Obfuscated Key Exchange

Felix Günther, Douglas Stebila, **Shannon Veitch**

27 March 2024
Real World Crypto 2024
Toronto, Canada

Are you currently using one of these services?

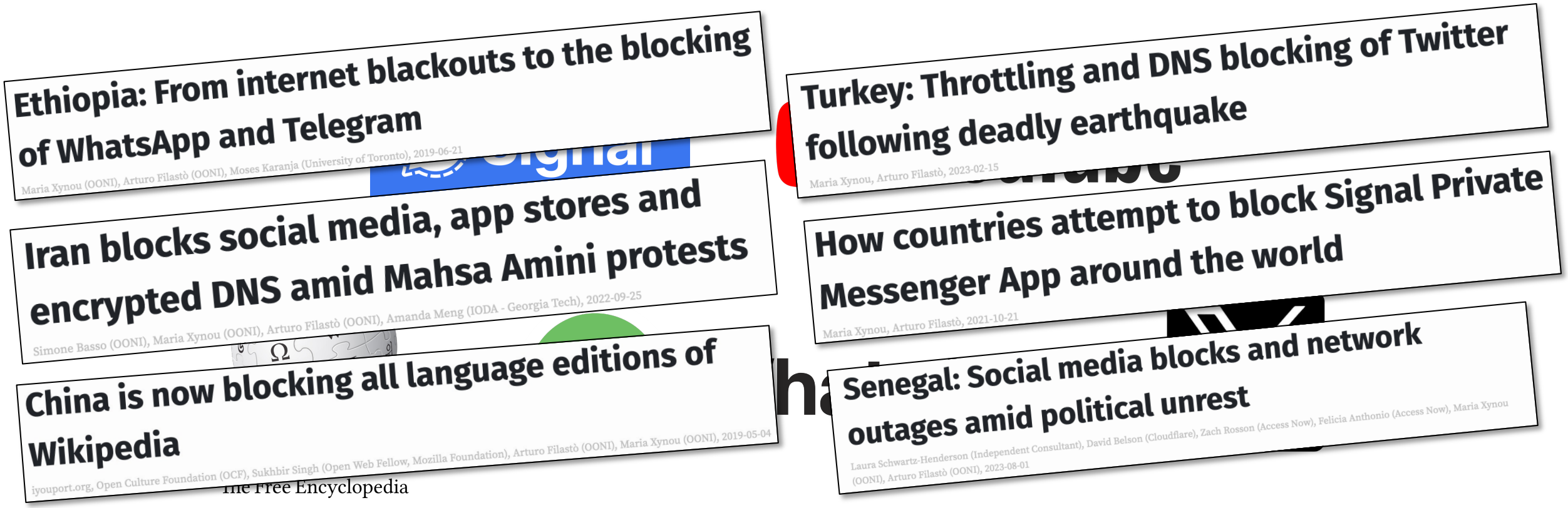
Signal

YouTube

WIKIPEDIA
The Free Encyclopedia

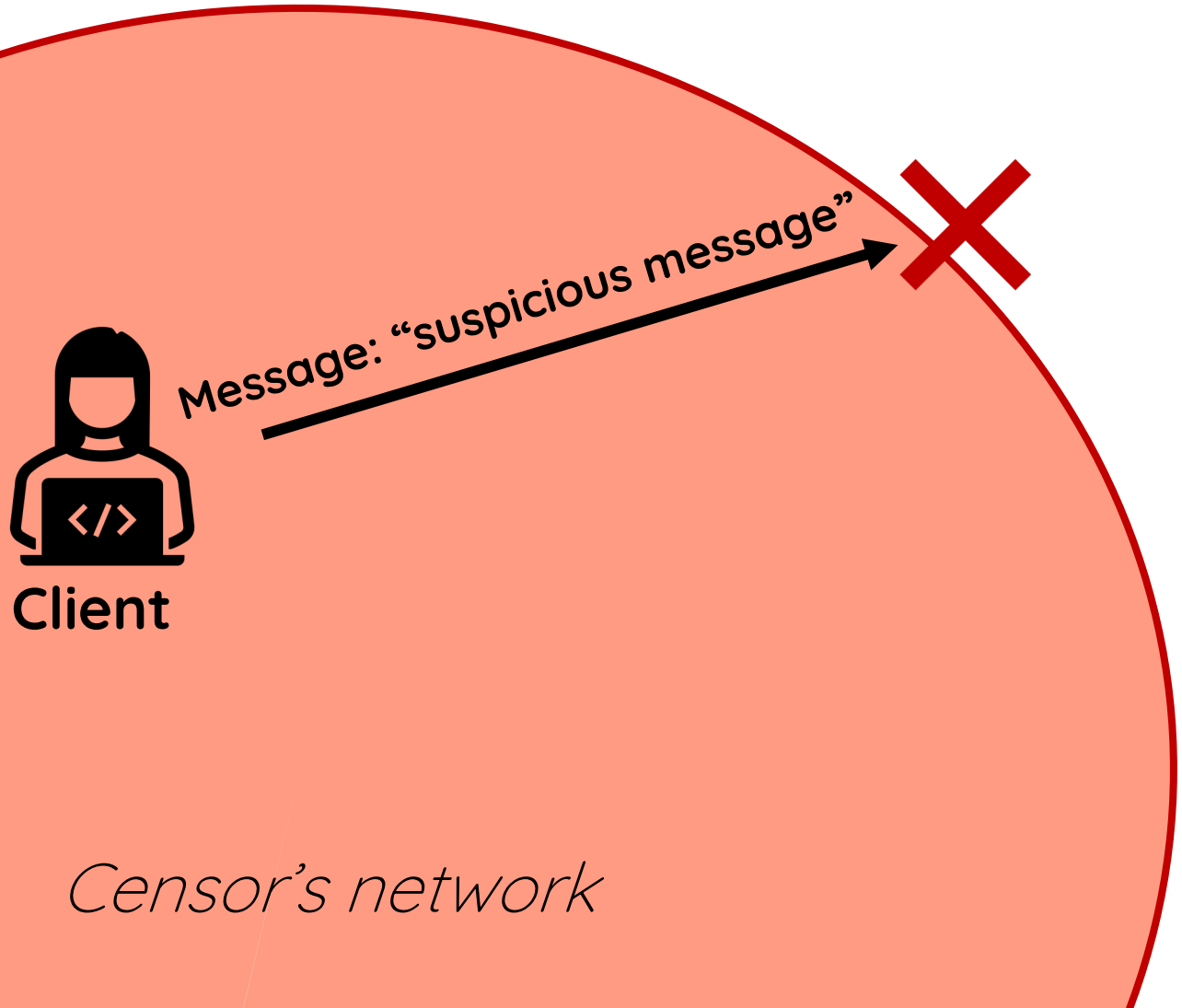
WhatsApp

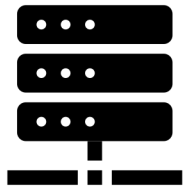
Are you currently using one of these services?



Reports from https://ooni.org/reports/

Setting



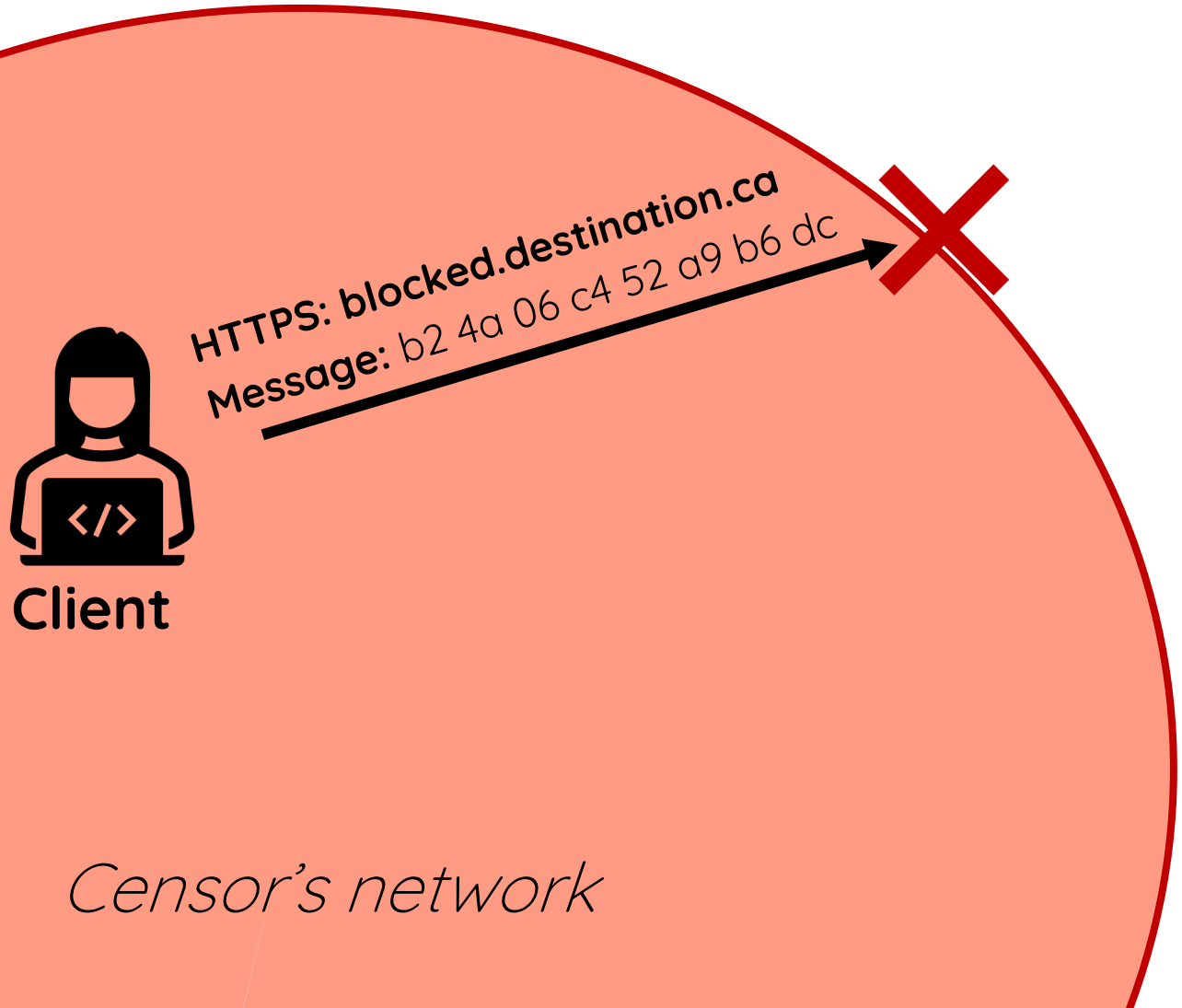


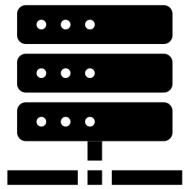
blocked.destination.ca

Censor’s techniques:

1. Detection by (plaintext) content

Setting



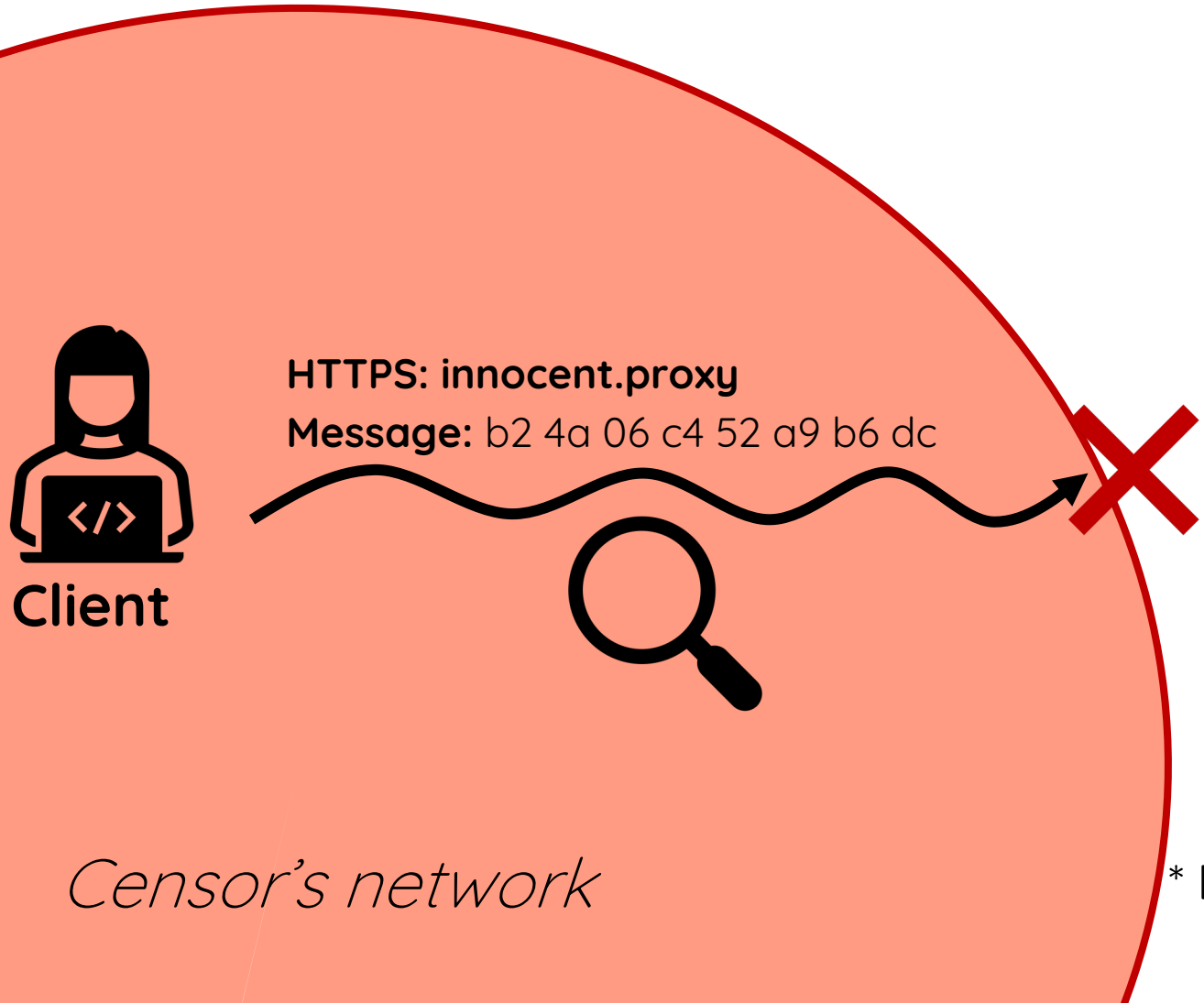


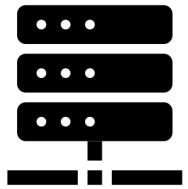
blocked.destination.ca

Censor's techniques:

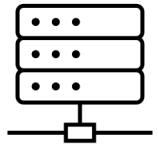
1. Detection by (plaintext) content
2. Detection by address

Setting





blocked.destination.ca



innocent.proxy

Censor's techniques:

1. Detection by (plaintext) content
2. Detection by address
3. Detection by behaviour*

* Deep packet inspection, active probing, etc.

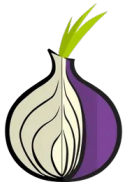
How to: evade detection by behaviour

1. Look like nothing
   - Strategy employed by *fully encrypted protocols*


Shadowsocks


V2ray


Outline


lyrebird (obfsproxy)


Psiphon

2. Look like allowed traffic
   - e.g., tunneling in TLS/Skype traffic

3. Be allowed traffic

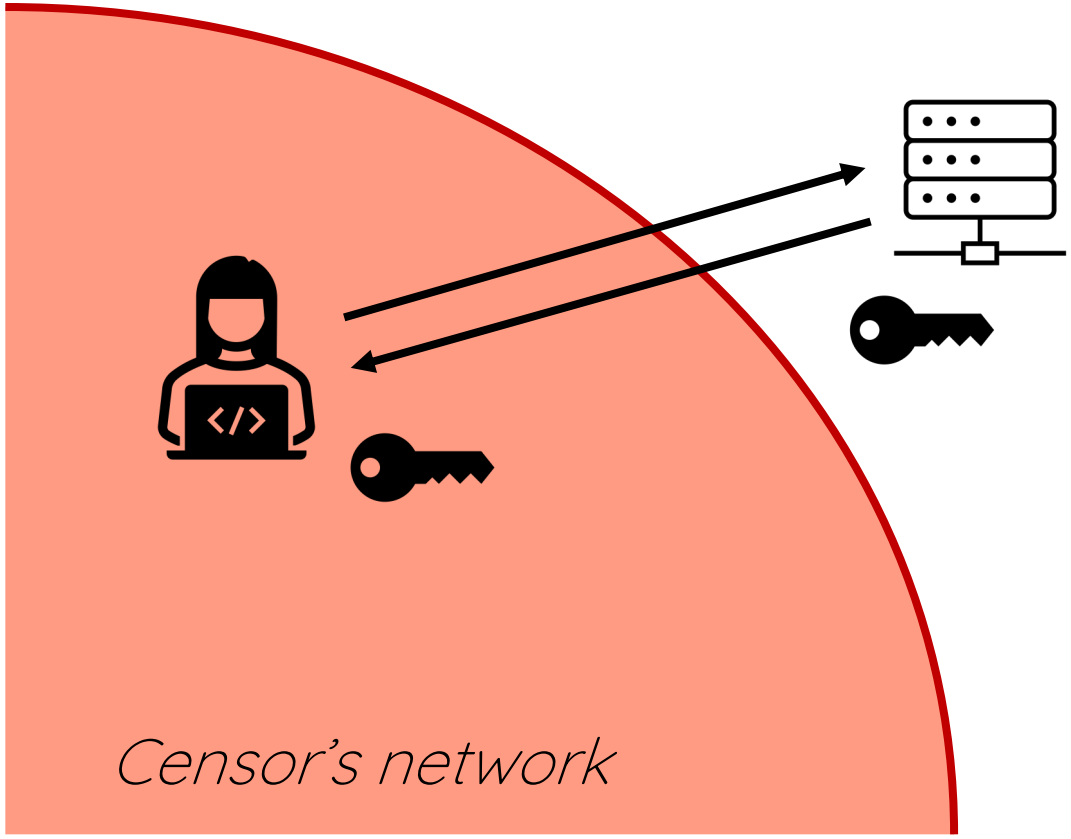
Anatomy of a Fully Encrypted Protocol

Requires: Key Exchange

- To obtain strong encryption keys

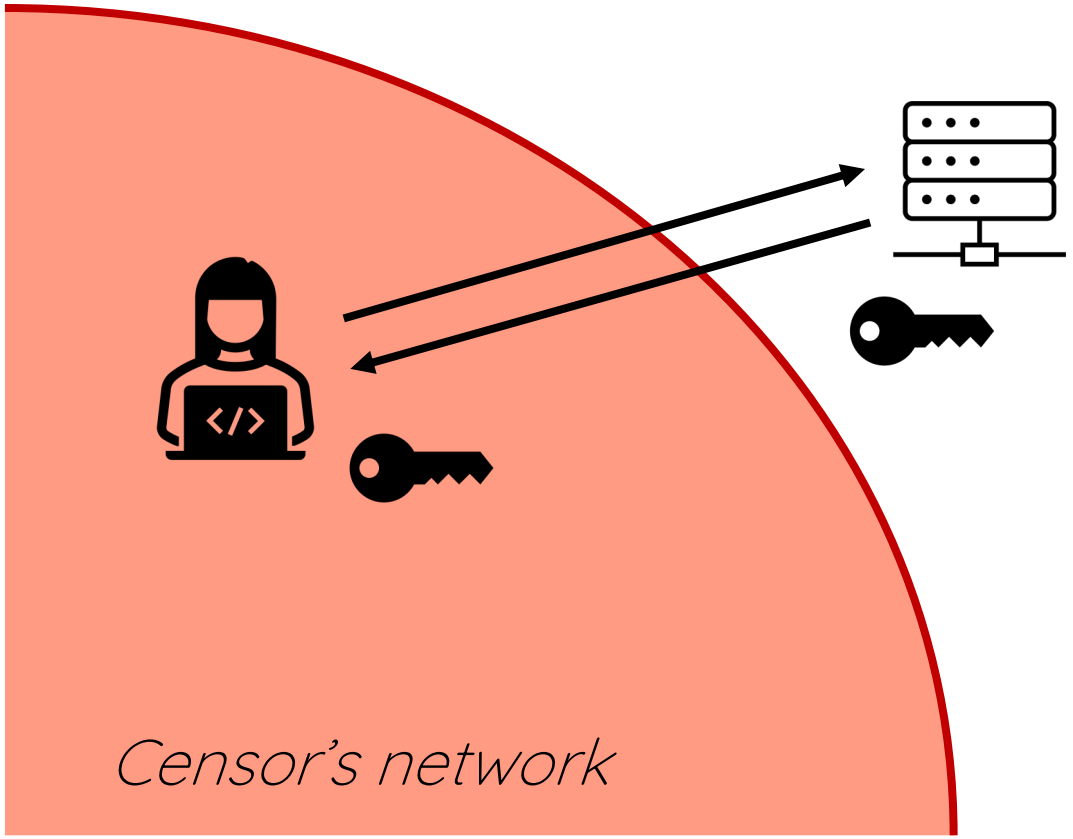
Data Transfer

- Transfer encrypted data in a look-like-nothing/ randomized way
- [FJ23] introduces formal model



[FJ23] Fenske and Johnson. Security Notions of Fully Encrypted Protocols. FOCI23

Desirable Key Exchange Properties



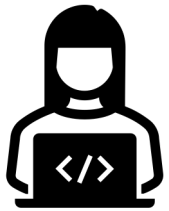
Desirable Properties:
- Good session keys
- Obfuscation
- ??

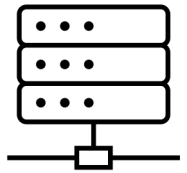
Case Study: obfsproxy

pre 2012: unobfuscated Tor

2012: obfs2

```
[ random seed ] [ content | padlength | ..... padding ..... ]
-- plaintext -- -- encrypted with key generated from seed ---
```





Design inspired by `obfuscated-openssh`.

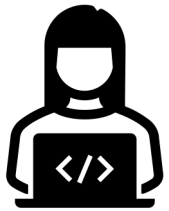
Case Study: obfsproxy

Desirable Properties:
- ~~Good session keys~~
- Obfuscation ?

pre 2012: unobfuscated Tor

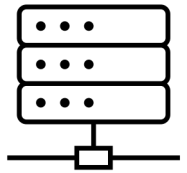
2012: obfs2

```
[ random seed ] [ content | padlength | ..... padding ..... ]
-- plaintext -- -- encrypted with key generated from seed ---
```

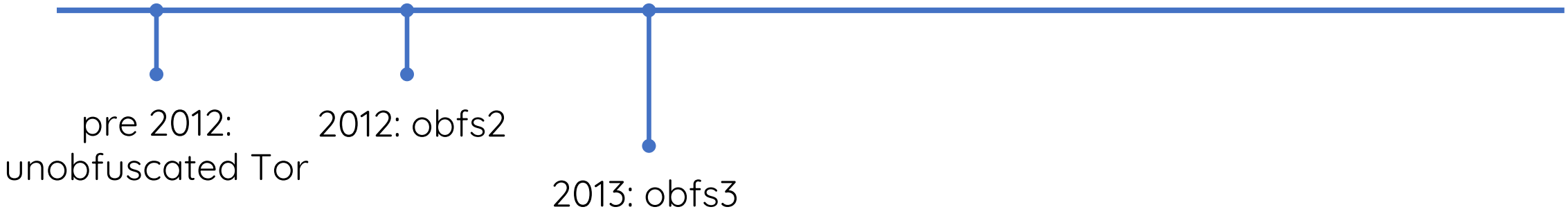


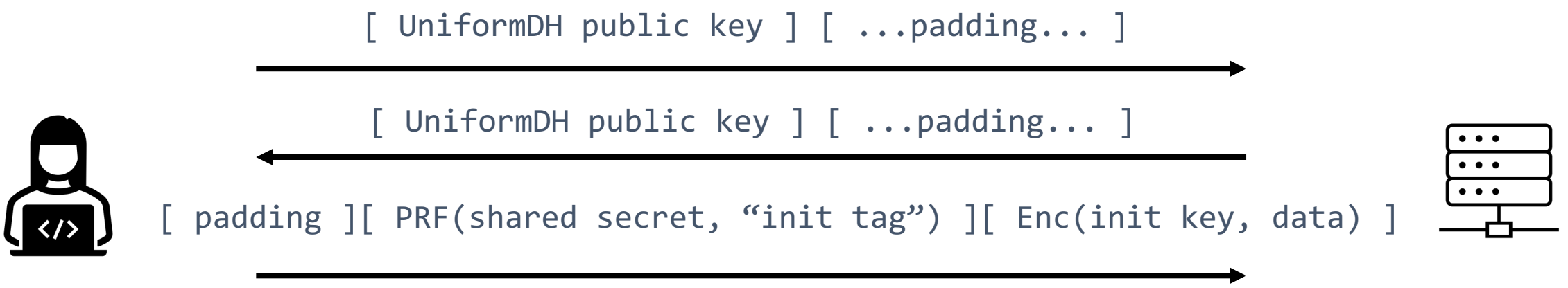
Problems:

- no “good” session keys
- obfuscation undermined when content is sent in data transfer phase



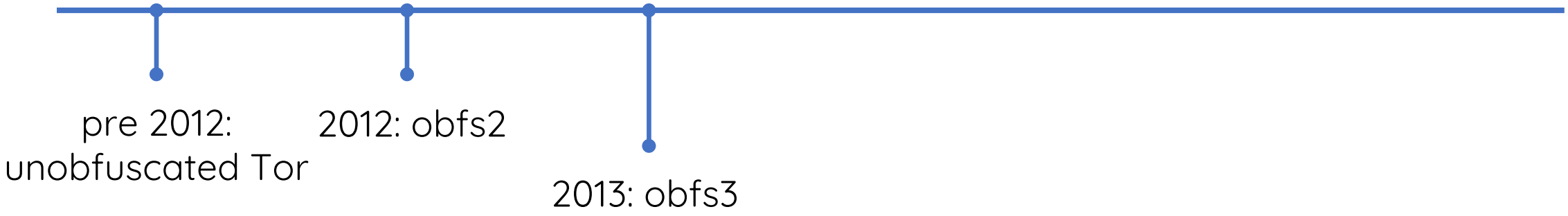
Case Study: obfsproxy

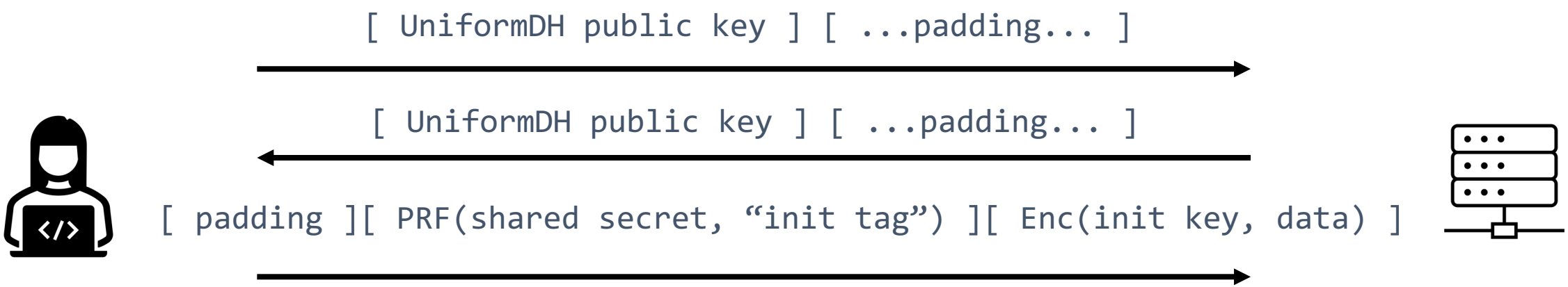




Case Study: obfsproxy

Desirable Properties:
- Good session keys
- Obfuscation
- ~~Authentication~~





Problem: no authentication of the server

Case Study: obfsproxy

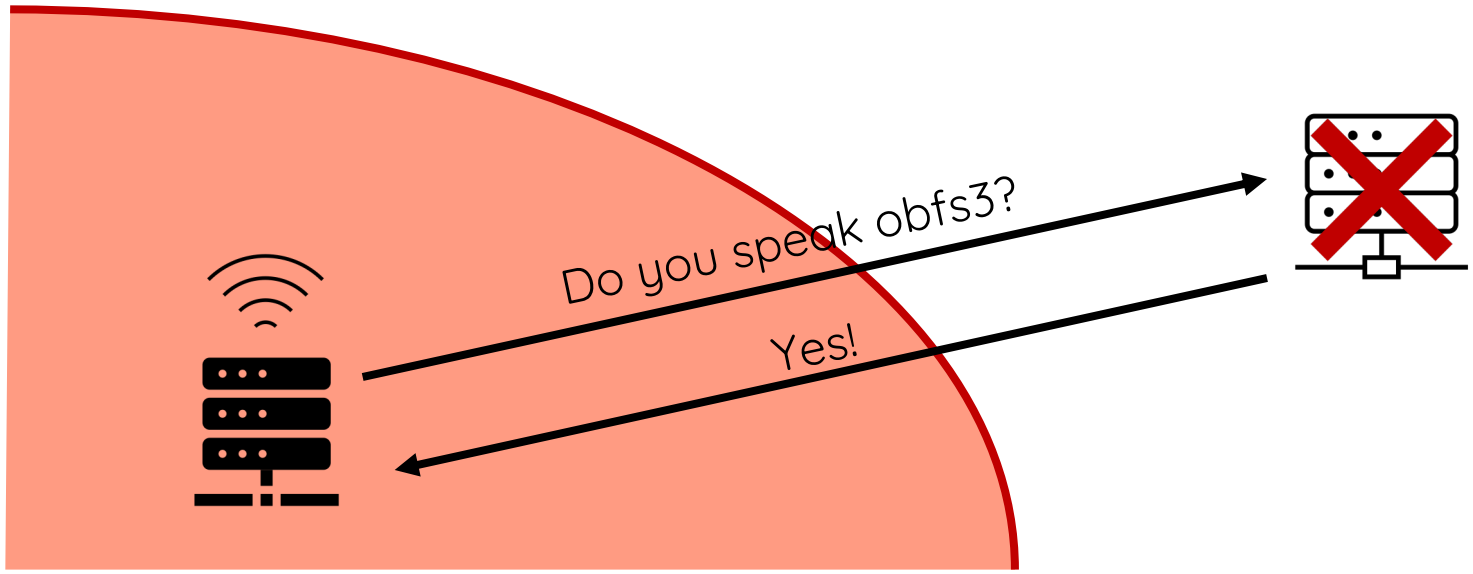
Desirable Properties:
- Good session keys
- Obfuscation
- Authentication
- Probing resistance

pre 2012: unobfuscated Tor

2012: obfs2

2013: obfs3

2014: evidence of **active probing** from the GFW



Ensafi, Fifield, Winter, Feamster, Weaver, Paxson. Examining How the Great Firewall Discovers Hidden Circumvention Servers. IMC 2015.

Case Study: obfsproxy

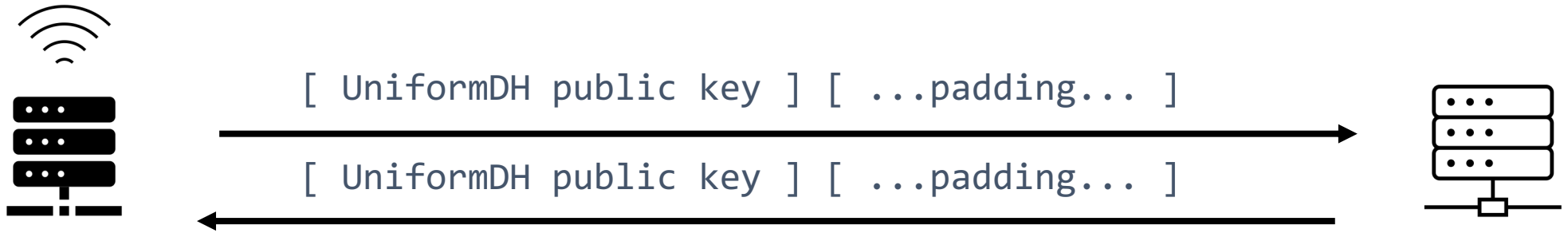
Desirable Properties:
- Good session keys
- Obfuscation
- ~~Authentication~~
- ~~Probing resistance~~

pre 2012: unobfuscated Tor

2012: obfs2

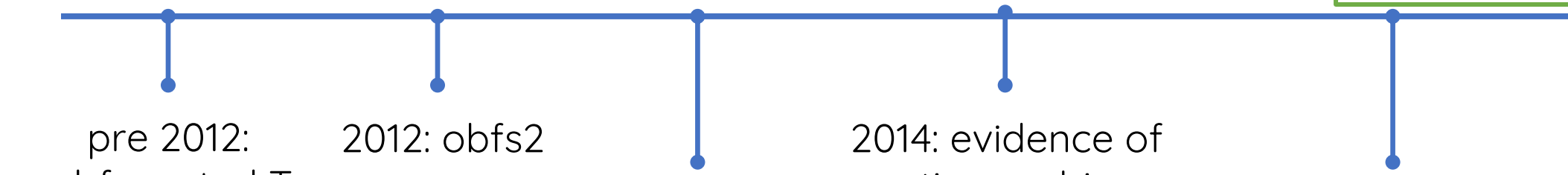
2013: obfs3

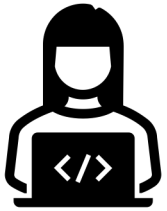
2014: evidence of **active probing** from the GFW



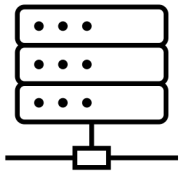
Case Study: obfsproxy

Desirable Properties:
- Good session keys
- Obfuscation
- Authentication
- Probing resistance





`[ Elligator pk ][ padding ][ PRF(B|NODEID, pk) ][ PRF(B|NODEID, ...) ]`



`[ Elligator pk ][ auth tag ][ padding ][ PRF(B|NODEID, pk) ][ PRF(B|NODEID, ...) ]`

Design inspired by **`ScrambleSuit`**. Encapsulates **ntor** handshake.

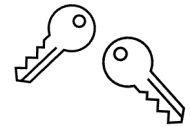
Winter, Pulls, Fuss. *ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship.* WPES 2013.

Goldberg, Stebila, Ustaoglu. *Anonymity and one-way authentication in key exchange protocols.* DCC 2012.

Modeling Obfuscated Key Exchange

- Development has followed an iterative design process
- We hope to move away from the cat-and-mouse style of development, and towards protocols grounded in formal analysis

Key exchange model:

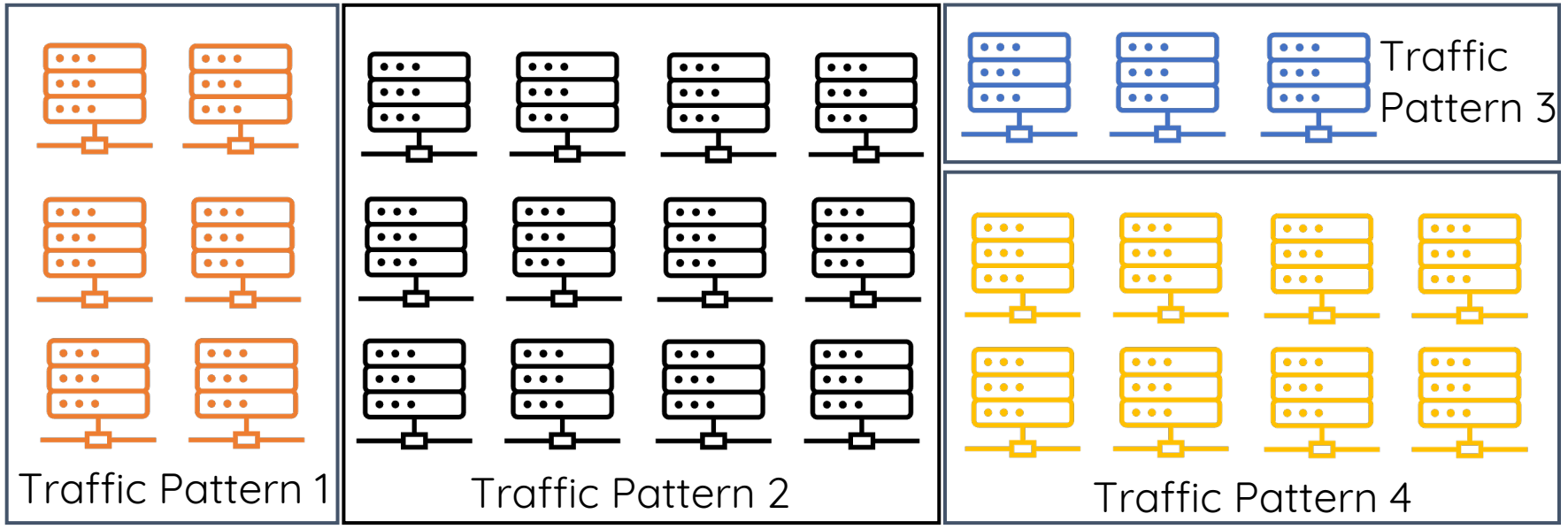


Desirable Properties:

- Key indistinguishability
- Obfuscation
- Explicit authentication
- Probing resistance

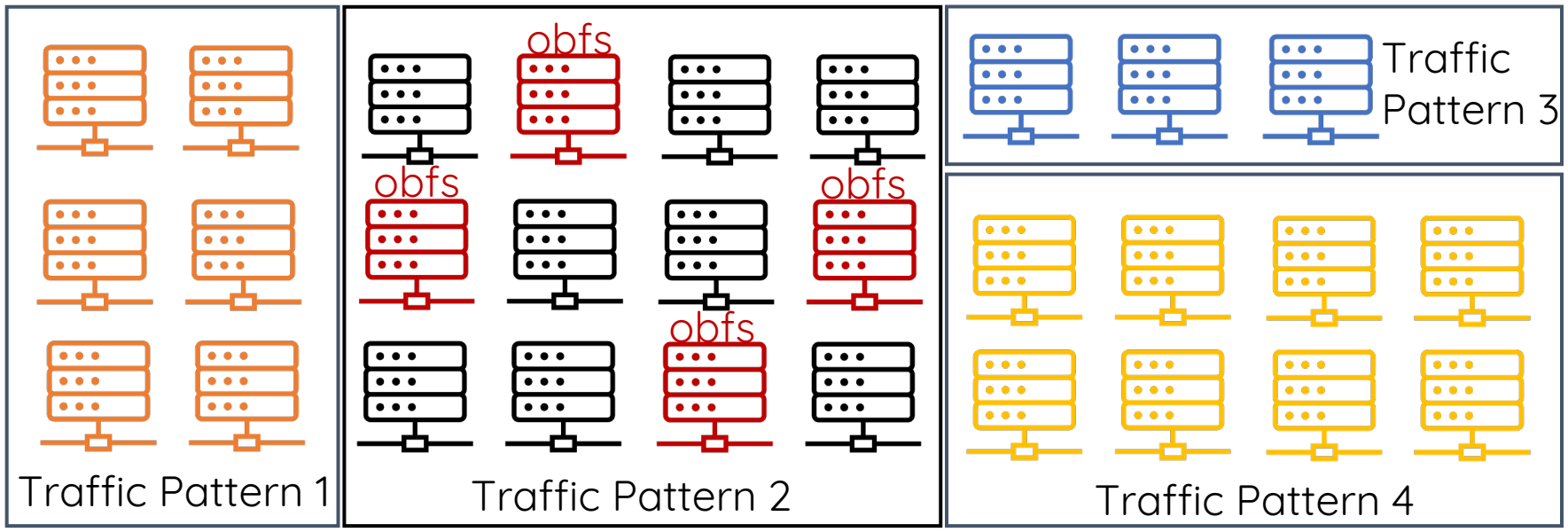
Modeling Obfuscated Key Exchange

- Protocols observed in the wild can be grouped into classes (capturing traffic patterns, entropy, etc.)



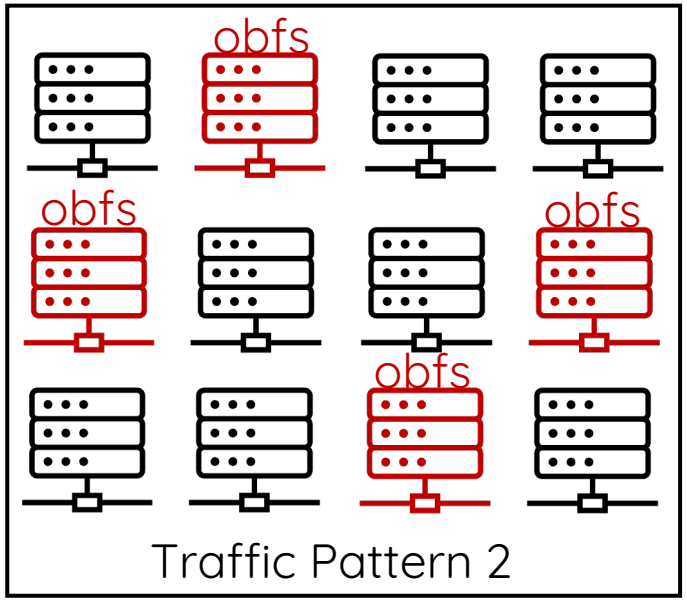
Modeling Obfuscated Key Exchange

- Protocols observed in the wild can be grouped into classes (capturing traffic patterns, entropy, etc.)

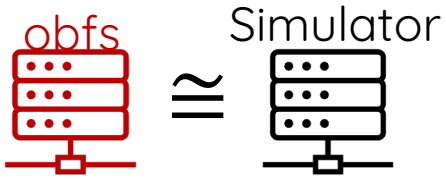


Anonymity set (obfs servers in red)

Modeling Obfuscated Key Exchange



Anonymity set (obfs servers in red)

Obfuscation:  ≅

We define obfuscation with respect to a *simulated* protocol, where the simulator determines the set of protocols we can hide within.

- If the simulator captures a large class of protocols, our anonymity set is large.
- The properties that this simulator should adhere to (i.e., ideal cover traffic) are yet to be completely determined.

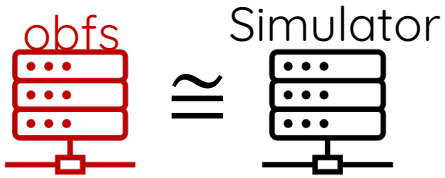
Summary of obfs4 Analysis

✓Good session keys
(key indistinguishability)

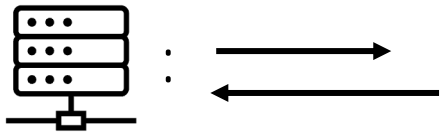
ntor base protocol analysis [GSU12]
+ public key (Elligator) encoding*

*introduces a small loss

✓Obfuscation



Elligator + random oracle



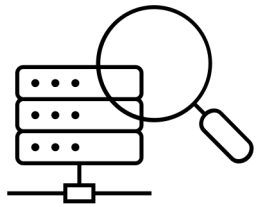
✓(Explicit) Authentication

✓Probing resistance

[GSU12] Goldberg, Stebila, Ustaoglu. *Anonymity and one-way authentication in key exchange protocols*. DCC 2012.

Extending Obfuscated Key Exchange

Setting: What happens if bridge information is revealed?



Adversary obtains:

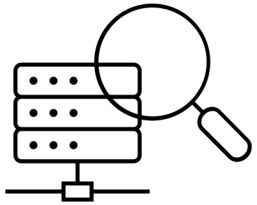
`B, NODEID`

The adversary can identify all (past and future) obfs4 traffic to this server.

```
[ Elligator pk ][ padding ][ PRF(B|NODEID, pk) ][ PRF(B|NODEID, ...) ]
```

Extending Obfuscated Key Exchange

New Property:
+ Strong obfuscation



An adversary that knows `B, NODEID` cannot identify obfs4 traffic.

(Like “forward security” for obfuscation with respect to public keys)

Previous approach:

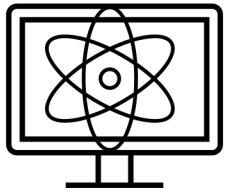
```
[ Elligator pk ][ padding ][ PRF(B|NODEID, pk) ][ PRF(B|NODEID, ...) ]
```

New construction (simplified):

```
[ Elligator pk ][ padding ][ PRF(B^x, B|NODEID|pk) ][ PRF(B^x, B|NODEID|...) ]
```

Extending Obfuscated Key Exchange

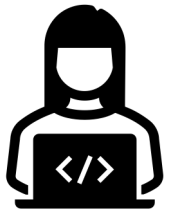
Setting: What happens in the presence of a quantum computer?



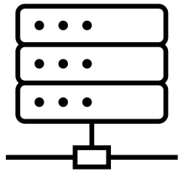
Existing construction based on elliptic curve cryptography is no longer secure.

New Property:
+ Post-quantum security

Post-Quantum Obfuscated Key Exchange



sk_S^{KEM}, pk_S^{KEM}



$(pk_e, sk_e) \leftarrow KEM.KGen()$
$(c_S, K_S) \leftarrow KEM.Encap(pk_S^{KEM})$

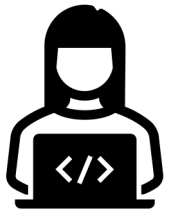
$[\ pk_e|c_S\ ][\ padding\ ][\ PRF(K_S, pk_e|c_s)\ ][\ PRF(K_S, pk_e|c_s|...)\ ]$

$K_S \leftarrow KEM.Decap(sk_S^{KEM}, c_S)$
$(c_e, K_e) \leftarrow KEM.Encap(pk_e)$

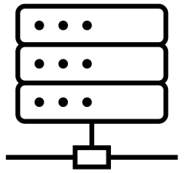
Key derived from K_S and K_e (and NODEID).

$[\ c_e\ ][\ auth\ tag\ ][\ padding\ ][\ PRF(K_S, c_e)\ ][\ PRF(K_S, c_e|...)\ ]$

Post-Quantum Obfuscated Key Exchange



sk_S^{KEM}, pk_S^{KEM}



$(pk_e, sk_e) \leftarrow KEM.KGen()$
$(c_S, K_S) \leftarrow KEM.Encap(pk_S^{KEM})$

$[\ pk_e^* | c_S^*\ ][\ padding\ ][\ PRF(K_S, pk_e|c_s)\ ][\ PRF(K_S, pk_e|c_s|...)\ ]$

$K_S \leftarrow KEM.Decap(sk_S^{KEM}, c_S)$
$(c_e, K_e) \leftarrow KEM.Encap(pk_e)$

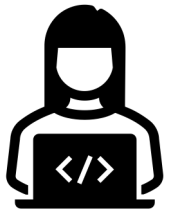
Problem: pk_e, c_S, c_e distinguishable from random

Solution: use encoded pk_e^*, c_S^*, c_e^* that look random

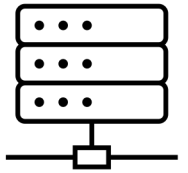
$[\ c_e^*\ ][\ auth\ tag\ ][\ padding\ ][\ PRF(K_S, c_e)\ ][\ PRF(K_S, c_e|...)\ ]$

Post-Quantum Obfuscated Key Exchange*

*Simplified



sk_S^{KEM}, pk_S^{KEM}



$(pk_e, sk_e) \leftarrow KEM.KGen()$
$(c_S, K_S) \leftarrow KEM.Encap(pk_S^{KEM})$

$[\ pk_e^* | c_S^*\ ][\ padding\ ][\ PRF(K_S, pk_e | c_s)\ ][\ PRF(K_S, pk_e | c_s | ...)\ ]$

$K_S \leftarrow KEM.Decap(sk_S^{KEM}, c_S)$
$(c_e, K_e) \leftarrow KEM.Encap(pk_e)$

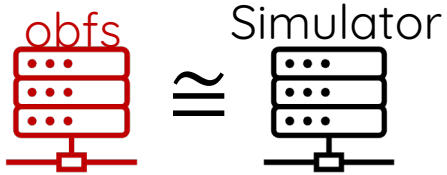
$[\ c_e^*\ ][\ auth\ tag\ ][\ padding\ ][\ PRF(K_S, c_e)\ ][\ PRF(K_S, c_e | ...)\ ]$

Post-Quantum Obfuscated Key Exchange*

*Analysis in progress

✓Good session keys (key indistinguishability)

✓(Strong) Obfuscation



✓(Explicit) Authentication

✓Probing resistance

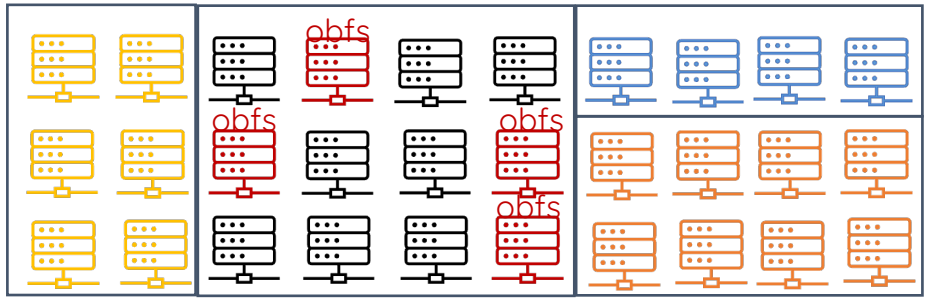
Requirements & Assumptions

+ KEM: IND-CCA
+ KEM anonymity properties (SPR-CCA [MX22])
+ Public key (to random) encoding
+ Ciphertext (to random) encoding
+ Dual PRF security

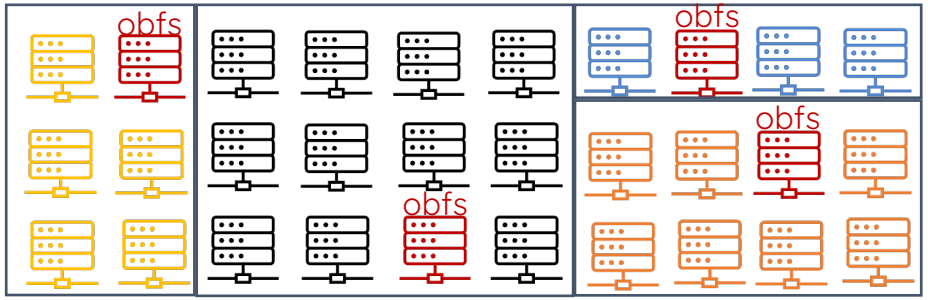
Construction:
ML-KEM, HMAC, and (new) ML-KEM-specific encodings

Summary and Challenges

Goal: evade detection by behaviour.



Current approach to obfuscation: all instances of the protocol lie within the same anonymity class.



Alternative approach: each instance of the protocol lies within a different anonymity class.

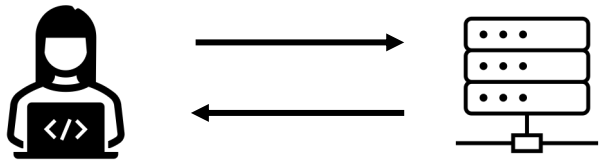
Intuition: identifying one instance tells you nothing about other instances.

Summary and Challenges

Goal: evade detection by behaviour.

Desirable Properties:
- \- Good session keys
- \- Obfuscation
- \- Authentication
- \- Probing resistance
- \+ Strong obfuscation
- \+ Post-quantum security

\+ Timing sequences



The obfs4 handshake pattern has an identifiable “down” period after the client’s first message.

Modeling this requires a streaming-like notion for key exchange.

Summary and Challenges

Goal: evade detection by behaviour.

Desirable Properties:
- \- Good session keys
- \- Obfuscation
- \- Authentication
- \- Probing resistance
- \+ Strong obfuscation
- \+ Post-quantum security

- \+ Timing sequences
- \+ …

Thank you!

Obfuscated Key Exchange

Felix Günther, Douglas Stebila, **Shannon Veitch**