Protocol Misidentification Made Easy with

Format-Transforming Encryption

Kevin Dyer, Portland State University (did most of the hard work)

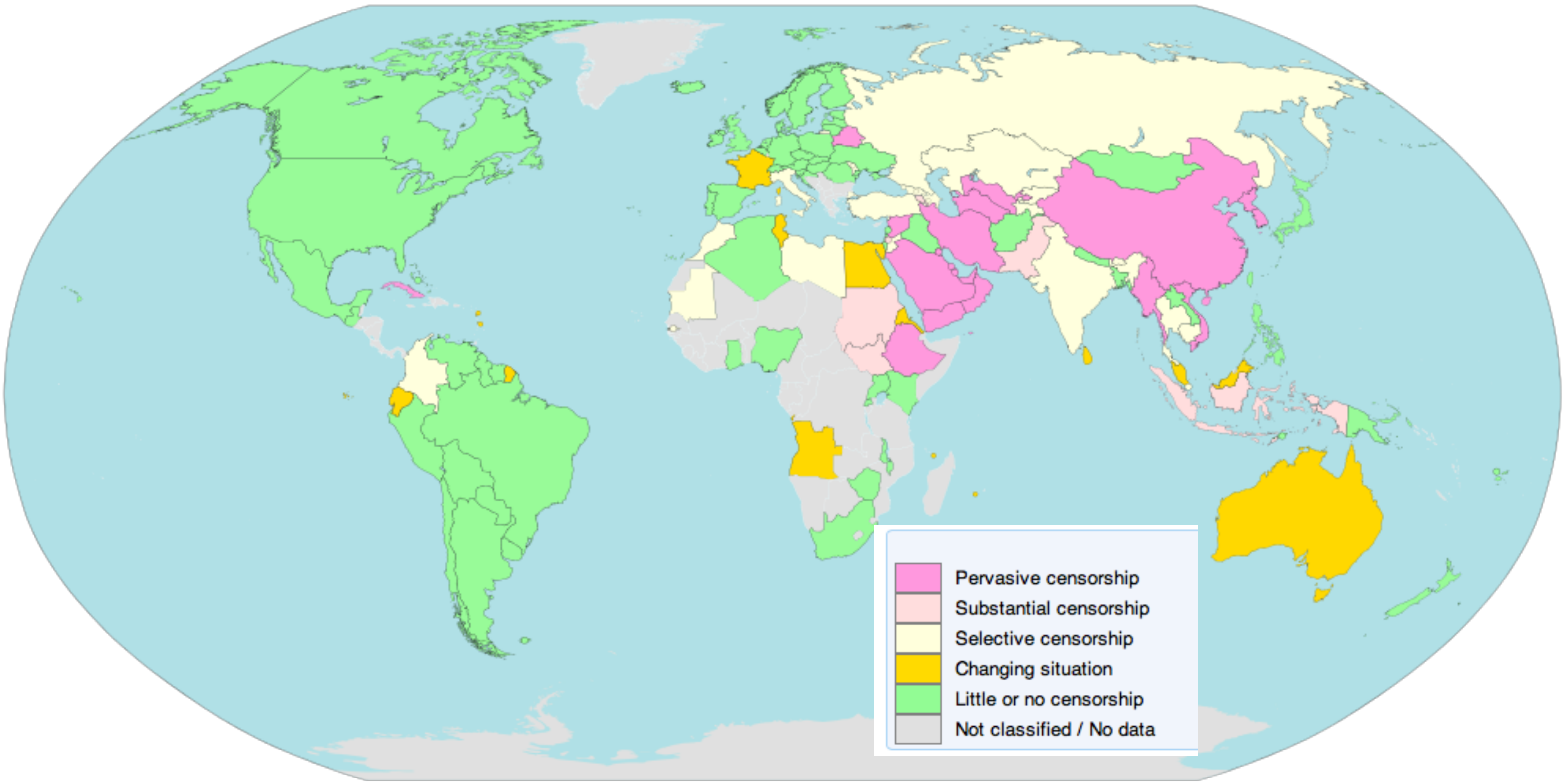
Scott Coull, RedJack

Thomas Ristenpart, University of Wisconsin-Madison

Thomas Shrimpton, Portland State University

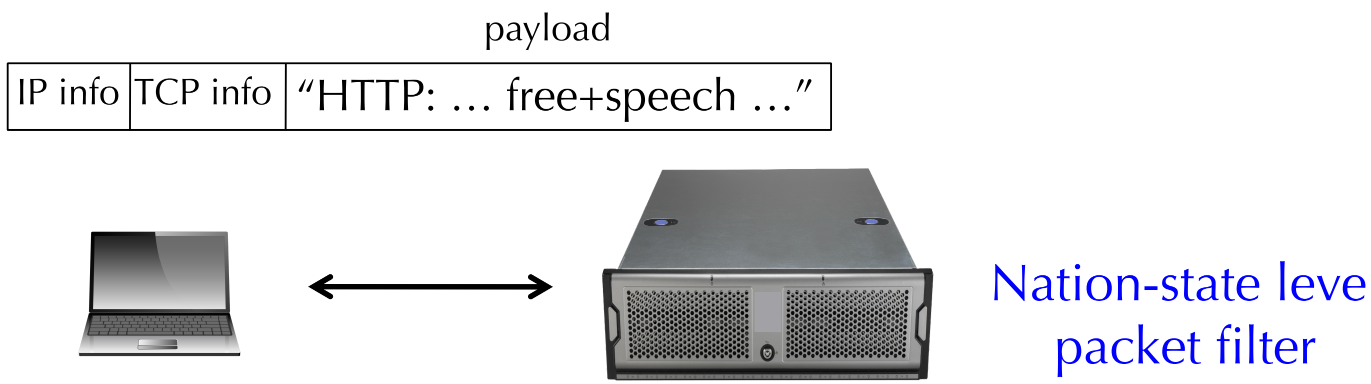
Current Estimates of Internet Censorship

OpenNet Initiative (ONI),
Reporters Without Borders
(via wikipedia; updated Jan 6, 2014)



Magenta-colored countries are “**internet black holes**”:
have heavy censorship of political, social, and news sites, internet tools, etc.

Discriminatory policies enabled by packet filtering



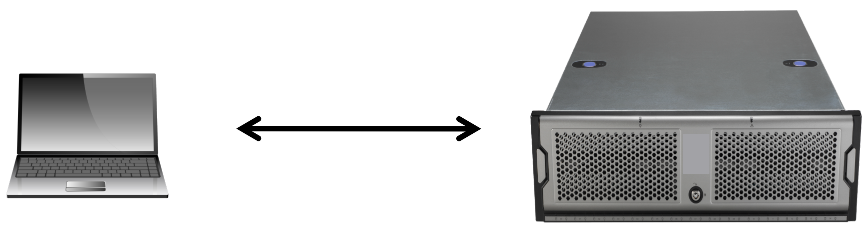
A packet can tell you:

- source address
- destination address/port
- application-level protocols
- keywords in payloads
- …

Tools exist to obfuscate “shallow” information

payload

IP info	TCP info	“HTTP: … free+speech …”



Use a proxy service, e.g. Tor

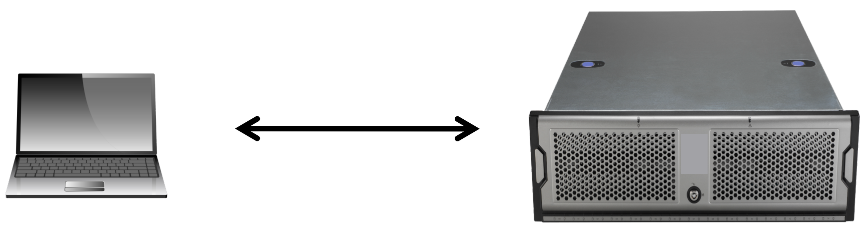
A packet can tell you:

- source address
- destination address/port
- application-level protocols
- keywords in payloads
- …

Modern filters look deeper into the packet: Deep Packet Inspection (DPI)

payload

IP info	TCP info	**"HTTP: … free+speech …"**

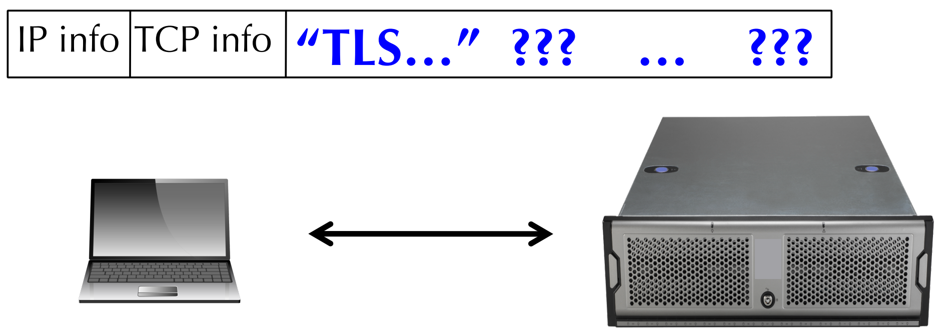


Making payload information unhelpful is the new challenge

A packet can tell you:

- source address
- destination address/port
- application-level protocols
- keywords in payloads
- …

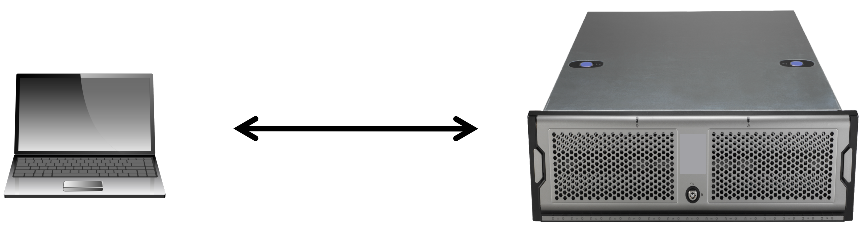
Why not just use an encrypted tunnel?
(TLS, SSH, VPNs, Tor)



Hides the protocol inside the encrypted tunnel...

Why not just use an encrypted tunnel?
(TLS, SSH, VPNs, Tor)

IP info	TCP info	"TLS..." ??? ... ???



Hides the protocol inside the encrypted tunnel...

But use of the encryption protocol is still visible.

Pakistan Bans Encryption

Posted by **Soulskill** on Tuesday August 30, 2011 @06
from the for-undecipherable-reasons dept.

Iran reportedly blocking encrypted Internet traffic

The Iranian government is reportedly blocking access to websites that use the ...

by **Jon Brodkin** - Feb 10 2012, 9:44pm IST

How the Great Firewall of China is Blocking Tor

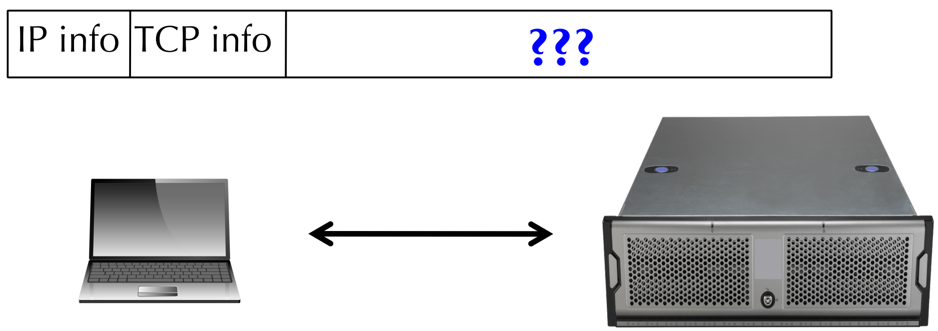
Philipp Winter and Stefan Lindskog

NEWS

Ethiopian government blocks Tor Network online anonymity

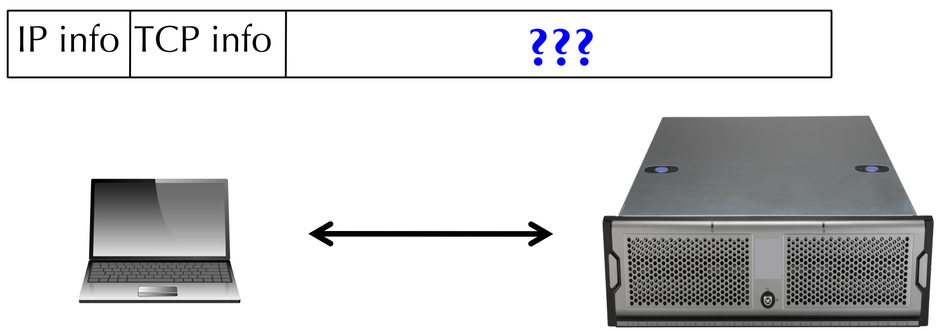
Why not make the *whole payload* look random?

(e.g. with a stream cipher)
(e.g. Tor's "obfs" pluggable transport)



Why not make the *whole payload* look random?

(e.g. with a stream cipher)
(e.g. Tor's "obfs" pluggable transport)



"I don't recognize this as any legitimate protocol."

What happens if DPI allows only whitelisted protocols?

Recent efforts in DPI Circumvention

Stegotorus [Weinberg et al., 2012],

SkypeMorph [Moghaddam et al. 2012],

FreeWave [Houmansadr et al., 2013], etc.

These represent nice steps in the right direction, but

1. **Poor performance:** 16-256Kbps reported (best case)

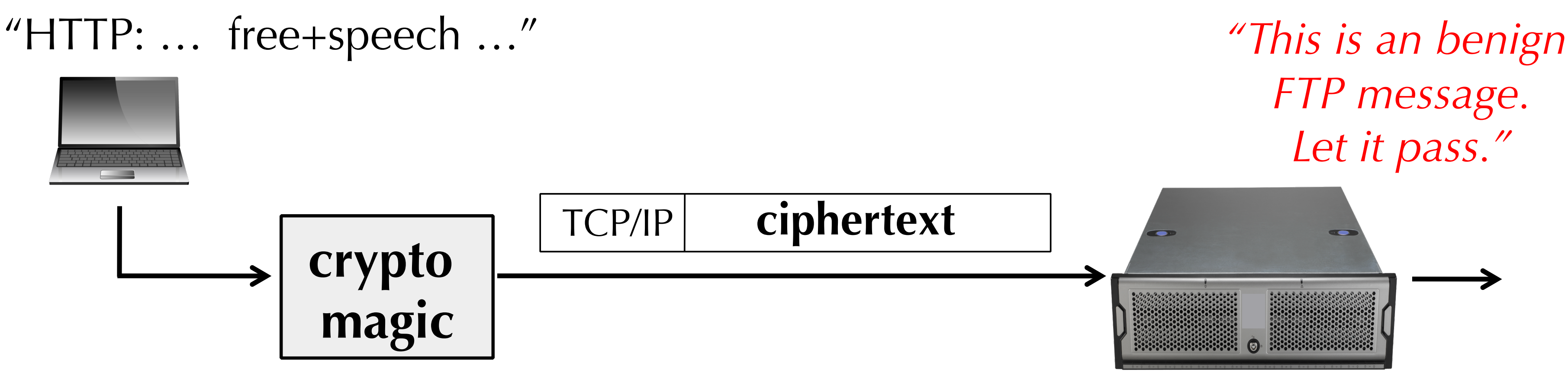
2. **Inflexible:** not quickly adaptable to changes in DPI rules.

   e.g. what if you're using SkypeMorph,
   and Skype becomes blocked? (Ethiopia 2013)

3. **Not empirically validated:** do they work against real DPI?

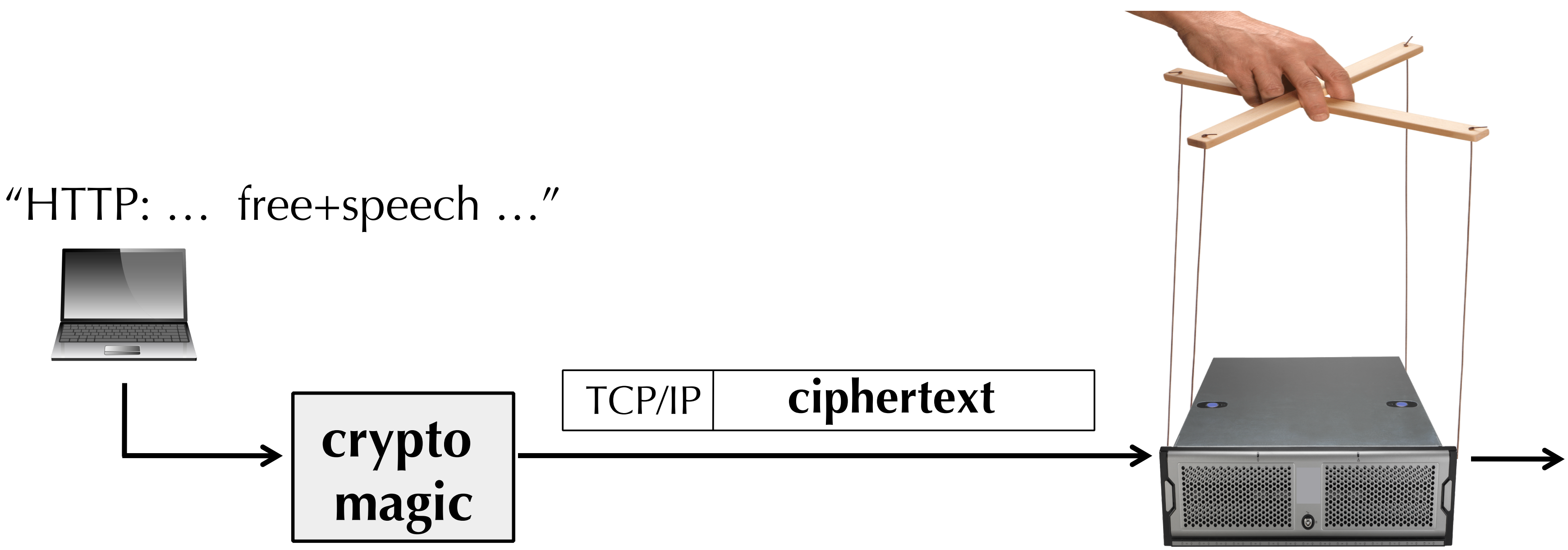
Our goal: to cause real DPI systems to reliably misclassify our traffic

for example: HTTP misclassified as FTP



(and in a way that is flexible and has good throughput/low latency…)

Our goal: to cause real DPI systems to reliably misclassify our traffic as whatever protocol we want.



(and in a way that is flexible and has good throughput/low latency…)

To this end, we:

Introduce a new cryptographic tool, Format Transforming Encryption

Characterize how real DPI systems make classification decisions

Implement an FTE-powered proxy system

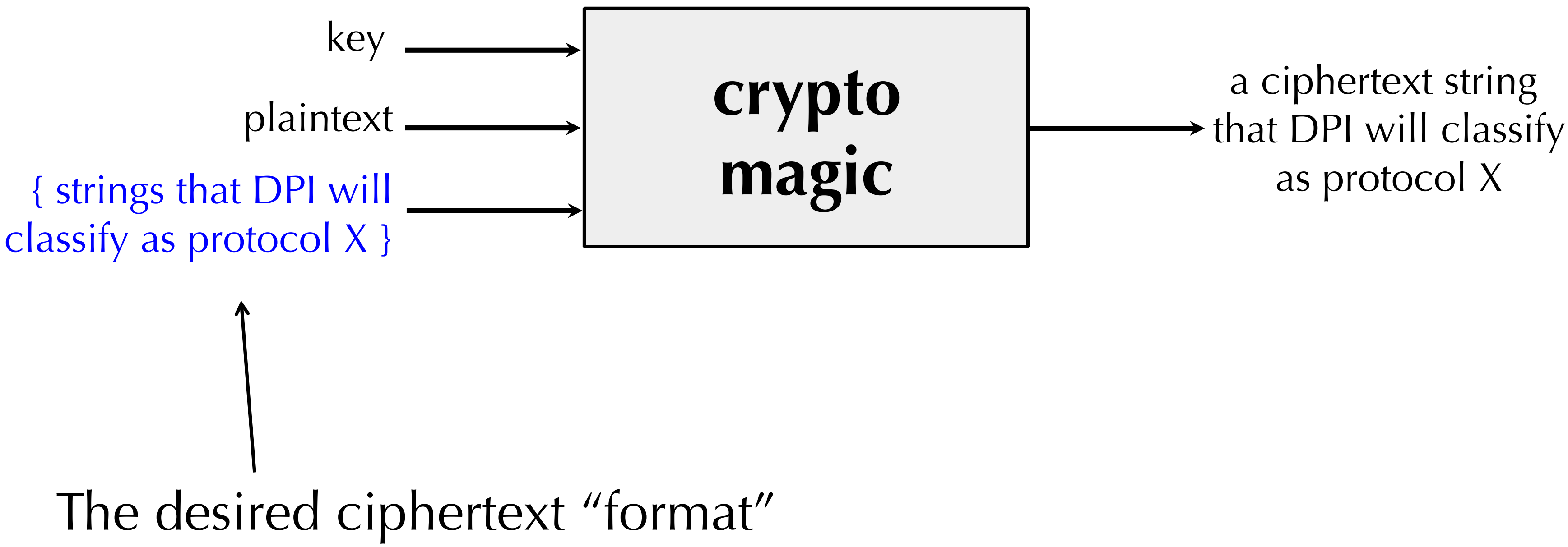
Empirically evaluate FTE against real DPI in the lab

Report on some live “field tests”

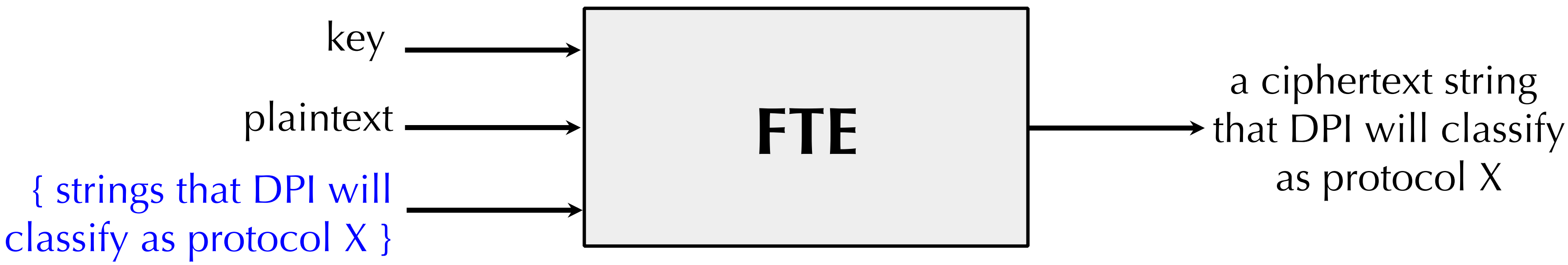

key
plaintext
crypto magic
a ciphertext string that DPI will classify as protocol X

We took inspiration from Format-Preserving Encryption

[Bellare et al., 2009]

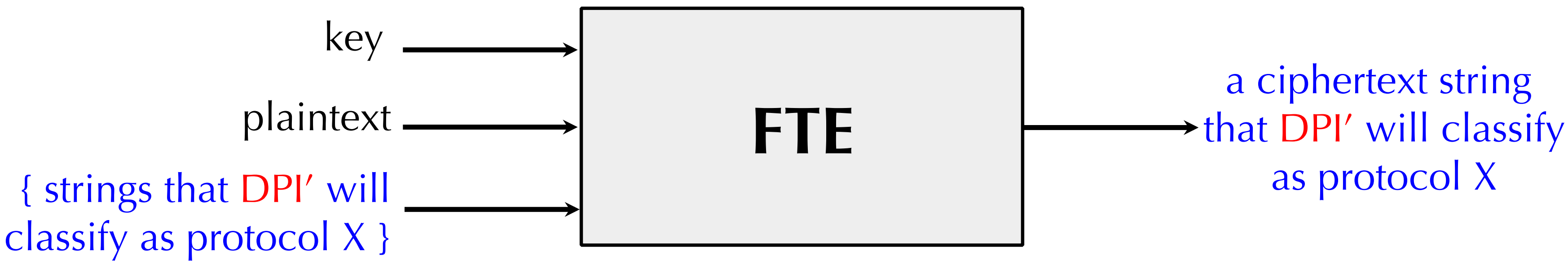


Format-*Transforming* Encryption



Like traditional encryption, with the extra operational requirement that ciphertexts fall within the format.

Ciphertext flexibility is built into the FTE syntax



Conceptually, adapting to new DPI rules requires changing only the format

We wondered:
How do real DPI devices determine to what protocol a message belongs?

"This is an _____ message."

System	Classification Tool	Price
appid		free
l7-filter		free
YAF		free
bro		free
nProbe		~300 Euros
DPI-X		~$10K

Enterprise grade DPI, well-known company

We wondered:
How do real DPI devices determine to what protocol a message belongs?

System	Classification Tool	Price
appid	**Regular expressions**	free
l7-filter	**Regular expressions**	free
YAF	**Regular expressions** (sometimes hierarchical)	free
bro	Simple **regular expression** triage, then additional parsing and heuristics	free
nProbe	Parsing and heuristics (many of them "**regular**")	~300 Euros
DPI-X	**???**	~$10K

Regular languages/expressions figure heavily in state-of-the-art DPI classification tools

Regular-expression-based FTE

Whence the regex?

If the DPI is open source (appid, l7-filter, YAF), extract them!

Build them manually, using RFCs and (when possible) DPI source code.

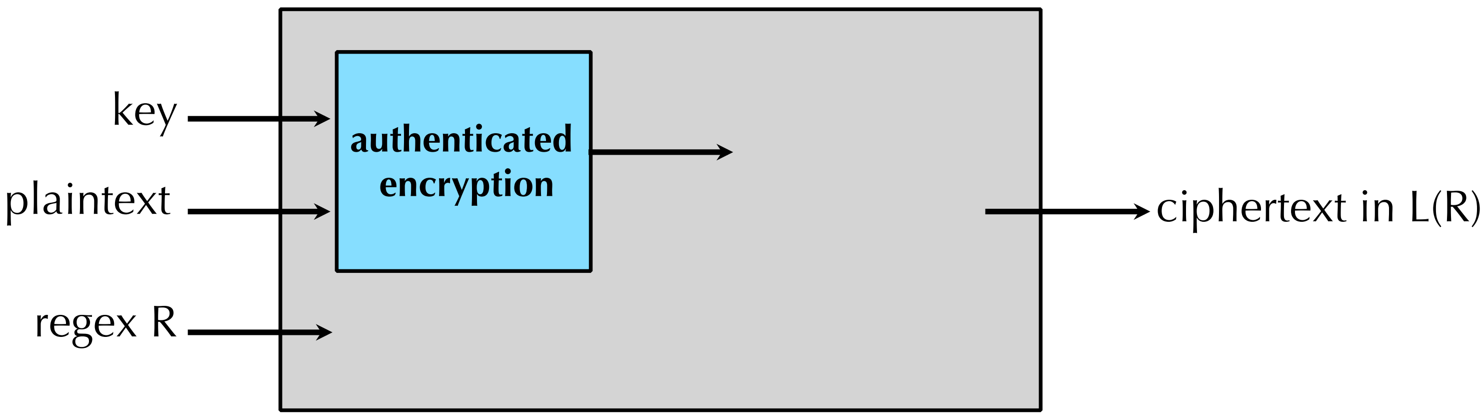
Learn them from traffic that was allowed by the DPI.

Realizing regex-based FTE

How should we realize regex-based FTE?

We want:
- Cryptographic protection for the plaintext
- Ciphertexts in L(R)

Realizing regex-based FTE

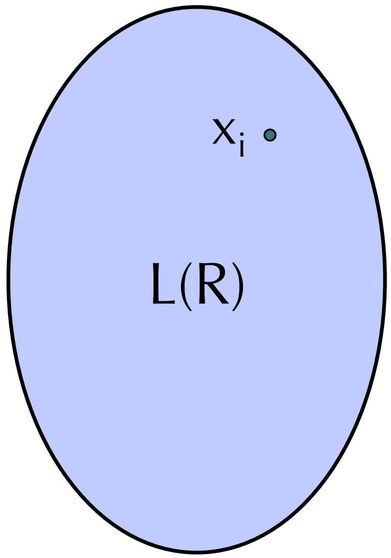


How should we realize regex-based FTE?

We want:
- Cryptographic protection for the plaintext
- Ciphertexts in L(R)

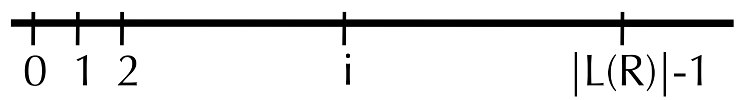
Ranking a Regular Language

[Goldberg, Sipser '85]
[Bellare et al. '09]



Let L(R) be lexicographically ordered

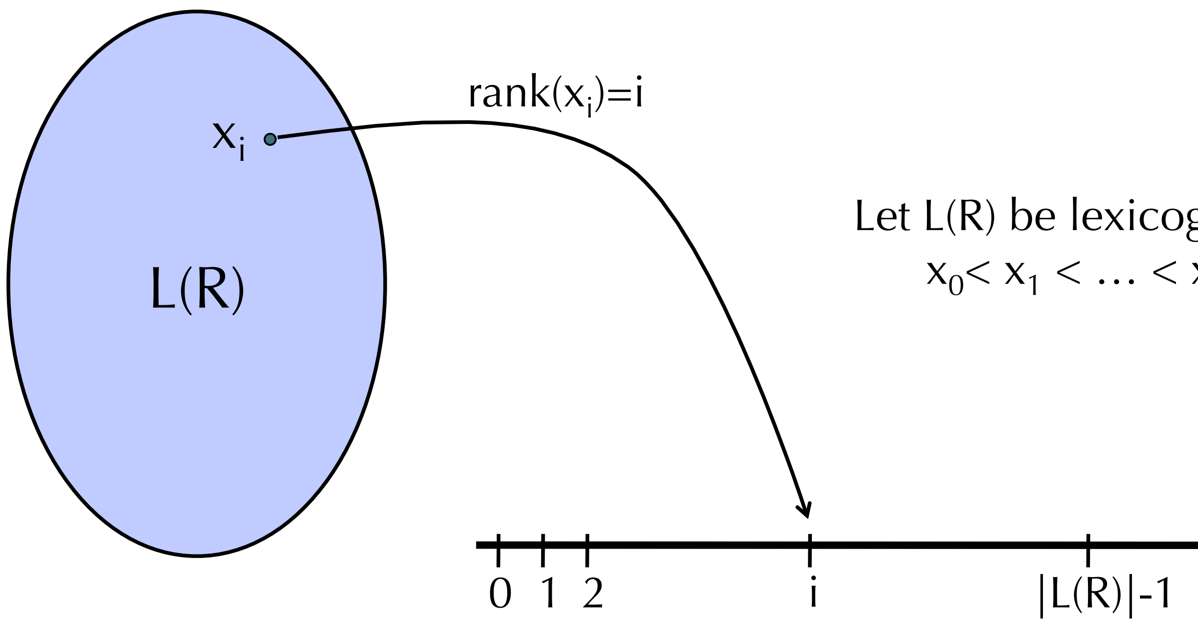
$x_0 < x_1 < \ldots < x_i < \ldots < x_{|L(R)-1|}$



Given a **DFA** for L(R), there are efficient algorithms

Ranking a Regular Language

[Goldberg, Sipser '85]
[Bellare et al. '09]



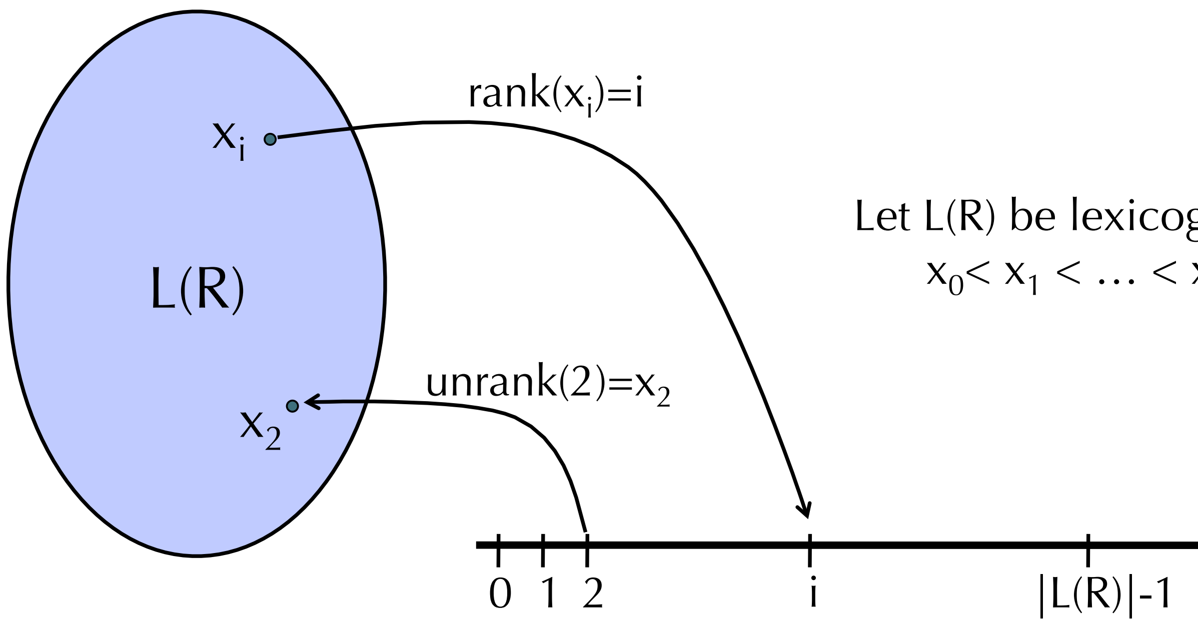
Let L(R) be lexicographically ordered
$x_0 < x_1 < \ldots < x_i < \ldots < x_{|L(R)-1|}$

Given a **DFA** for L(R), there are efficient algorithms

rank: L(R) $\longrightarrow$ {0,1,...,|L(R)|-1}

Ranking a Regular Language

[Goldberg, Sipser '85]
[Bellare et al. '09]



Let L(R) be lexicographically ordered
$x_0 < x_1 < \ldots < x_i < \ldots < x_{|L(R)-1|}$

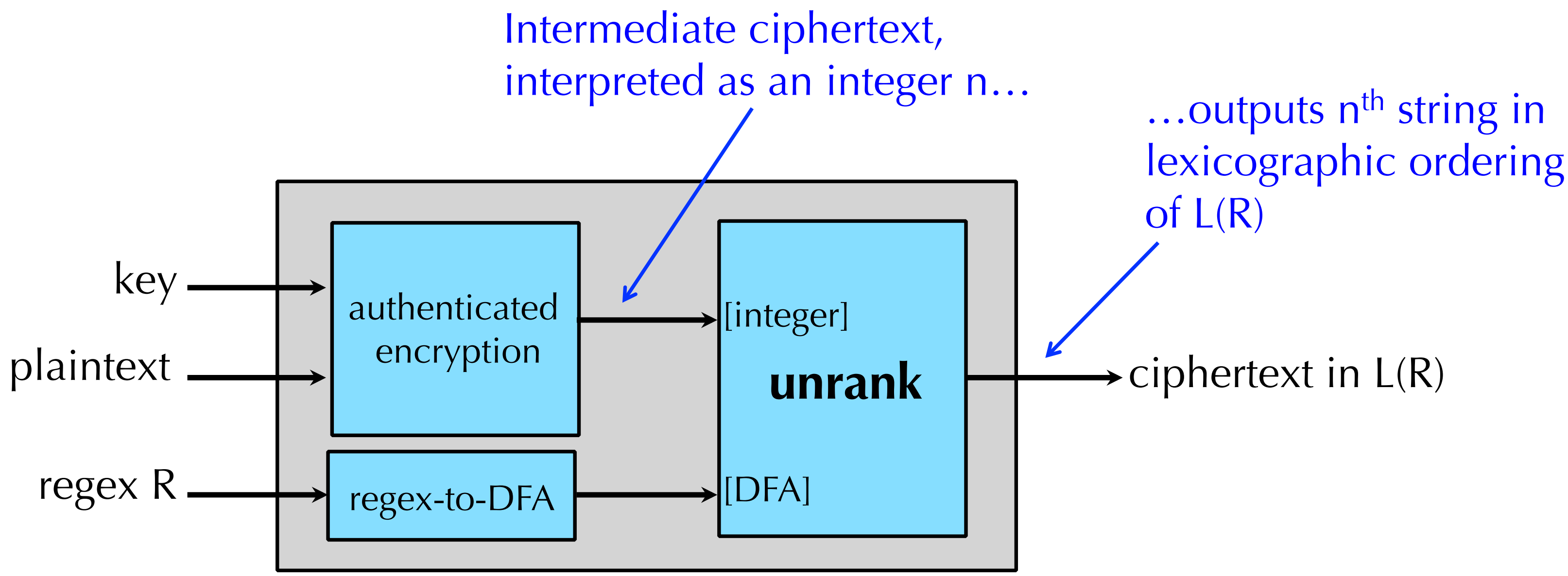
Given a **DFA** for L(R), there are efficient algorithms

rank: L(R) $\longrightarrow$ {0,1,...,|L(R)|-1}
unrank: {0,1,...,|L(R)|-1} $\longrightarrow$ L(R)

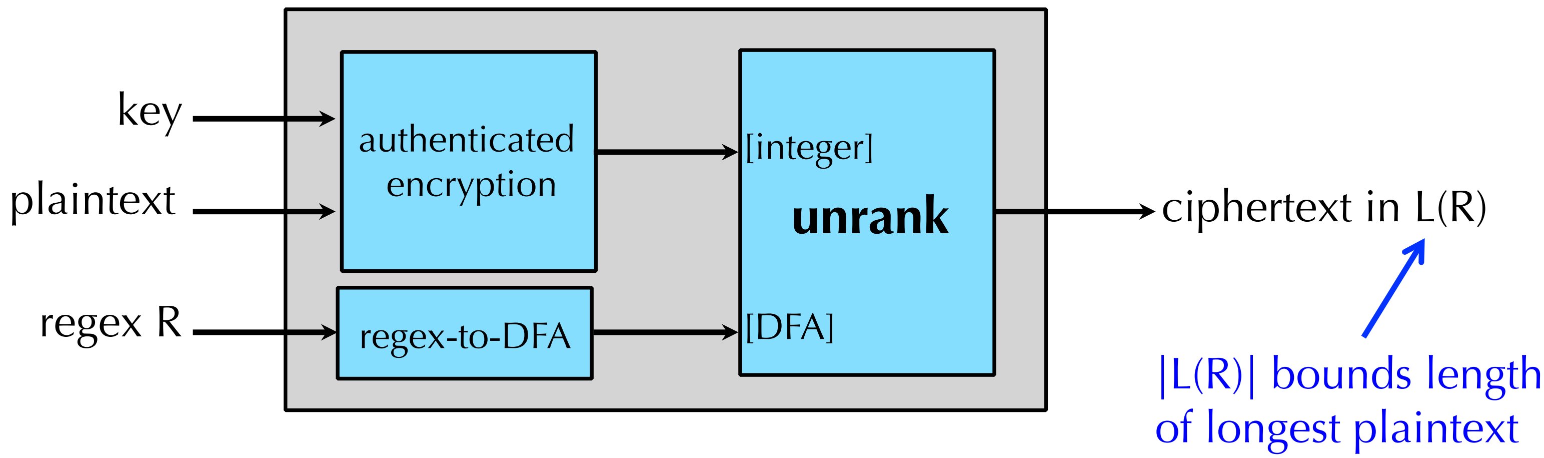
such that rank(unrank(i)) = i
and unrank(rank(x_i)) = x_i

With precomputed tables, rank, unrank are *O*(n)

Realizing regex-based FTE



FTE engineering challenge: large plaintexts



Using very large languages leads to:

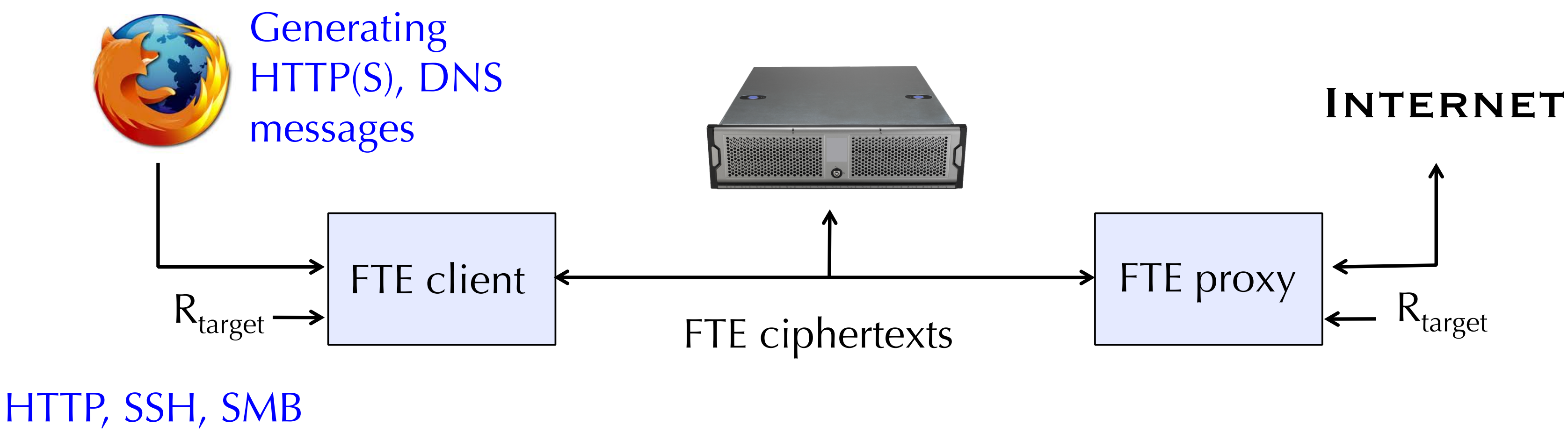
- **large tables** – naively, (#DFA states) x (length of longest plaintext)
- **latency issues** – waiting for long plaintext to buffer

Chunking, and using unrank(C_1), unrank(C_2), unrank(C_3), leads to:

- **receiver-side parsing issues** – how to affect the commas?

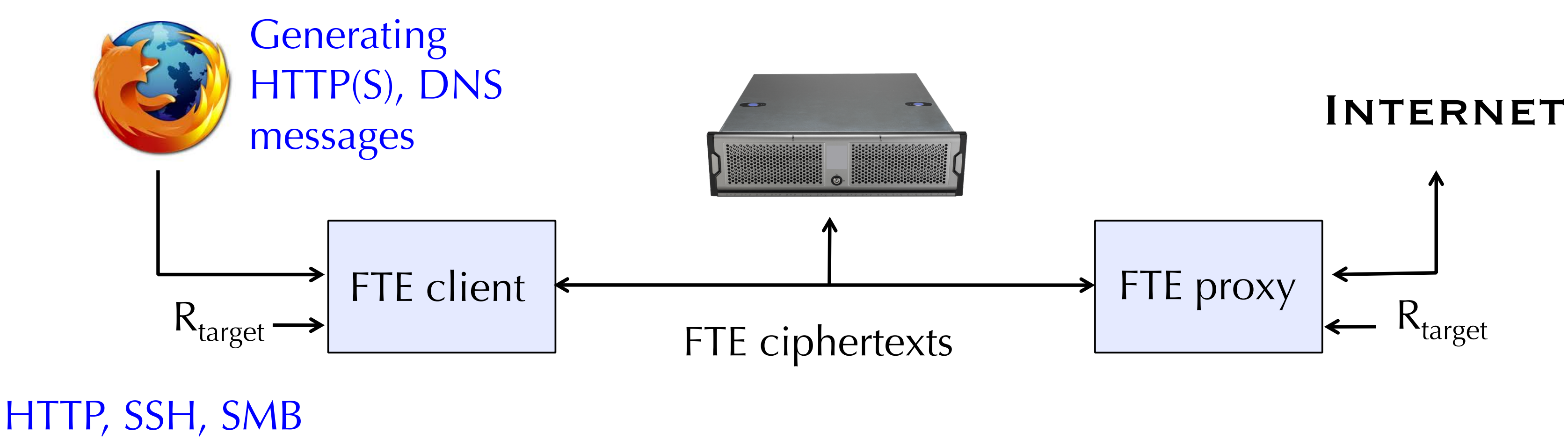
Use case: Browsing the web through an FTE tunnel

FTE "wins" if the DPI classifies the stream it sees as the target protocol



Use case: Browsing the web through an FTE tunnel

FTE “wins” if the DPI classifies the stream it sees as the target protocol



Using each “target” format, we visited each of the Alexa Top 50 websites five times.

We recorded the fraction of times that FTE won, as well as performance data.

Misclassification rates with extracted regex

DPI

regex

	appid	l7-filter	YAF	DPI-X
appid-http				
l7-http				
yaf-http1 yaf-http2				
appid-ssh				
l7-ssh				
yaf-ssh1 yaf-ssh2				
appid-smb				
l7-smb				
yaf-smb1 yaf-smb2				

Misclassification rates with extracted regex

DPI

regex

	appid	l7-filter	YAF	DPI-X
appid-http	**1.0**	0.0	1.0	1.0
l7-http	0.0	**1.0**	0.16	1.0
yaf-http1	0.0	0.0	**1.0**	1.0
yaf-http2	0.0	0.0	**1.0**	1.0
appid-ssh	**1.0**	0.32	1.0	1.0
l7-ssh	0.16	**1.0**	0.16	1.0
yaf-ssh1	1.0	0.21	**1.0**	1.0
yaf-ssh2	1.0	0.31	**1.0**	1.0
appid-smb	**1.0**	1.0	1.0	1.0
l7-smb	0.0	**1.0**	0.38	1.0
yaf-smb1	0.0	0.04	**1.0**	1.0
yaf-smb2	0.0	0.04	**1.0**	1.0

Misclassification rates with extracted regex

DPI (columns) / regex (rows)

	appid	l7-filter	YAF	DPI-X
appid-http	**1.0**	0.0	1.0	1.0
l7-http	0.0	**1.0**	0.16	1.0
yaf-http1 **yaf-http2**	0.0 0.0	0.0 0.0	**1.0** **1.0**	1.0 1.0
appid-ssh	**1.0**	0.32	1.0	1.0
l7-ssh	0.16	**1.0**	0.16	1.0
yaf-ssh1 **yaf-ssh2**	1.0 1.0	0.21 0.31	**1.0** **1.0**	1.0 1.0
appid-smb	**1.0**	1.0	1.0	1.0
l7-smb	0.0	**1.0**	0.38	1.0
yaf-smb1 **yaf-smb2**	0.0 0.0	0.04 0.04	**1.0** **1.0**	1.0 1.0

Since these all have 1.0 on the diagonals,
we made "intersection" regexs for HTTP, SSH, SMB,
and **got 1.0 everywhere**

Misclassification rates with extracted regex

DPI

regex

	appid	l7-filter	YAF	DPI-X
appid-http	1.0	0.0	1.0	**1.0**
l7-http	0.0	1.0	0.16	**1.0**
yaf-http1	0.0	0.0	1.0	**1.0**
yaf-http2	0.0	0.0	1.0	**1.0**
appid-ssh	1.0	0.32	1.0	**1.0**
l7-ssh	0.16	1.0	0.16	**1.0**
yaf-ssh1	1.0	0.21	1.0	**1.0**
yaf-ssh2	1.0	0.31	1.0	**1.0**
appid-smb	1.0	1.0	1.0	**1.0**
l7-smb	0.0	1.0	0.38	**1.0**
yaf-smb1	0.0	0.04	1.0	**1.0**
yaf-smb2	0.0	0.04	1.0	**1.0**

!

Misclassification rates with manual/learned regex

Built manually, using RFCs and (when possible) DPI source code.

DPI

regex

	appid	l7-filter	YAF	DPI-X	bro	nProbe
manual-http						
manual-ssh						
manual-smb						
learned-http						
learned-ssh						
learned-smb						

Learned (via simple technique) from traffic that was allowed by the DPI.

Misclassification rates with manual/learned regex

DPI

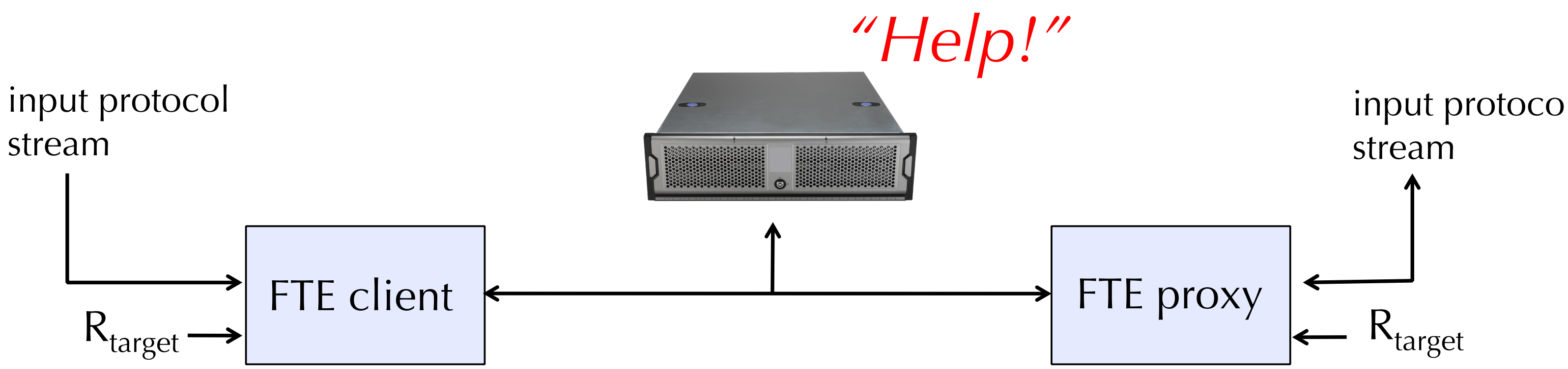
regex

	appid	l7-filter	YAF	DPI-X	bro	nProbe
manual-http						
manual-ssh						
manual-smb						
learned-http						
learned-ssh						0.0
learned-smb						

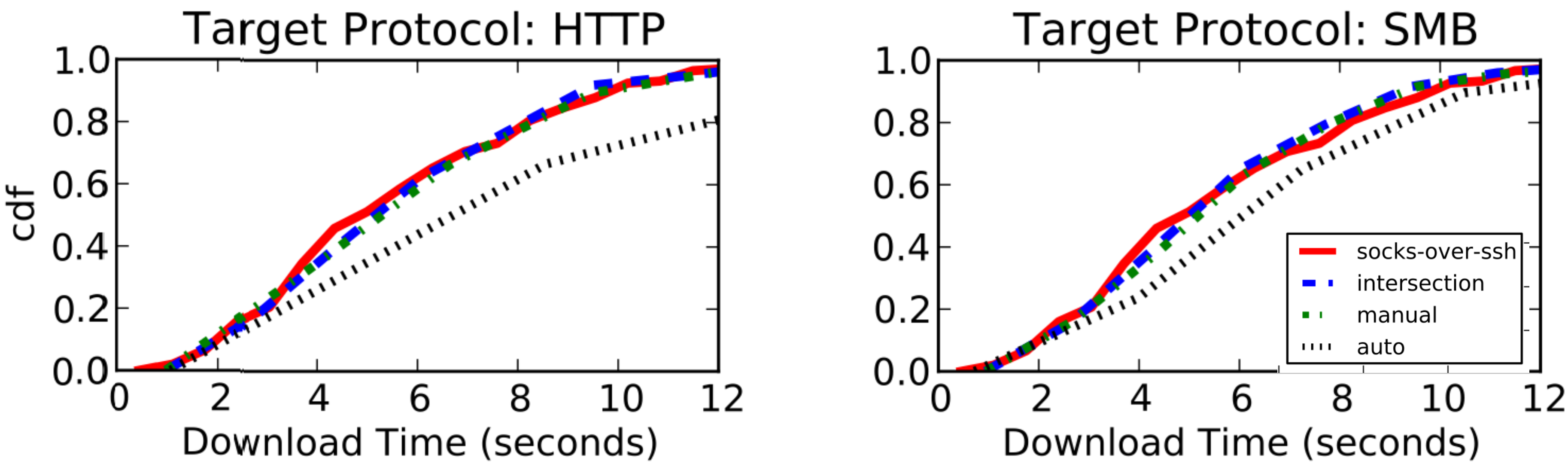
1.0

(except this, which we explain in the paper)

Punchline: regex-based FTE can make real DPI say whatever we want it to.

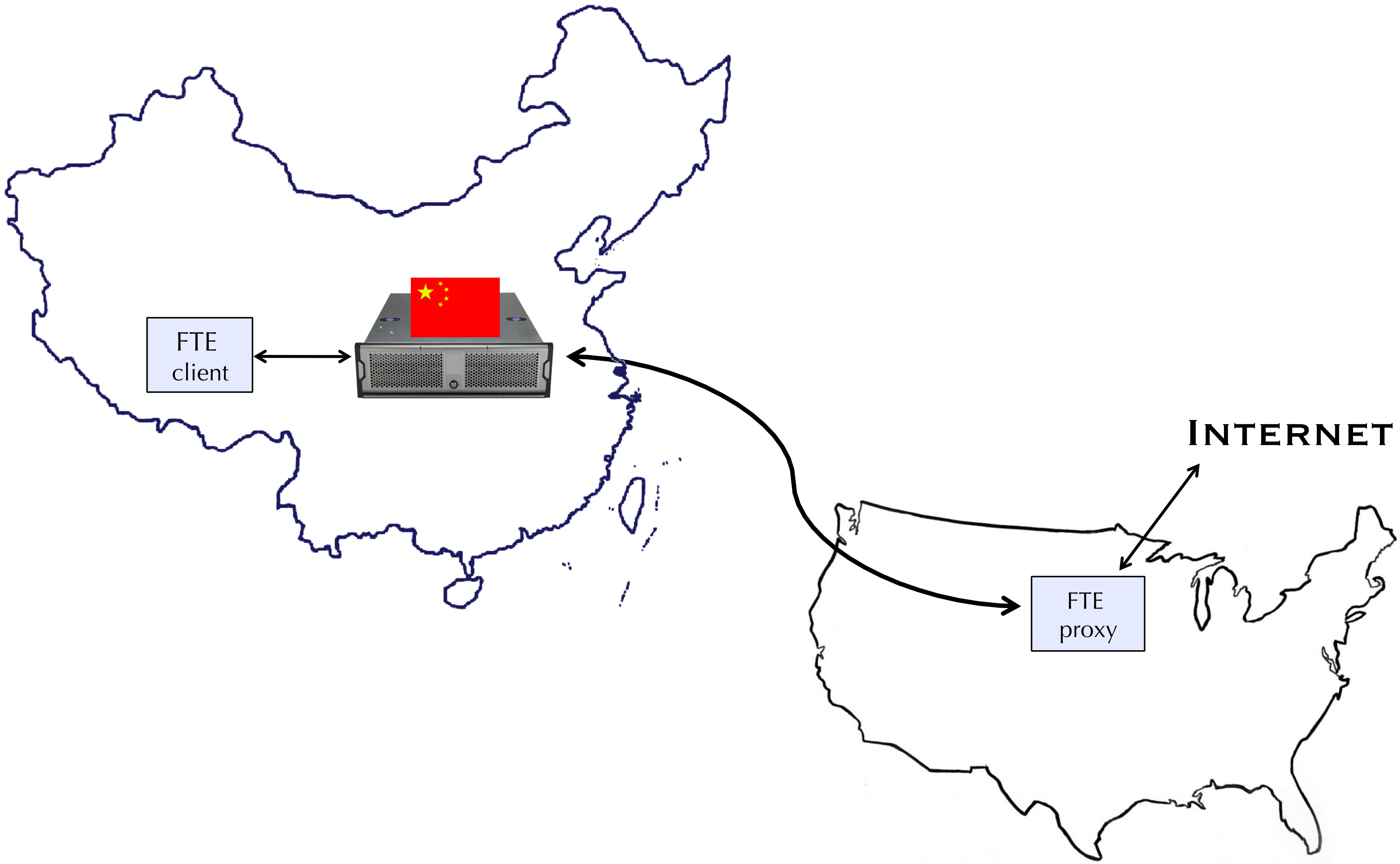


Web-browsing performance



Punchline: FTE or SSH tunnel result in the same user web-browsing experience

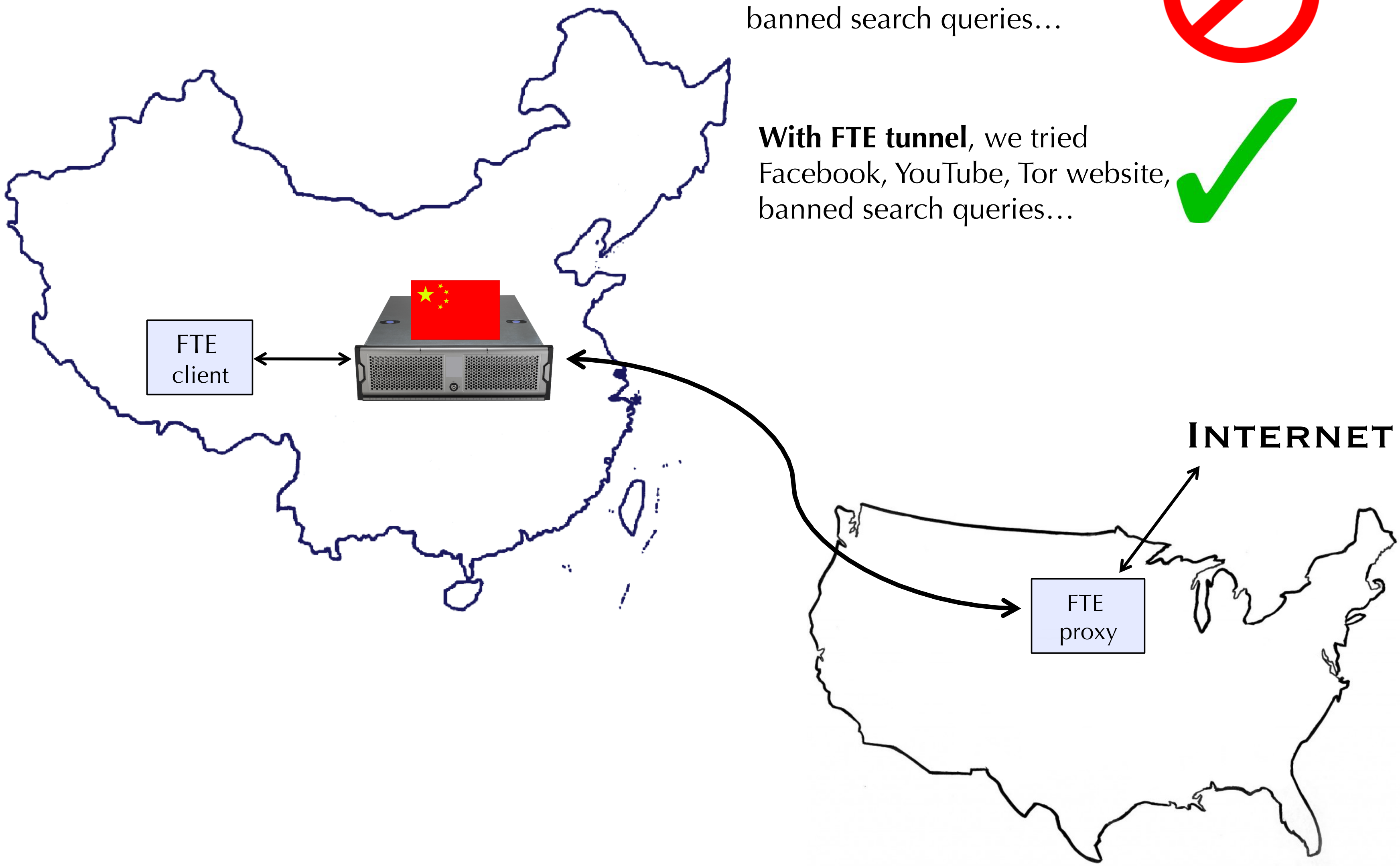
A field test…



A field test...

Without FTE tunnel, we tried Facebook, YouTube, Tor website, banned search queries…

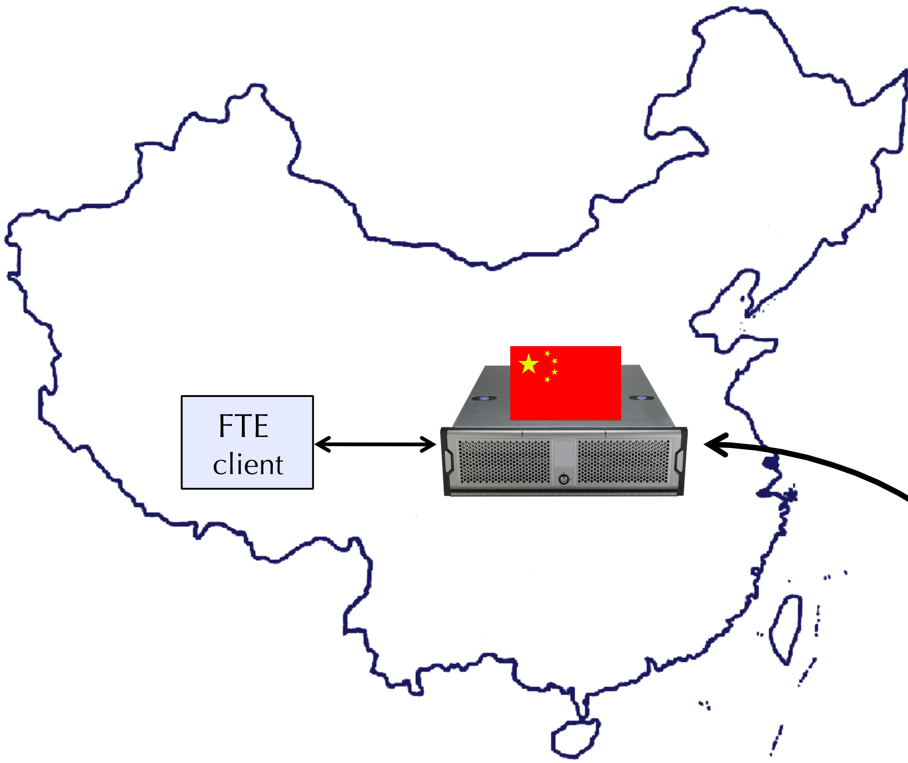
With FTE tunnel, we tried Facebook, YouTube, Tor website, banned search queries…

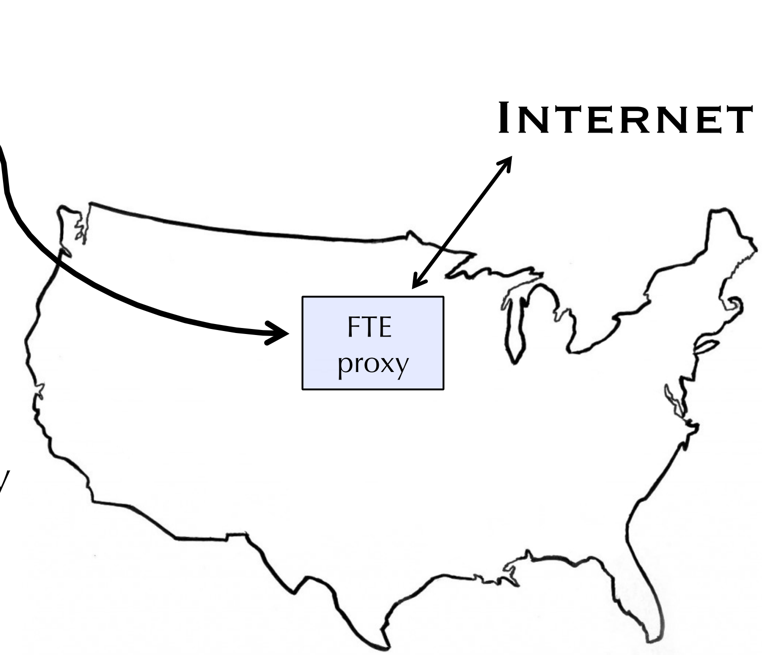


A field test…

Without FTE tunnel, we tried
Facebook, YouTube, Tor website,
banned search queries…

With FTE tunnel, we tried
Facebook, YouTube, Tor website,
banned search queries…





Used FTE to download Tor bundle:

Tor without FTE: "active blacklisting" attack on proxy
Tor through FTE: no problems

Ran various tests every 5 minutes for one month,
no sign of detection in logs. (We shut it down after that.)

FTE is open source,
runs on multiple platforms/OS,
and **fully integrated with** Tor

We even have a nice website:
https://fteproxy.org/

Get it, run it, help us make it better!