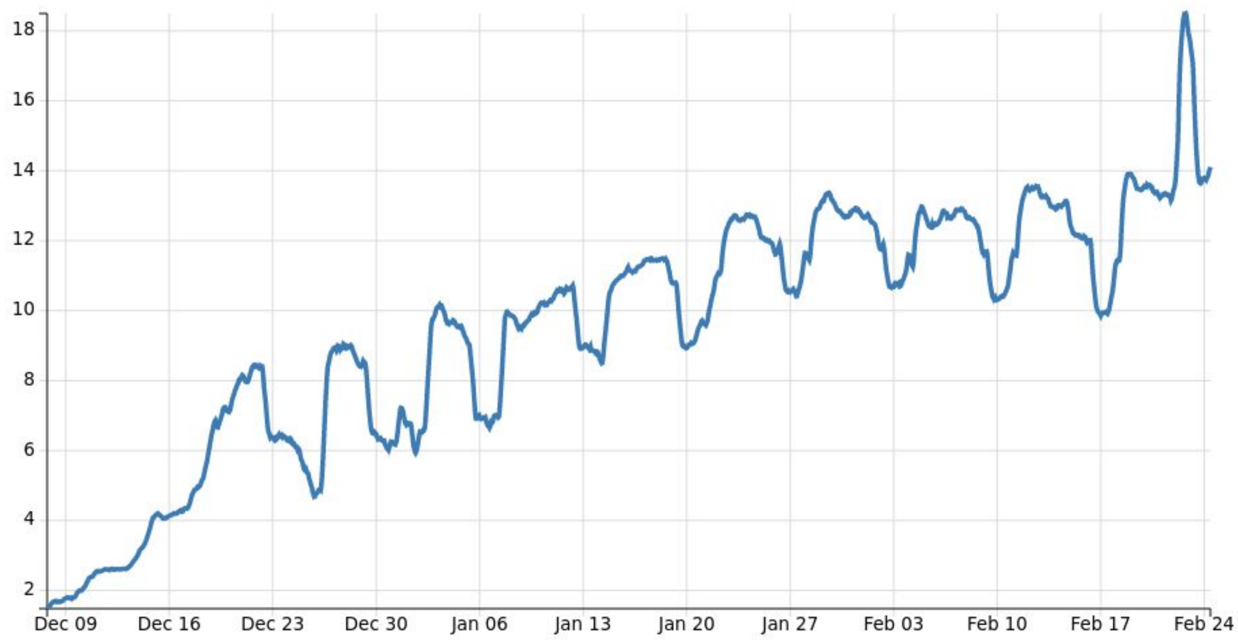
bbf04e5f1881f506

Seen	(all time) (past week)	693M times (11.37%) 99M times (15.61%)
Rank	(all time) (past week)	1 / 119884 1 / 12701
TLS Version		TLS 1.0
Handshake Version		TLS 1.2
Cipher Suites exact match		GREASE (0x0a0a) TLS_AES_128_GCM_SHA256 (0x1301) TLS_AES_256_GCM_SHA384 (0x1302) TLS_CHACHA20_POLY1305_SHA256 (0x1303) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c) TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d) TLS_RSA_WITH_AES_128_CBC_SHA (0x002f) TLS_RSA_WITH_AES_256_CBC_SHA (0x0035) TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
Compression Methods		null (0x00)
Extensions exact match		GREASE (0x0a0a) server_name (0x0000) extended_master_secret (0x0017) renegotiation_info (0xff01) supported_groups (0x000a) ec_point_formats (0x000b) SessionTicket TLS (0x0023)

Times seen (per hour)



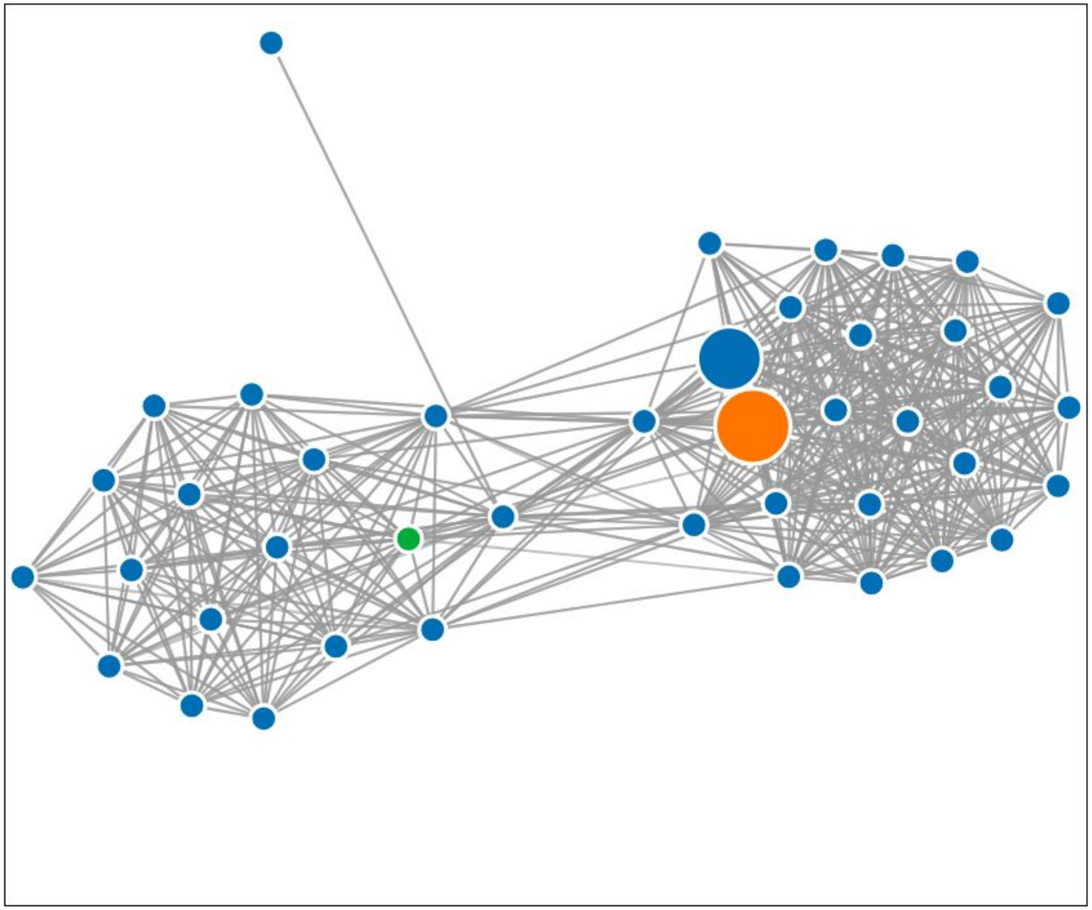
Percent seen (24 hour averaged)



Clustering Fingerprints

Cluster #7

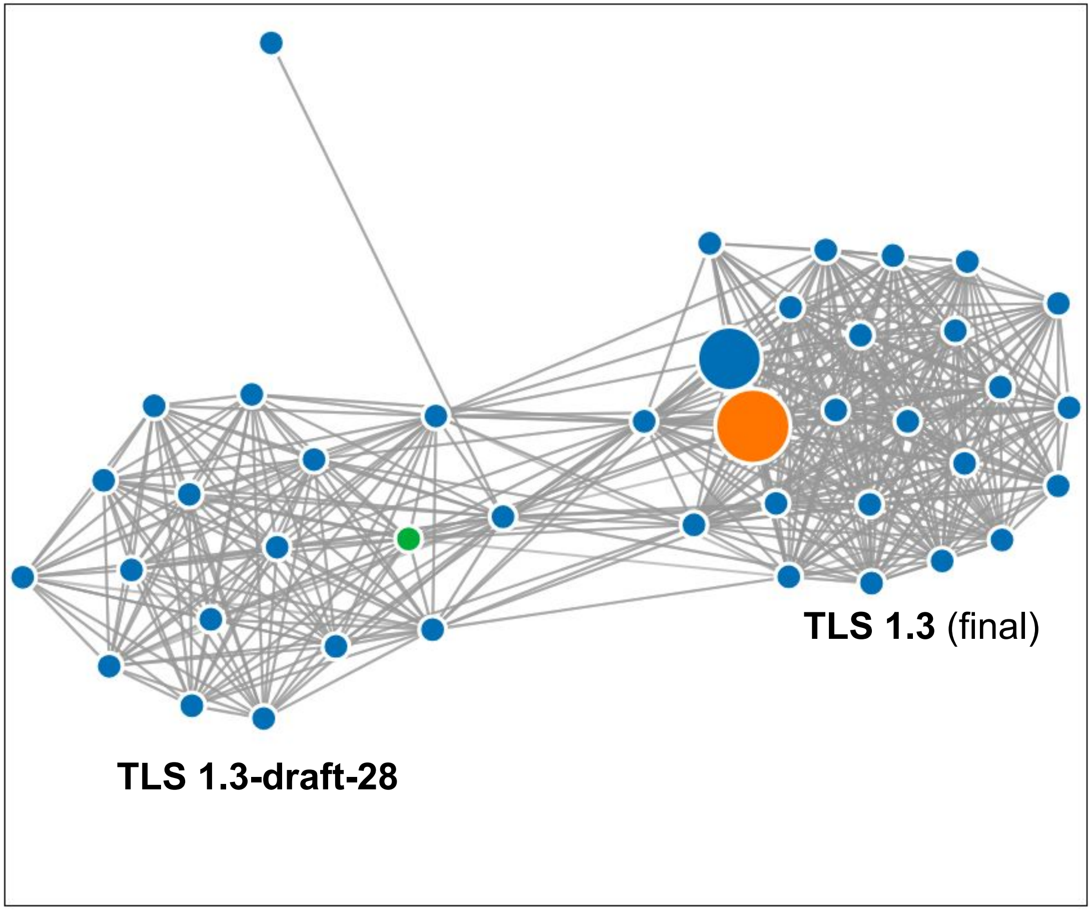
Fingerprints included 42
Connections 3.6%
Highlighted 6bfedc5d5c740d58
Comparing 4855b34a61fc0e2f



Clustering Fingerprints

Cluster #7

Fingerprints included 42
Connections 3.6%
Highlighted 6bfedc5d5c740d58
Comparing 4855b34a61fc0e2f



	Fingerprint 6bfedc5d5c740d58	Fingerprint 4855b34a61fc0e2f
EC Point Formats	uncompressed (0x00)	uncompressed (0x00)
ALPN	h2 http/1.1	h2 http/1.1
Key Share	x25519 (0x001d) secp256r1 (0x0017)	x25519 (0x001d) secp256r1 (0x0017)
PSK Key Exchange Modes	psk_dhe_ke (0x01)	psk_dhe_ke (0x01)
Supported Versions	- TLS 1.3 (0x0304) TLS 1.2 (0x0303) TLS 1.1 (0x0302) TLS 1.0 (0x0301)	+ TLS 1.3-draft-28 (0x7f1c) TLS 1.2 (0x0303) TLS 1.1 (0x0302) TLS 1.0 (0x0301)
Certificate Compression Algs		
Record Size Limit	16385	16385

Top Fingerprints

Rank	Fingerprint	Connections
1	bbf04e5f1881f506	14.0%
2	ec55e5b4136c7949	8.5%
3	bc3118430dde084a	6.2%
4	340fd4ec4e6b6c49	4.9%
5	19d534641d9ebeb7	3.7%

Top Extensions

Rank	Extension	Connections
1	supported_groups (0x000a)	99.3%
2	server_name (0x0000)	98.9%
3	signature_algorithms (0x000d)	98.4%
4	ec_point_formats (0x000b)	96.4%
5	extended_master_secret (0x0017)	85.8%

Top Named Groups

Rank	Named Group	Connections
1	secp256r1 (0x0017)	99.3%
2	secp384r1 (0x0018)	95.9%
3	x25519 (0x001d)	81.7%
4	sect239k1 (0x0008)	60.7%
5	secp521r1 (0x0019)	47.9%

Top `supported_versions`

Rank	Version	Connections
1	TLS 1.3 (0x0304)	30.7%
2	TLS 1.2 (0x0303)	28.8%
3	TLS 1.1 (0x0302)	28.7%
4	TLS 1.0 (0x0301)	28.7%
5	GREASE (0x0a0a)	24.0%

Top Cipher Suites

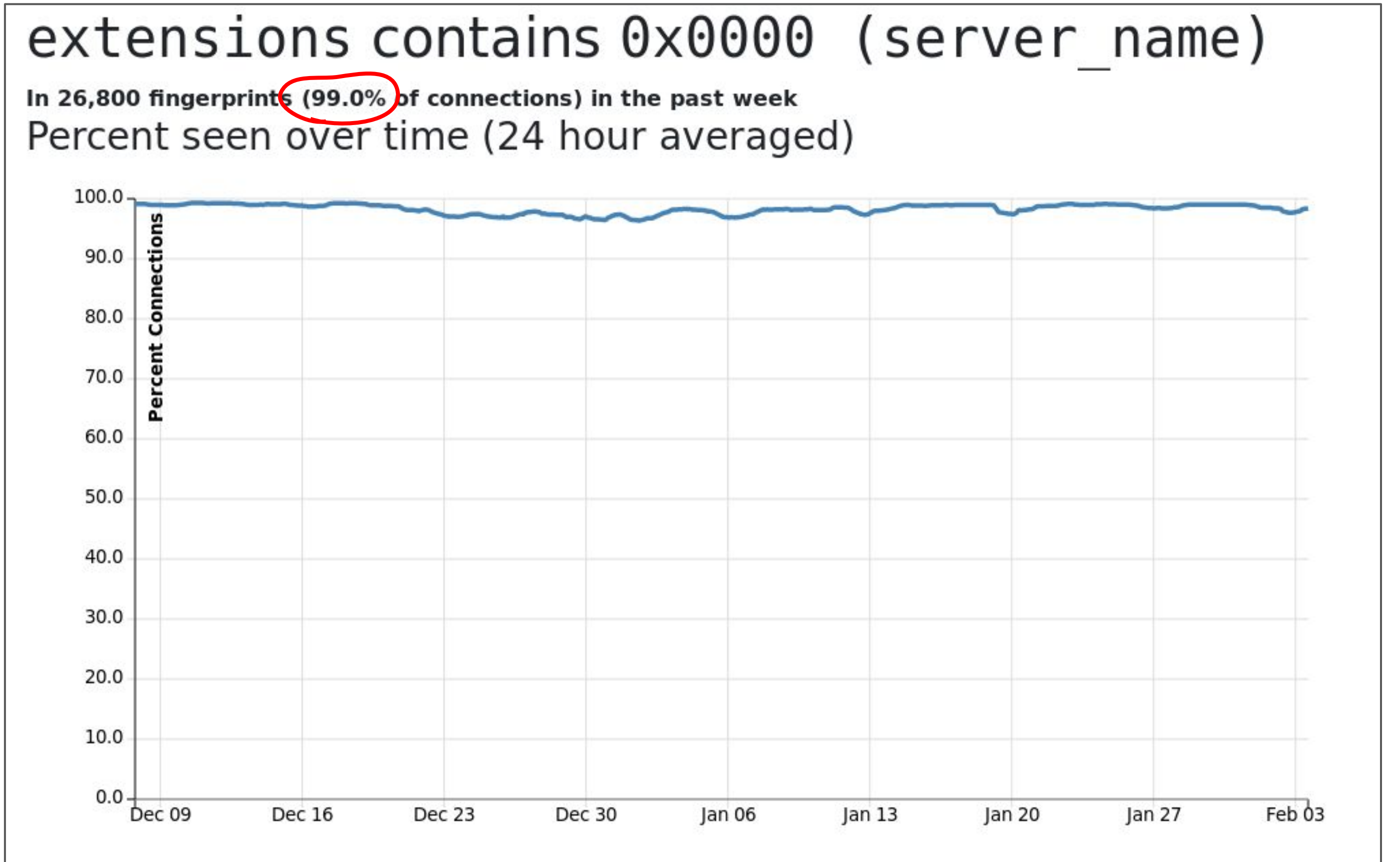
Rank	Cipher Suite	%Client Hellos	%Server Hellos
1	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	96.1%	1.0%
2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	95.9%	1.0%
3	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	93.4%	14.8%
4	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	92.9%	49.0%
5	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	92.8%	1.7%

Top Clusters

Cluster #	Fingerprints	Connections
#5	52	24.4%
#13	20	21.1%
#3	66	11.2%
#1	151	7.3%
#10	31	4.4%

Extension popularity: SNI

- Server Name Indication (SNI) specifies host name
- Some circumvention tools **do not** send SNI
  - **Uncommon** in our dataset (< 1% of connections)



Protecting circumvention tools

How do we **protect** tools from TLS fingerprinting?

- Bundle in latest Chrome? (Updates?)
- Use other (non-TLS) protocols?
  - Unpopular protocols can be blocked
- **Purpose-built TLS library for mimicry**

uTLS

We built a library to assist circumvention tool developers:

- **Mimic** popular fingerprints
- Generate **randomized** fingerprints to defeat blacklists

uTLS

uTLS is a fork of Golang's `crypto/tls`

- Designed to be *an addition* to `crypto/tls`
  - Enables **easy merges**
- Most crypto handled by `crypto/tls` code

uTLS

Other useful features for anti-censorship

- Fake Session Tickets
- Low-level access to handshake details
- Forge TLS connections with master secret / parameters exchanged out-of-band

uTLS + tlsfingerprint.io = code generation

uTLS
generated
code

Click to expand

```
// import tls "github.com/refraction-networking/utls"
tcpConn, err := net.Dial("tcp", "tlsfingerprint.io:443")
if err != nil {
	fmt.Printf("net.Dial() failed: %+v\n", err)
	return
}

config := tls.Config{ServerName: "tlsfingerprint.io"}
// unsupported features are highlighted with dark red
// only use this fingerprint to talk to servers that also do not support those features
tlsConn := tls.Client(tcpConn, &tlsConfig, utls.HelloCustom)
clientHelloSpec := tls.ClientHelloSpec {
	CipherSuites: []uint16{
		tls.GREASE_PLACEHOLDER,
		tls.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
		tls.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
		tls.TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
		tls.TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
		tls.TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305,
		tls.TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305,
		tls.TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
		tls.TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
		tls.TLS_RSA_WITH_AES_128_GCM_SHA256,
		tls.TLS_RSA_WITH_AES_256_GCM_SHA384,
		tls.TLS_RSA_WITH_AES_128_CBC_SHA,
		tls.TLS_RSA_WITH_AES_256_CBC_SHA,
		tls.TLS_RSA_WITH_3DES_EDE_CBC_SHA,
	},
	CompressionMethods: []byte{
		0x00, // compressionNone
	},
	Extensions: []tls.TLSExtension{
		&tls.UtlsGREASEExtension{},
		&tls.SNIExtension{},
```

Screenshot with auto-generated code to **mimic** Chromium 71 was taken on https://tlsfingerprint.io/id/e741ba143f8bd568

uTLS

https://github.com/refraction-networking/utls

uTLS is (being) integrated by numerous tools, including

- Psiphon
- Lantern
- meek_lite
- TapDance

Summary

- Collected tens of billions of TLS fingerprints
- Analyzed censorship circumvention tools and determined ones at risk
- Developed uTLS library to assist circumvention tool developers
- Ongoing study; open access to data

FIN

Questions?

Links:

https://tlsfingerprint.io

https://github.com/refraction-networking/utls

Acknowledgments:

Thanks to Ben Schwartz, David Fifield, Rod Hynes, Ox Cart, Dan Jones, Conan Moore, and J. Alex Halderman.

backup slides

Popular fingerprints change rapidly

August 2018

Rank	Client	% Connections
1	Chrome 65-68	16.51%
2	iOS 11/macOS 10.13 Safari	5.95%
3	MS Office 2016 (including Outlook)	5.34%
4	Chrome 65-68 (with padding)	4.62%
5	Edge 15-18, IE 11	4.05%
6	Firefox 59-61 (with padding)	3.62%
7	Safari 11.1 on Mac OS X	2.82%
8	iOS 10/macOS 10.12 Safari	2.49%
9	iOS 11/macOS 10.13 Safari (with padding)	2.42%
10	Firefox 59-61	2.22%

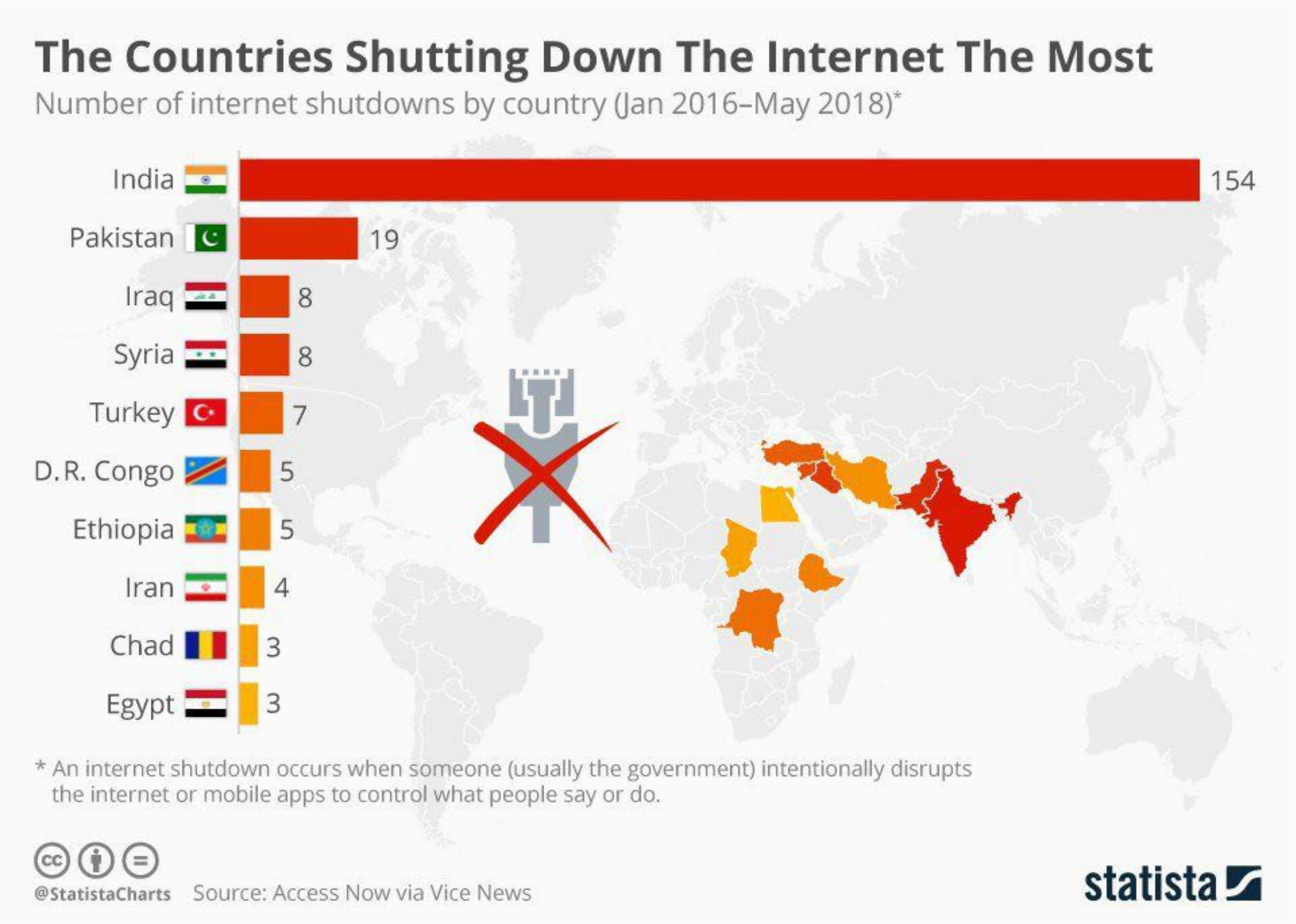
December 2018

	Rank	Client	% Connections
new	1	Chrome 70 (with padding)	8.49%
new	2	iOS 12/macOS 10.14 Safari	7.55%
new	3	iOS 12/macOS 10.14 Safari (without ALPN)	4.15%
new	4	Chrome 70	4.10%
new	5	iOS 12/macOS 10.14 Safari (with padding)	4.09%
	6	Edge 15-18, IE 11	3.27%
	7	MS Office 2016 (including Outlook)	3.01%
	8	iOS 10/macOS 10.12 Safari	2.72%
	9	iOS 11/macOS 10.13 Safari	2.68%
new	10	Chrome 71 (with padding)	2.48%

TLSFingerprint.io Home Top fingerprints Total Weak Ciphers Selected Ciphers Browsers pcap

Compare	ada0dd7bc92244b1	280d49bab731a7b1
Seen	312591691 times (7.09%)	2076979 times (0.05%)
Rank	3	157
TLS Version	TLS 1.0	TLS 1.0
Handshake Version	TLS 1.2	TLS 1.2
Cipher Suites	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009) TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d) TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c) TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d) TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c) TLS_RSA_WITH_AES_256_CBC_SHA (0x0035) TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009) TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) + TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc008) + TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012) TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d) TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c) TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d) TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c) TLS_RSA_WITH_AES_256_CBC_SHA (0x0035) TLS_RSA_WITH_AES_128_CBC_SHA (0x002f) + TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
Compression Methods	null (0x00)	null (0x00)
Extensions	renegotiation_info (0xff01) server_name (0x0000) extended_master_secret (0x0017) signature_algorithms (0x000d) status_request (0x0005) - NPN (0x3374) signed_certificate_timestamp (0x0012) - application_layer_protocol_negotiation (0x0010) ec_point_formats (0x000b) supported_groups (0x000a)	renegotiation_info (0xff01) server_name (0x0000) extended_master_secret (0x0017) signature_algorithms (0x000d) status_request (0x0005) signed_certificate_timestamp (0x0012) ec_point_formats (0x000b) supported_groups (0x000a)

Internet shutdowns are common



TLS Fingerprint

We collect anonymized TLS Client Hello messages from the University of Colorado Boulder campus network, in order to measure the popularity of various implementations actually used in practice.

Your browser generates the fingerprint e708f363a76a041d (from Cluster #7), which is seen in **0.01%** of connections, making it ranked **#505** by popularity.

Why collect fingerprints?

TLS fingerprints allow us to identify problems in common TLS applications. For example, we can investigate the ability of censorship circumvention tools to mimic and blend in with other popular TLS implementations. Censors can block circumvention tools that send uncommon TLS fingerprints. To combat this, we can observe the fingerprints that circumvention tools produce and compare them to our dataset. Less popular fingerprints are at risk of being blocked by censors at low cost.

We can also measure Weak cipher suites still in widespread use, and track the popularity of TLS Versions and Extensions.

uTLS Library

To help mimic rapidly-changing popular TLS implementations, we have developed the uTLS library, that provides fine-grained control over the Client Hello message, including what cipher suites and extensions are sent. This site also provides auto-generated uTLS configuration code to mimic fingerprints we observe.

More details

Details on our collected data, analysis, and uTLS library can be found in our paper:

Paper **The use of TLS in Censorship Circumvention**
Sergey Frolov and *Eric Wustrow* *(University of Colorado Boulder)*
To appear at *The Network and Distributed System Security Symposium 2019 (NDSS'19)*

(Visited using Firefox 65 on Linux Fedora 29)

TLSFingerprint.io Home Top Stats Cipher Suites Weak Ciphers Extensions Signature Algorithms ALPN Session Tickets pcap

TLS Fingerprint

We collect anonymized TLS Client Hello messages from the University of Colorado Boulder campus network, in order to measure the popularity of various implementations actually used in practice.

Your browser generates the fingerprint e708f363a76a041d (from Cluster #7), which is seen in **0.01%** of connections, making it ranked **#505** by popularity.

Why collect fingerprints?

TLS fingerprints allow us to identify problems in common TLS applications. For example, we can investigate the ability of censorship circumvention tools to mimic and blend in with other popular TLS implementations. Censors can block circumvention tools that send uncommon TLS fingerprints. To combat this, we can observe the fingerprints that circumvention tools produce and compare them to our dataset. Less popular fingerprints are at risk of being blocked by censors at low cost.

We can also measure Weak cipher suites still in widespread use, and track the popularity of TLS Versions and Extensions.

uTLS Library

To help mimic rapidly-changing popular TLS implementations, we have developed the uTLS library, that provides fine-grained control over the Client Hello message, including what cipher suites and extensions are sent. This site also provides auto-generated uTLS configuration code to mimic fingerprints we observe.

More details

Details on our collected data, analysis, and uTLS library can be found in our paper:

Paper **The use of TLS in Censorship Circumvention**
Sergey Frolov and *Eric Wustrow* *(University of Colorado Boulder)*
To appear at *The Network and Distributed System Security Symposium 2019 (NDSS'19)*

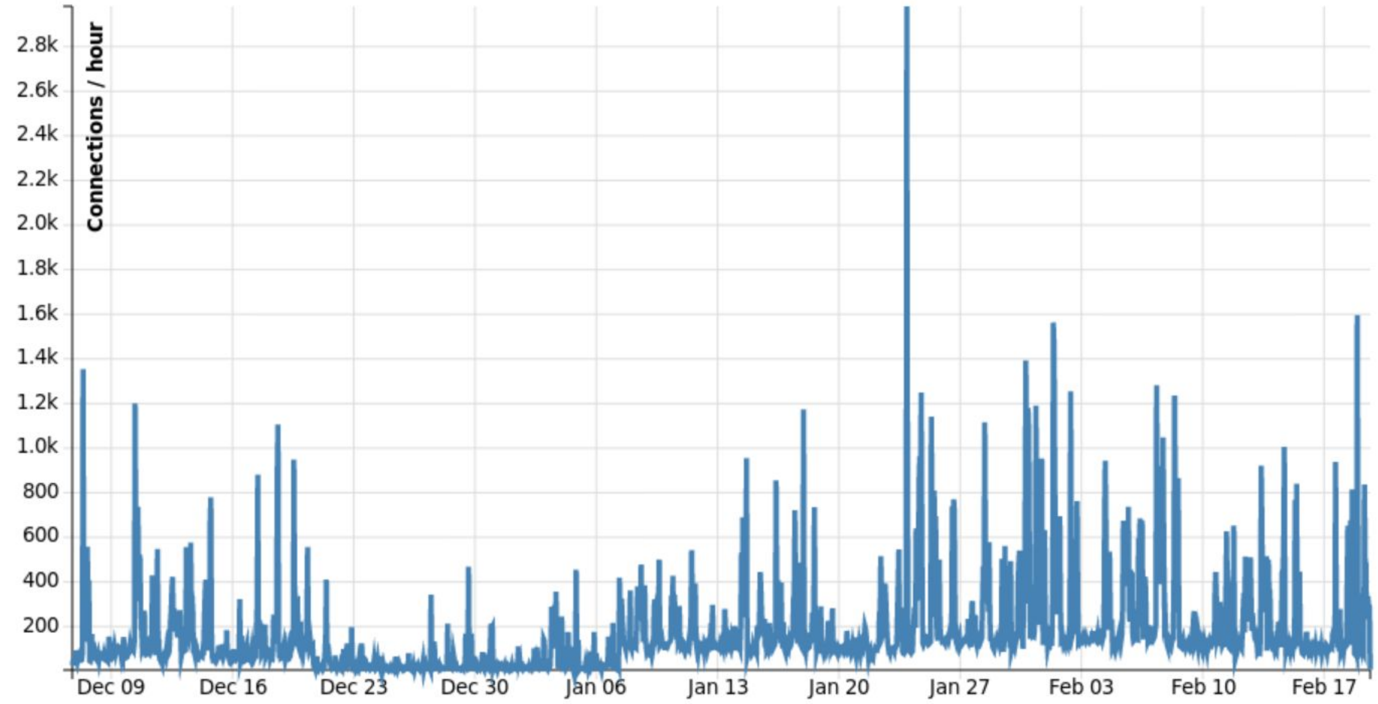
(Visited using Firefox 65 on Linux Fedora 29)

e708f363a76a041d

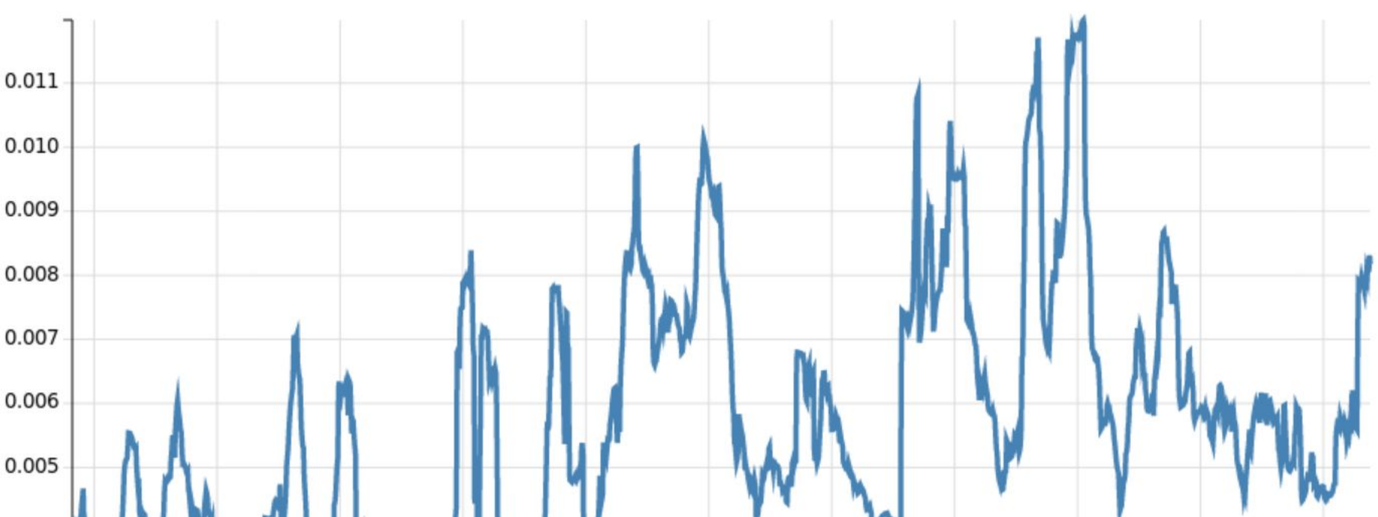
Seen	(all time)	306K times (0.01%)
	(past week)	36K times (0.01%)
Rank	(all time)	566 / 119267
	(past week)	505 / 12607
TLS Version		TLS 1.0
Handshake Version		TLS 1.2
Cipher Suites exact match		TLS_AES_128_GCM_SHA256 (0x1301) TLS_CHACHA20_POLY1305_SHA256 (0x1303) TLS_AES_256_GCM_SHA384 (0x1302) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009) TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033) TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039) TLS_RSA_WITH_AES_128_CBC_SHA (0x002f) TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
Compression Methods		null (0x00)
Extensions exact match		server_name (0x0000) extended_master_secret (0x0017)

Supported Groups exact match	x25519 (0x001d) secp256r1 (0x0017) secp384r1 (0x0018) secp521r1 (0x0019) ffdhe2048 (0x0100) ffdhe3072 (0x0101)
Signature Algorithms	ecdsa_secp256r1_sha256 (0x0403) ecdsa_secp384r1_sha384 (0x0503) ecdsa_secp521r1_sha512 (0x0603) rsa_pss_rsae_sha256 (0x0804) rsa_pss_rsae_sha384 (0x0805) rsa_pss_rsae_sha512 (0x0806) rsa_pkcs1_sha256 (0x0401) rsa_pkcs1_sha384 (0x0501) rsa_pkcs1_sha512 (0x0601) ecdsa_sha1 (0x0203) rsa_pkcs1_sha1 (0x0201)
EC Point Formats	uncompressed (0x00)
ALPN	h2 http/1.1
Key Share	x25519 (0x001d) - 32-byte key secp256r1 (0x0017) - 65-byte key
PSK Key Exchange Modes	psk_dhe_ke (0x01)
Supported Versions exact match	TLS 1.3 (0x0304) TLS 1.2 (0x0303) TLS 1.1 (0x0302) TLS 1.0 (0x0301)
Certificate Compression Algorithms	

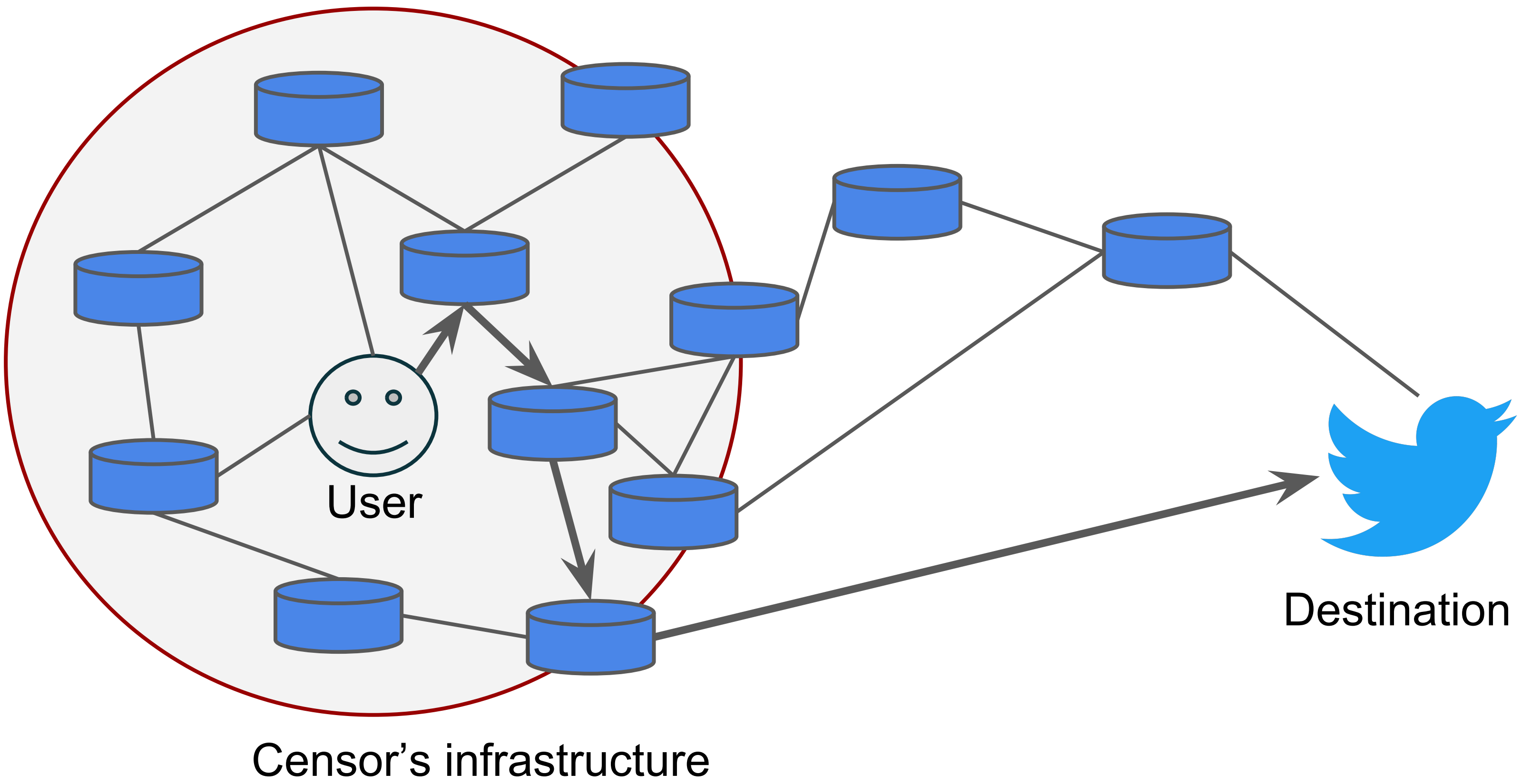
Times seen (per hour)



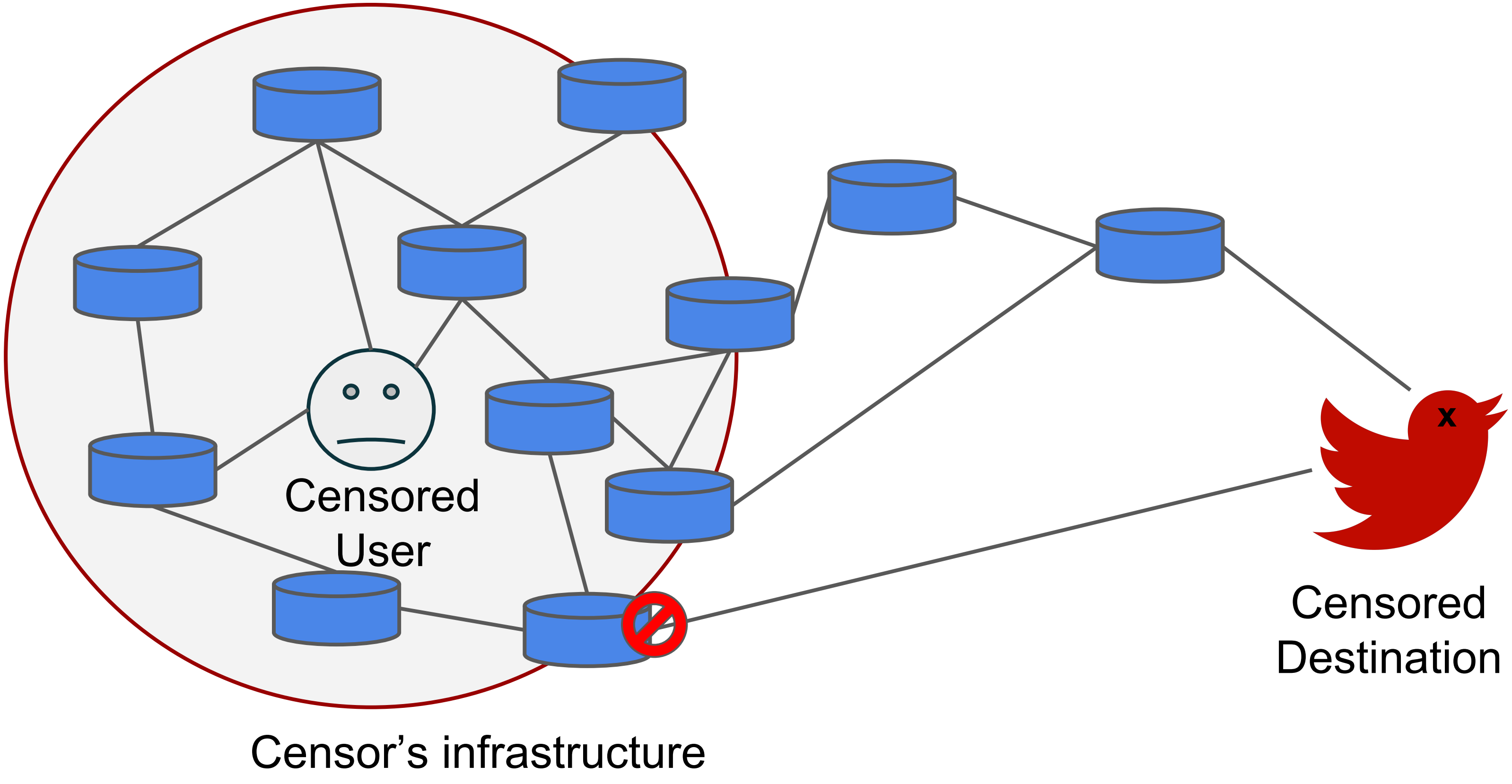
Percent seen (24 hour averaged)



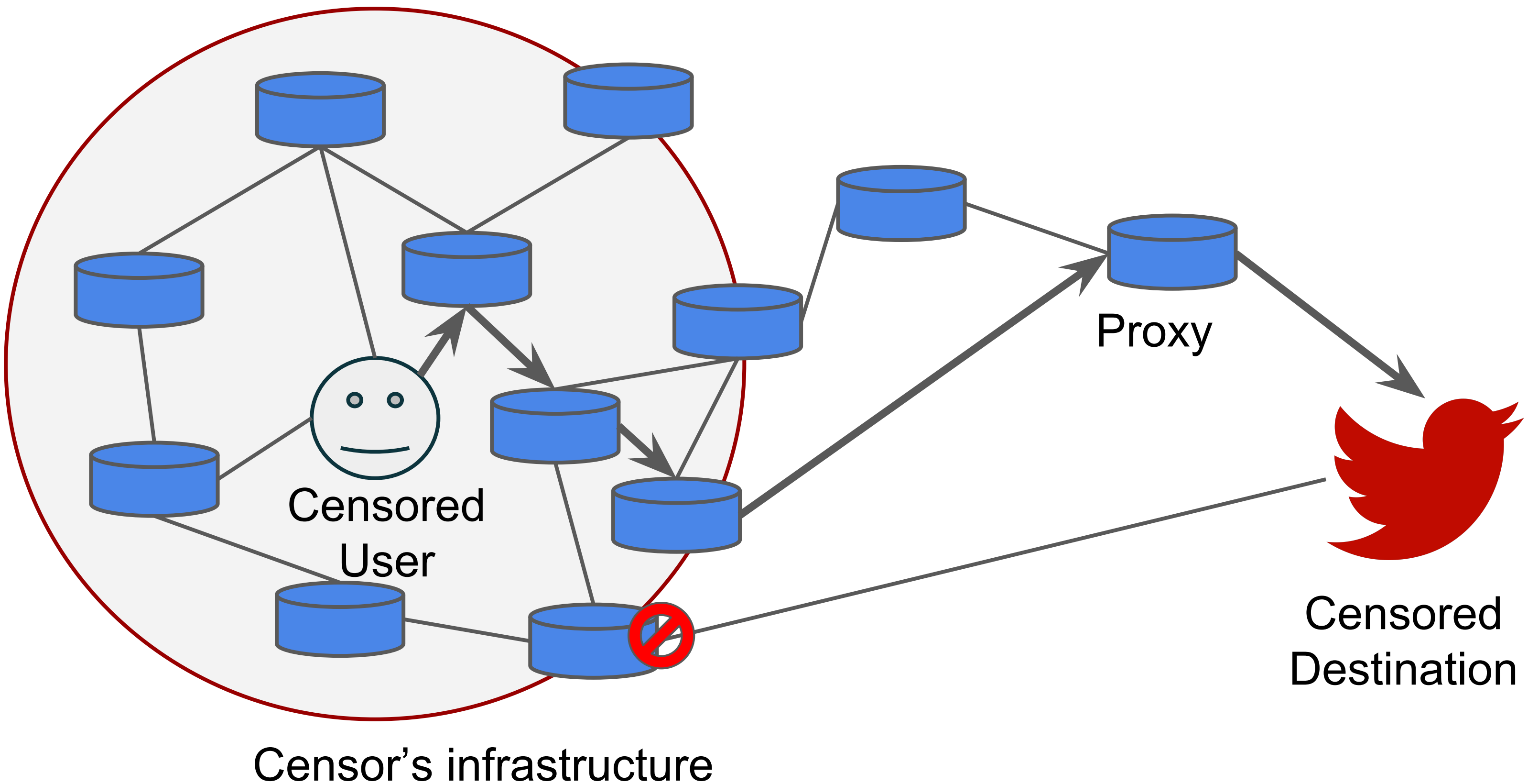
Censorship Threat Model



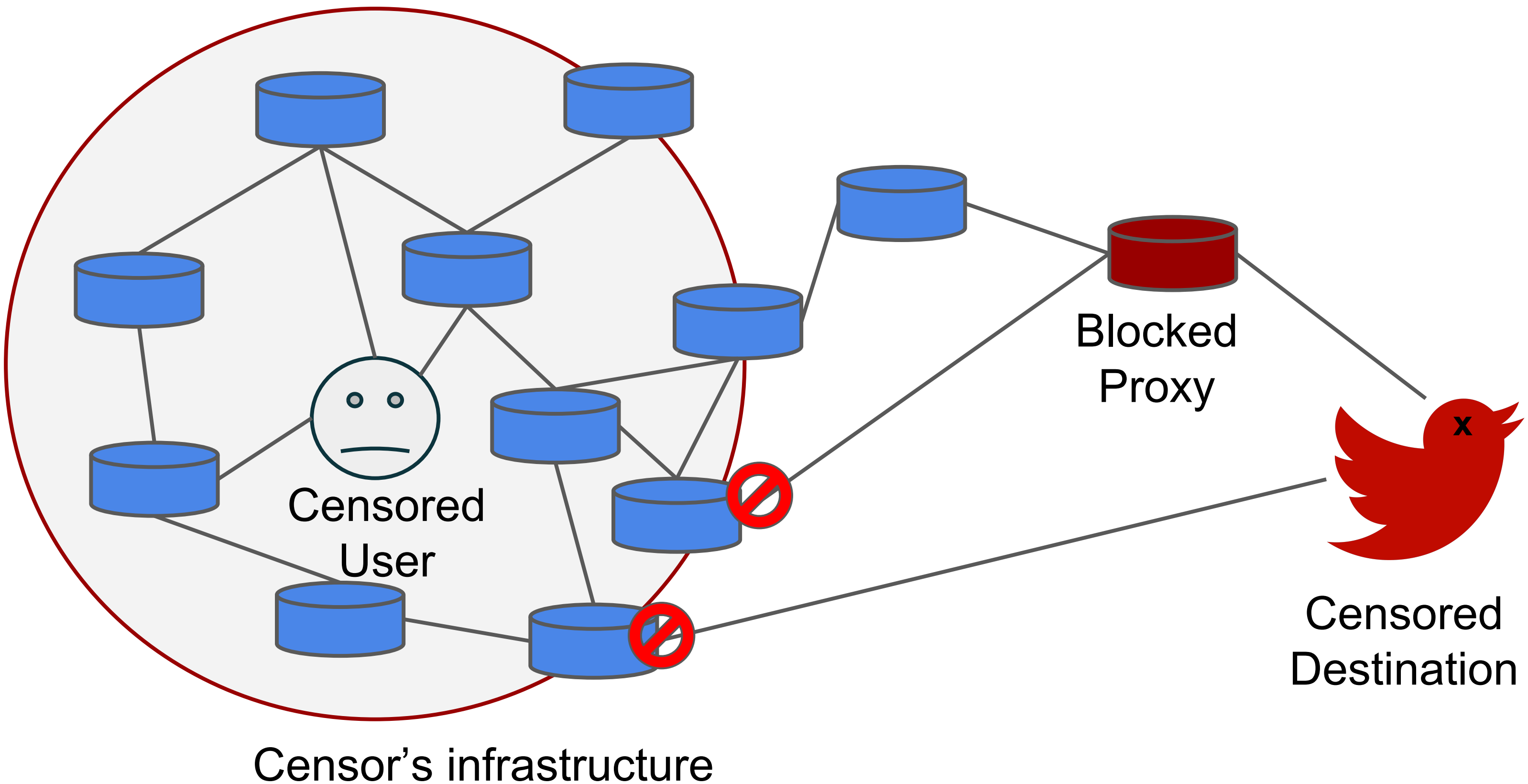
Censorship Threat Model



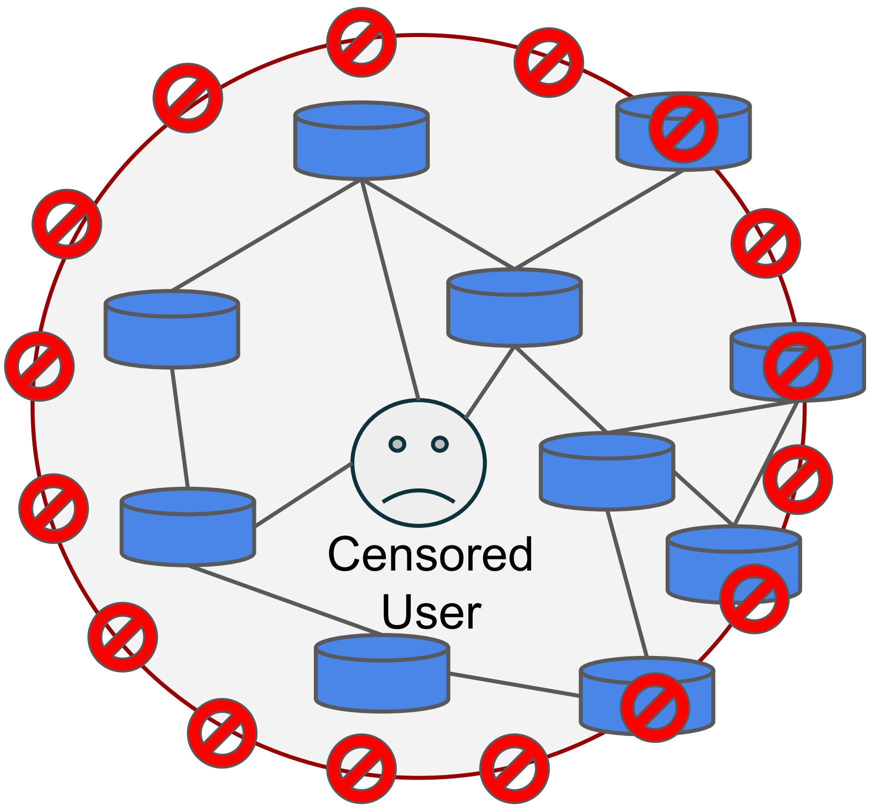
Censorship Threat Model



Censorship Threat Model



It’s all about collateral damage



Overblocking is costly:

- Damages economy
- Upsets productivity
- Angers residents
- Impacts trade