MultiFlow: Cross-Connection Decoy Routing using TLS 1.3 Session Resumption

Victoria Manfredi and Pi Songkuntham

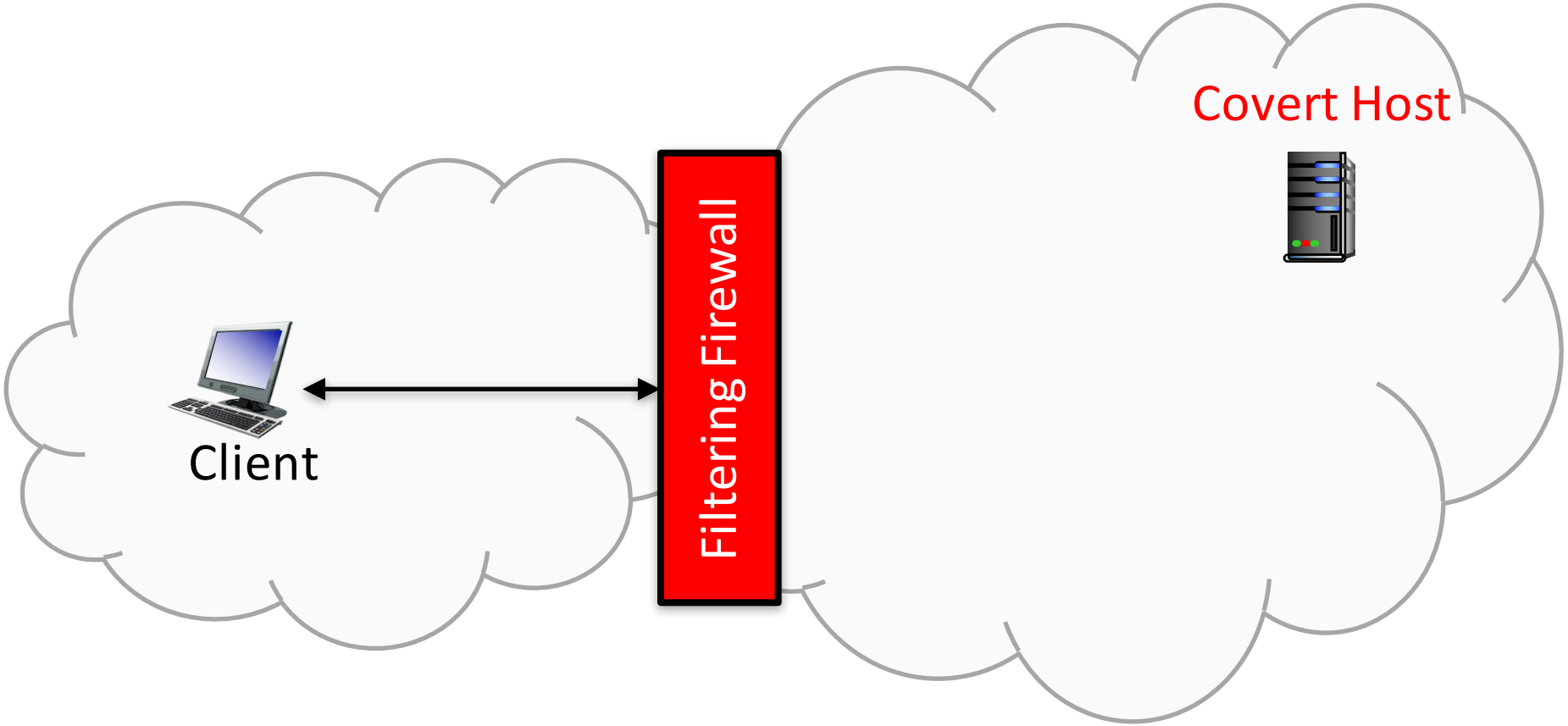
WESLEYAN UNIVERSITY

FOCI

August 14, 2018

Motivation

On the contemporary Internet, many sites are blocked or filtered

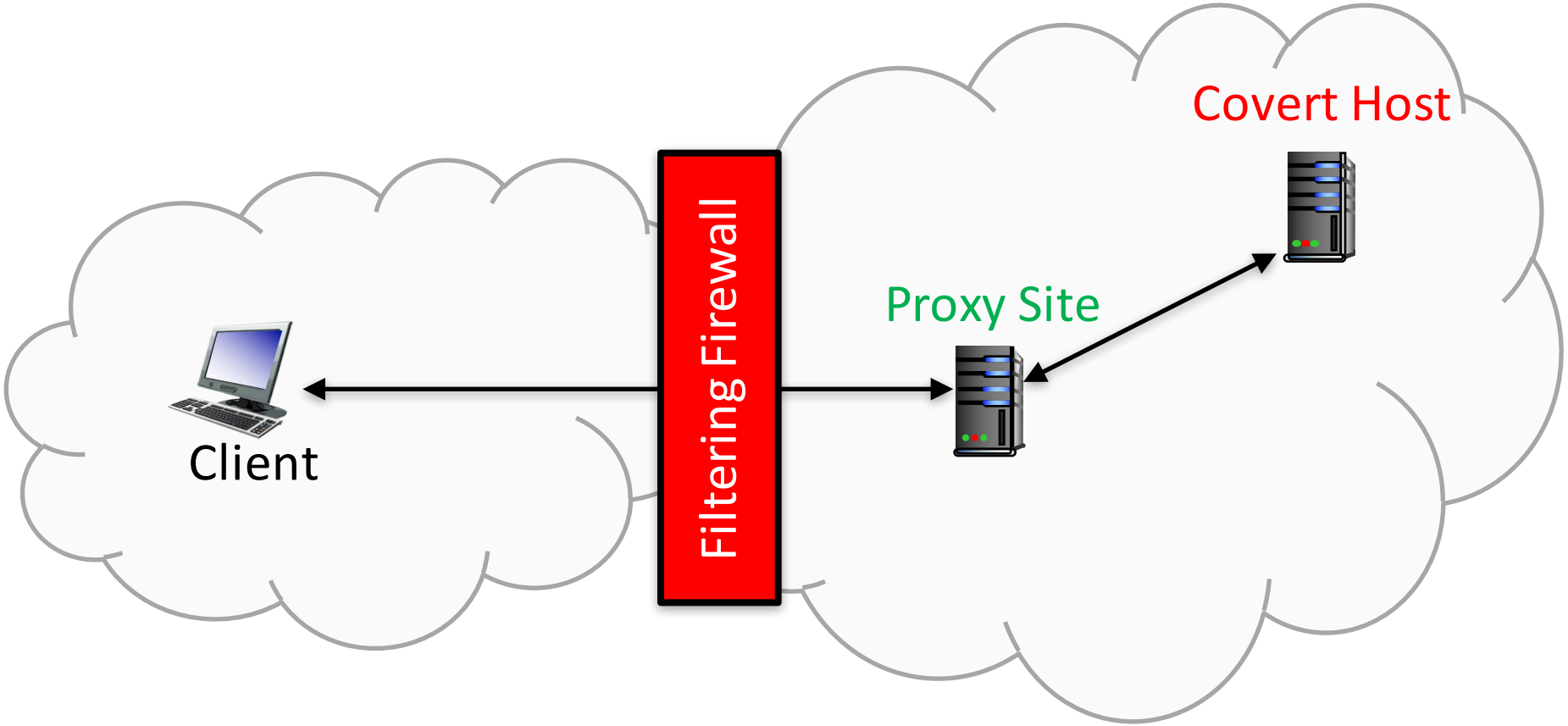


Adversary Network

Outside

Current solutions

Some users use proxies or VPNs to bypass filters

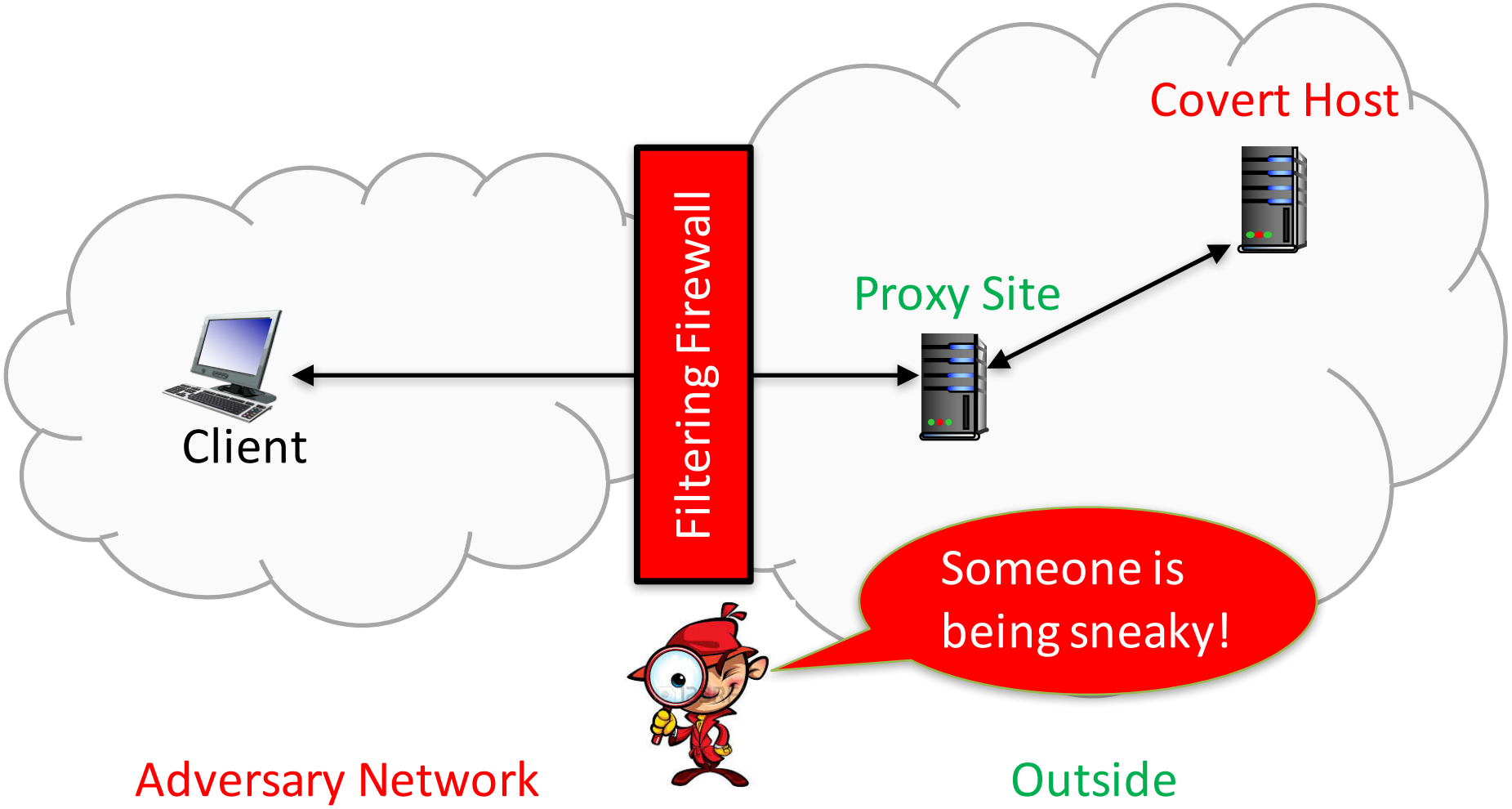


Adversary Network

Outside

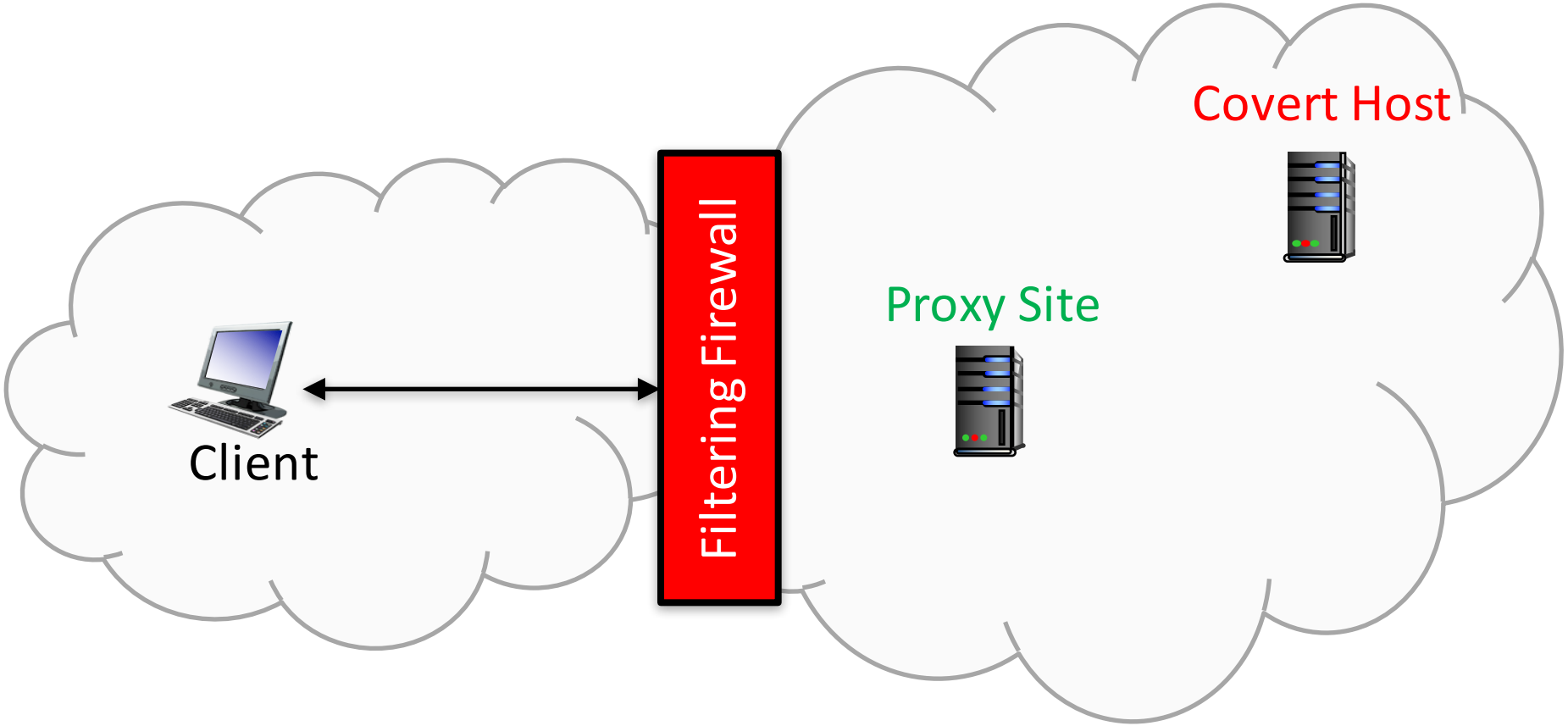
Problems with current solutions

Using a proxy can attract unwanted attention



Problems with current solutions

… and firewalls are getting better at blocking proxies



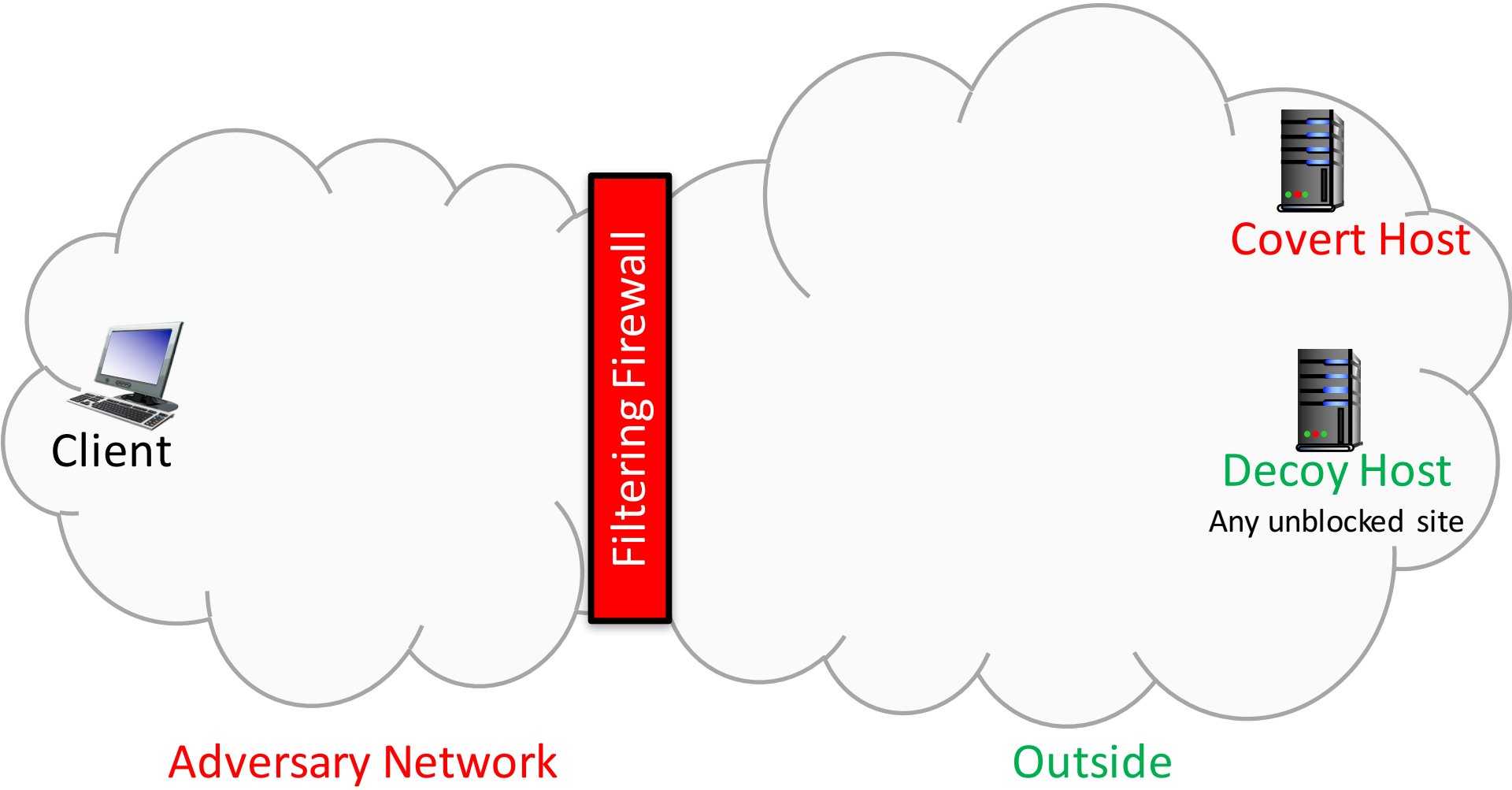
Adversary Network

Outside

An alternate approach: decoy routing

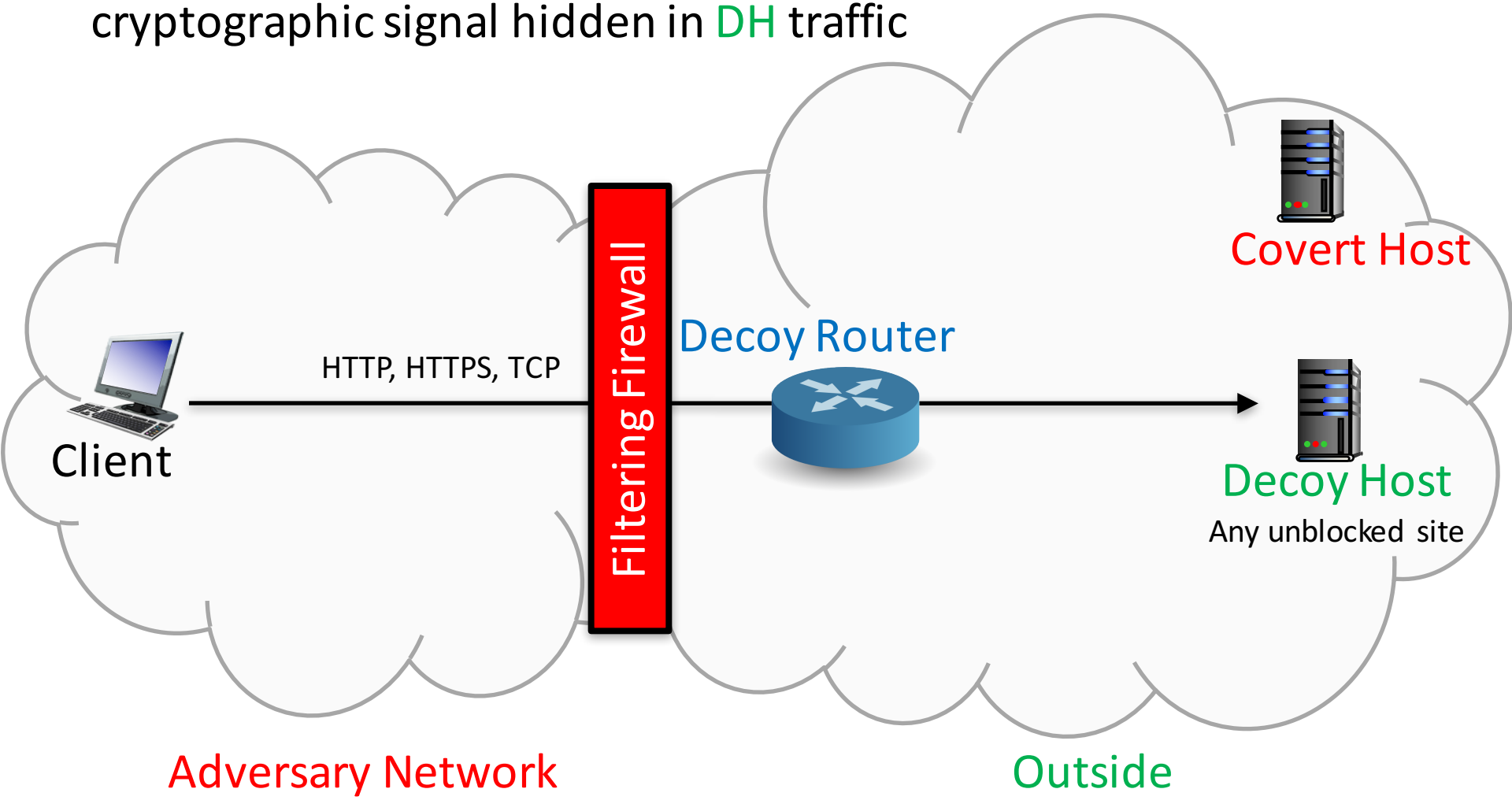
Decoy routing intuition

Relies on (1) routers being much harder to block than hosts, and (2) existence of sites too (un)important to block



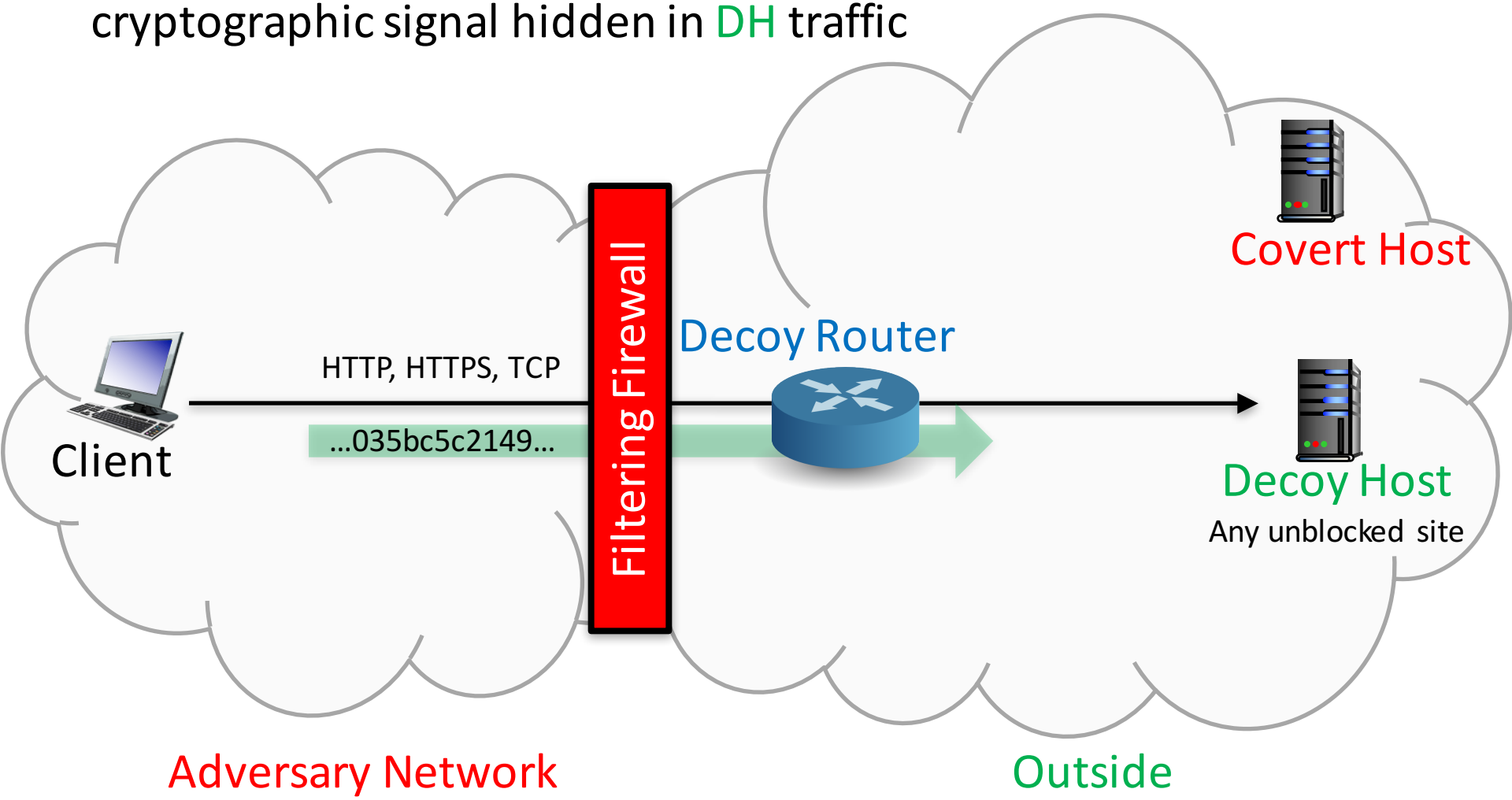
Implementation

Handshake: Client hides cryptographic signal in traffic sent to allowed host, DH. DR detects and responds to Client with its own cryptographic signal hidden in DH traffic



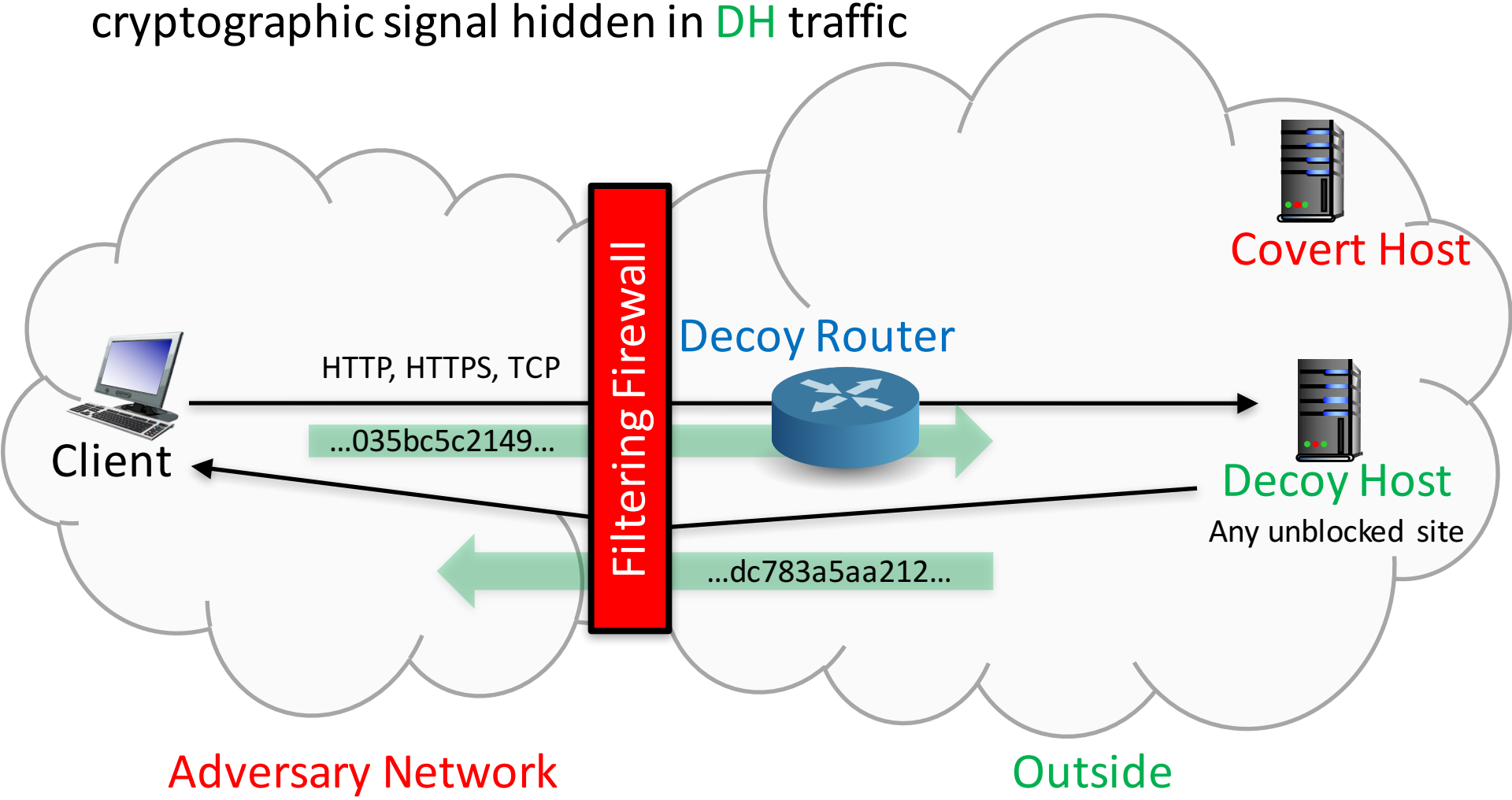
Implementation

Handshake: Client hides cryptographic signal in traffic sent to allowed host, DH. DR detects and responds to Client with its own cryptographic signal hidden in DH traffic



Implementation

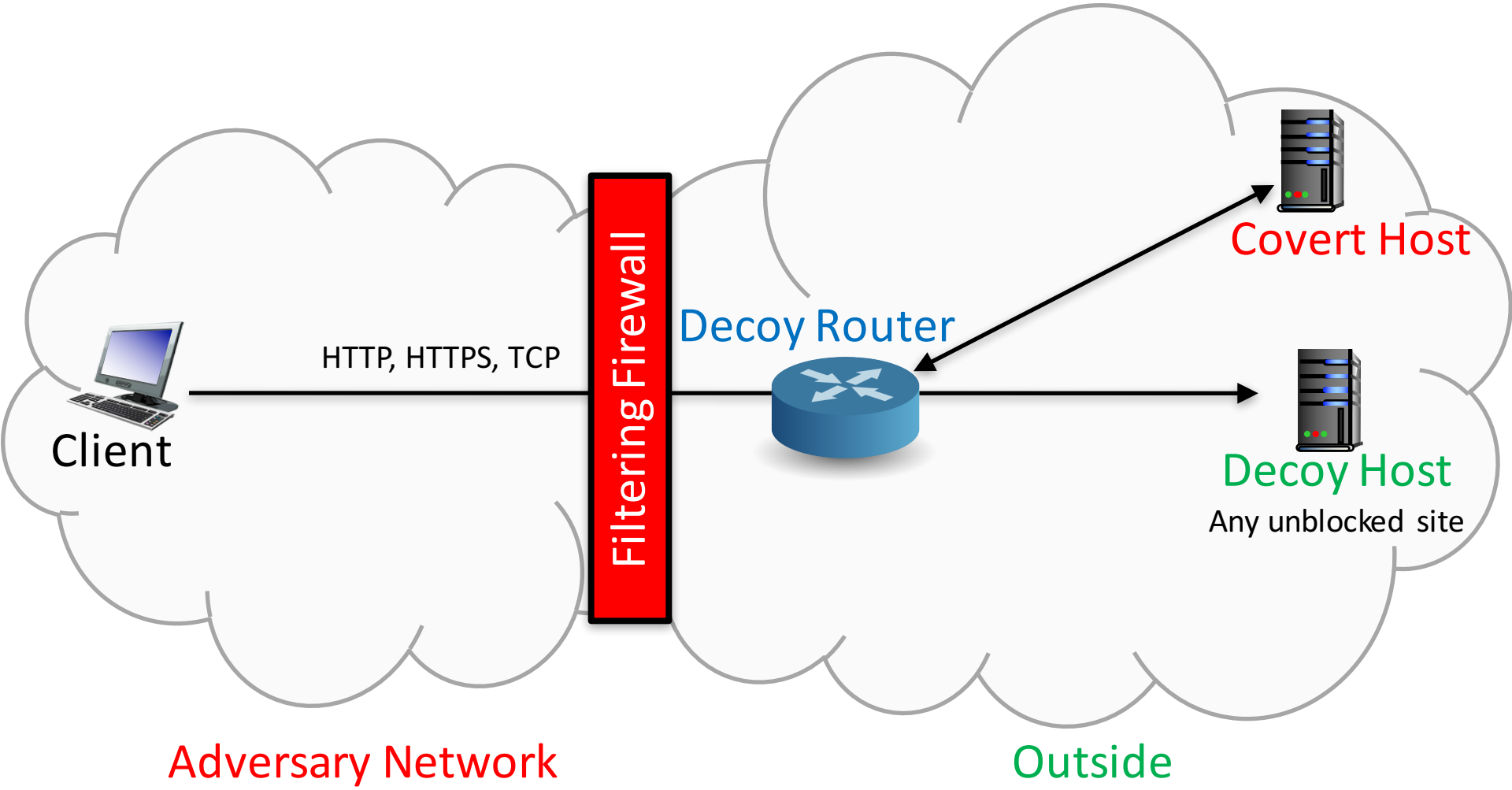
Handshake: Client hides cryptographic signal in traffic sent to allowed host, DH. DR detects and responds to Client with its own cryptographic signal hidden in DH traffic



Implementation

Tunnel: Client can now securely connect via DR to any IP address.

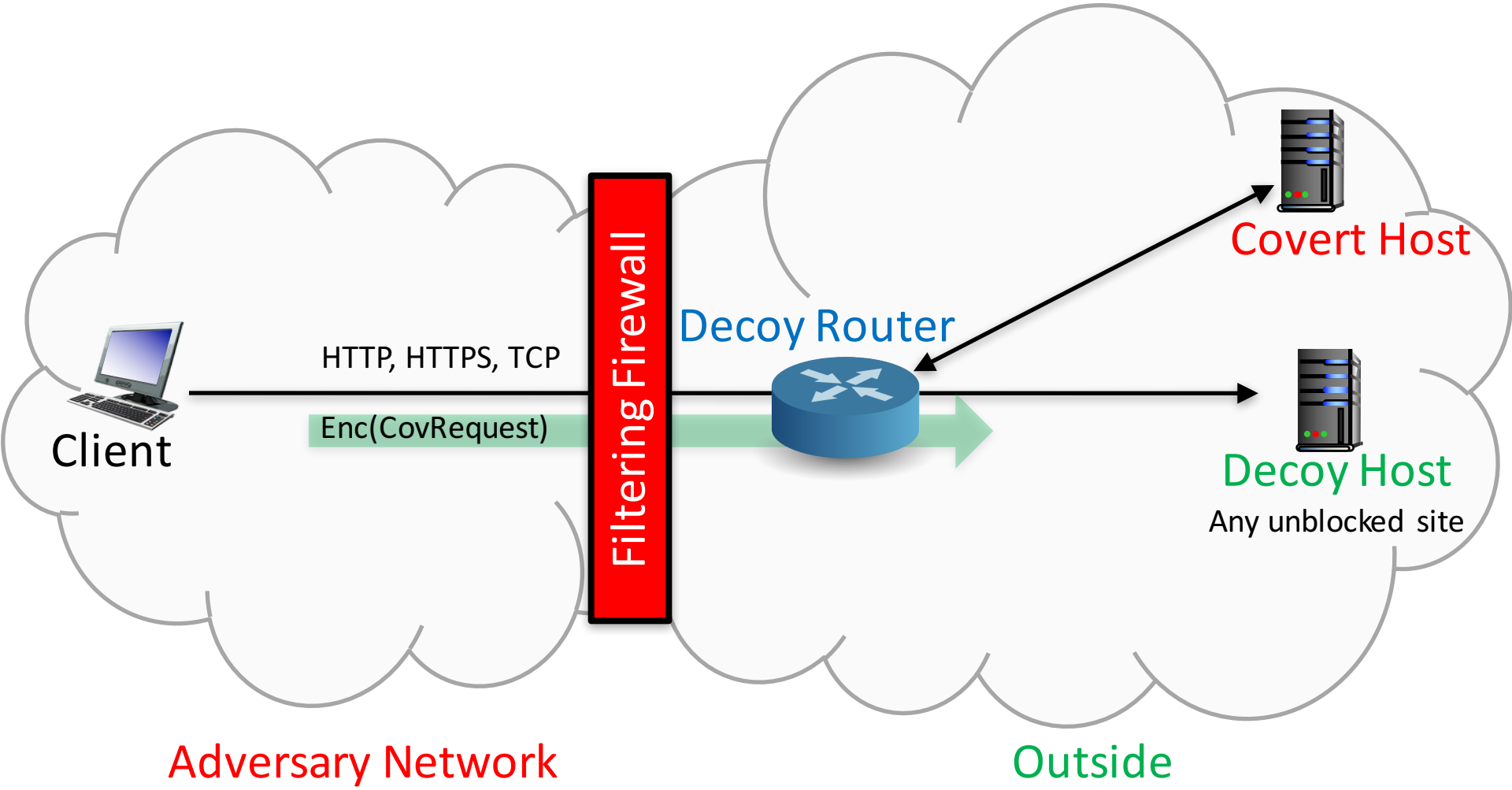
Client ⇔ DR hide their communication within Client ⇔ DH traffic



Implementation

Tunnel: Client can now securely connect via DR to any IP address.

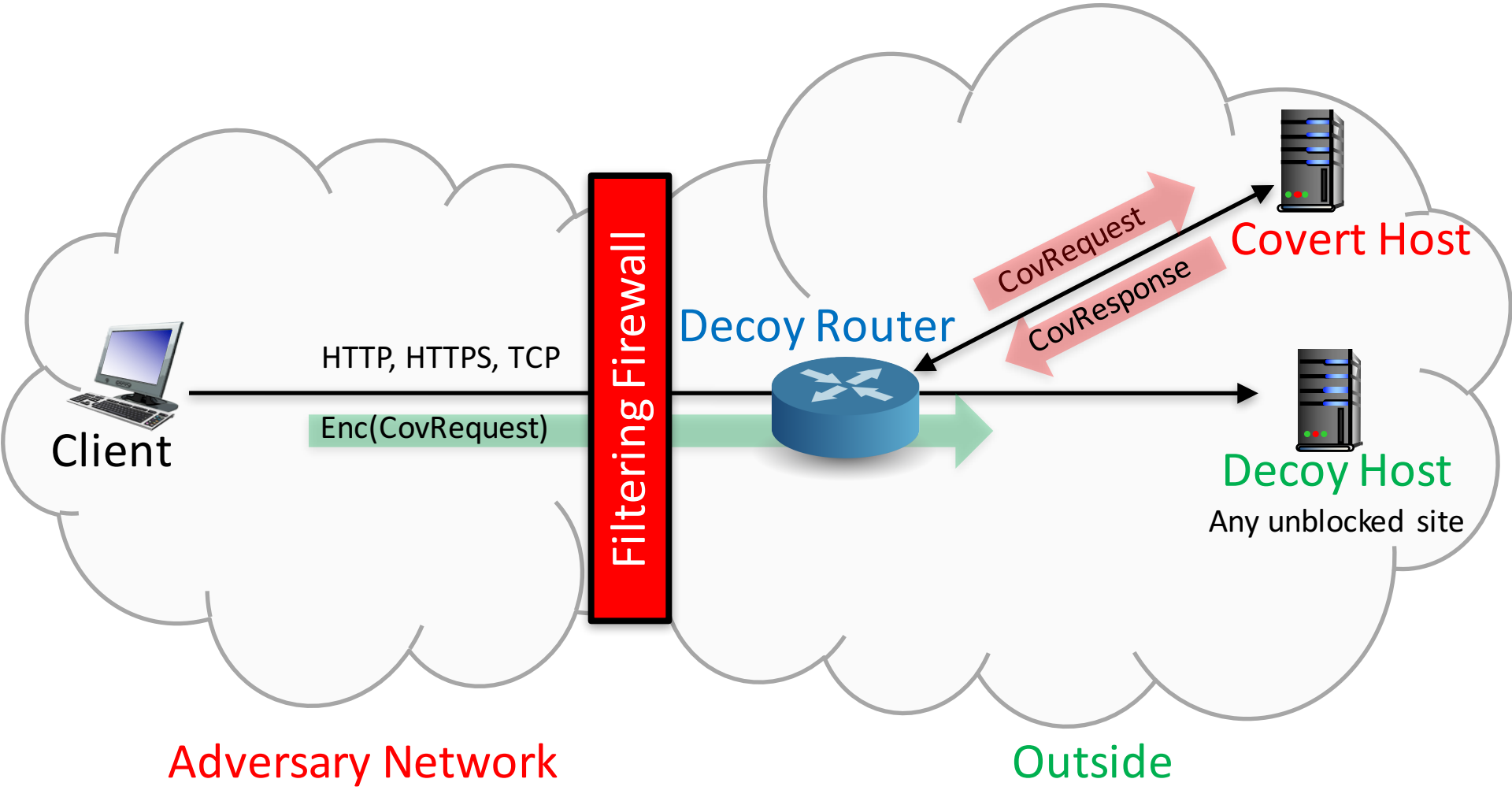
Client ⇔ DR hide their communication within Client ⇔ DH traffic



Implementation

Tunnel: Client can now securely connect via DR to any IP address.

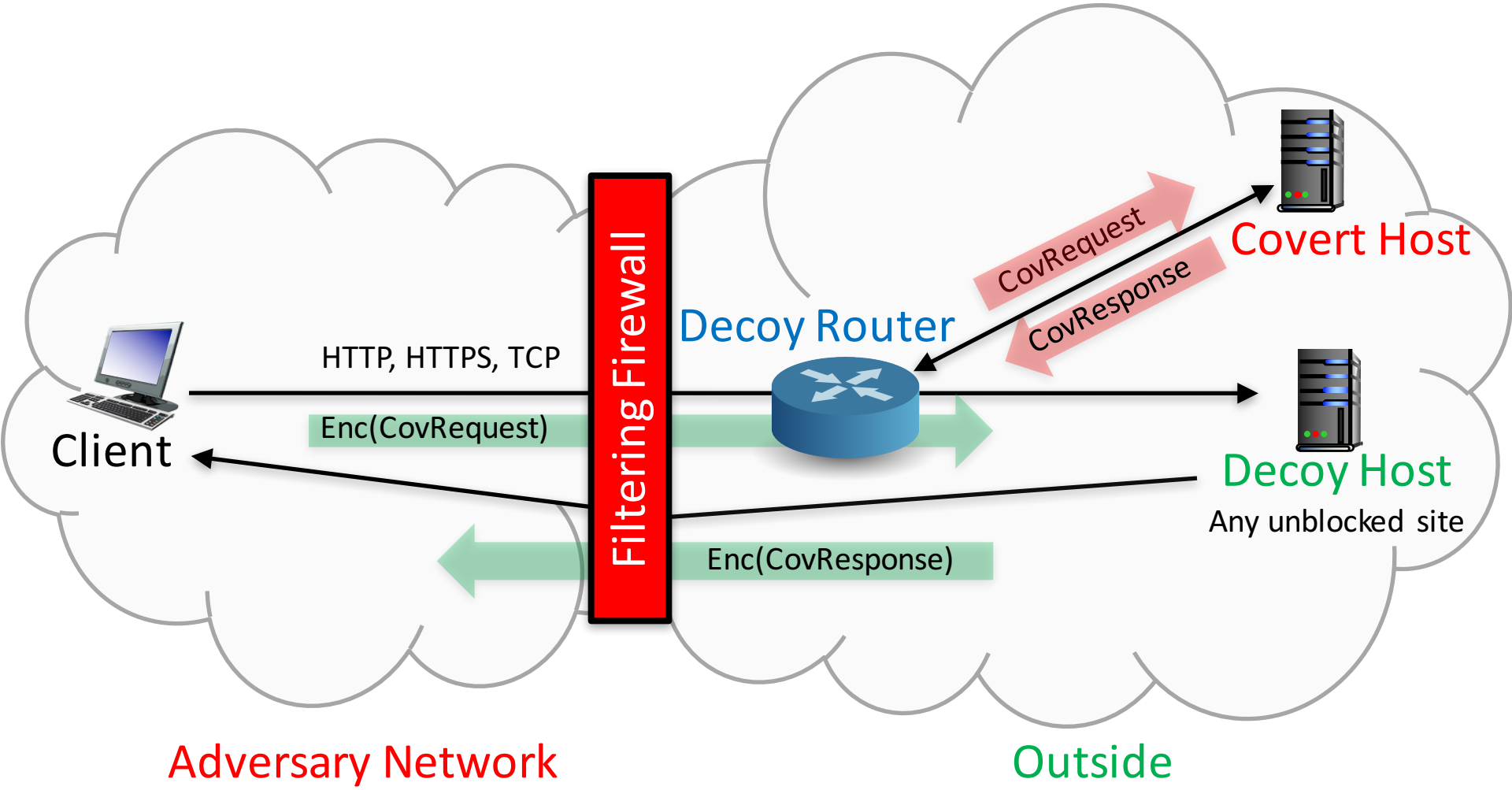
Client ⇔ DR hide their communication within Client ⇔ DH traffic



Implementation

Tunnel: Client can now securely connect via DR to any IP address.

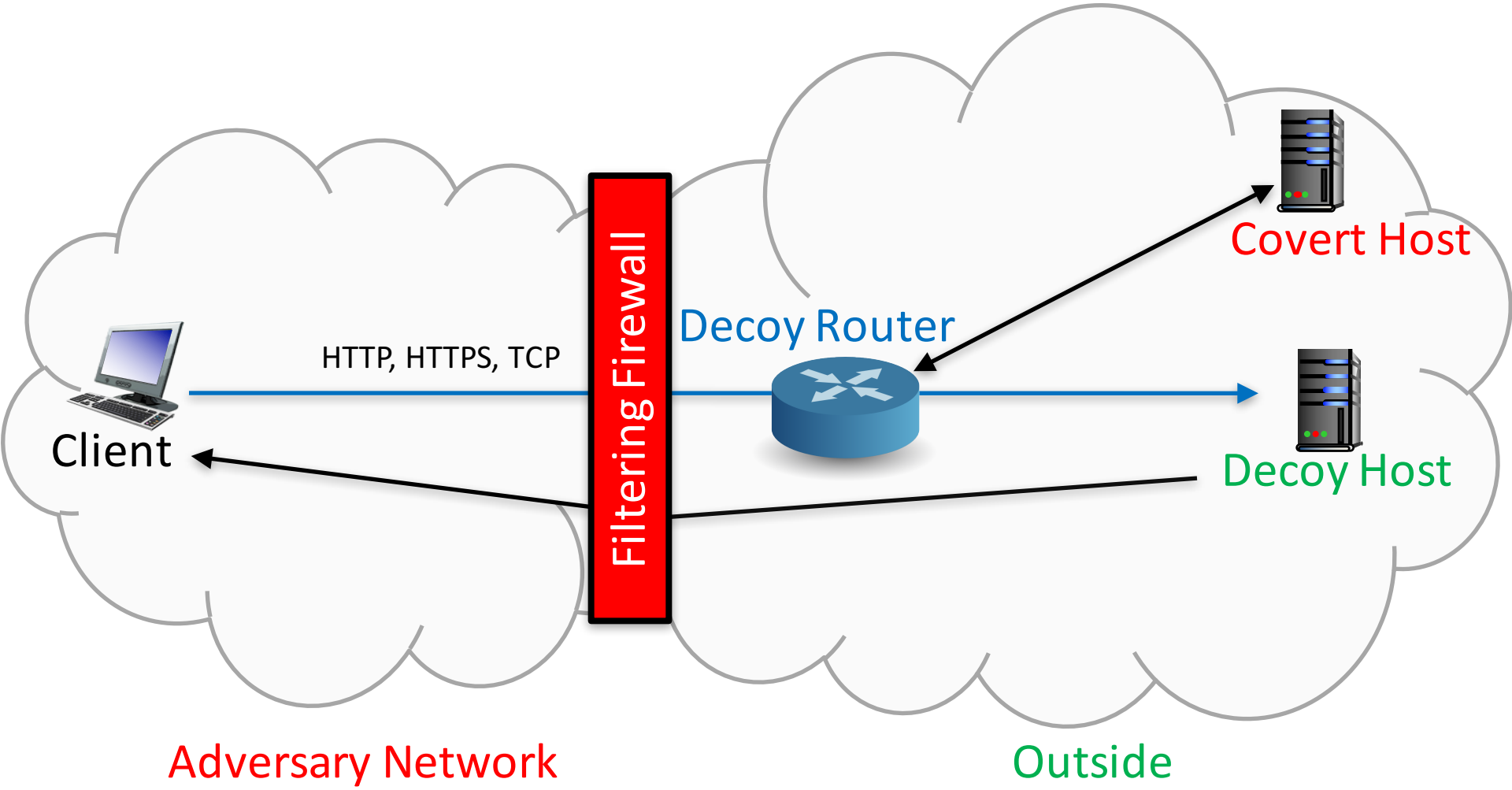
Client ⇔ DR hide their communication within Client ⇔ DH traffic



Decoy routing assumptions

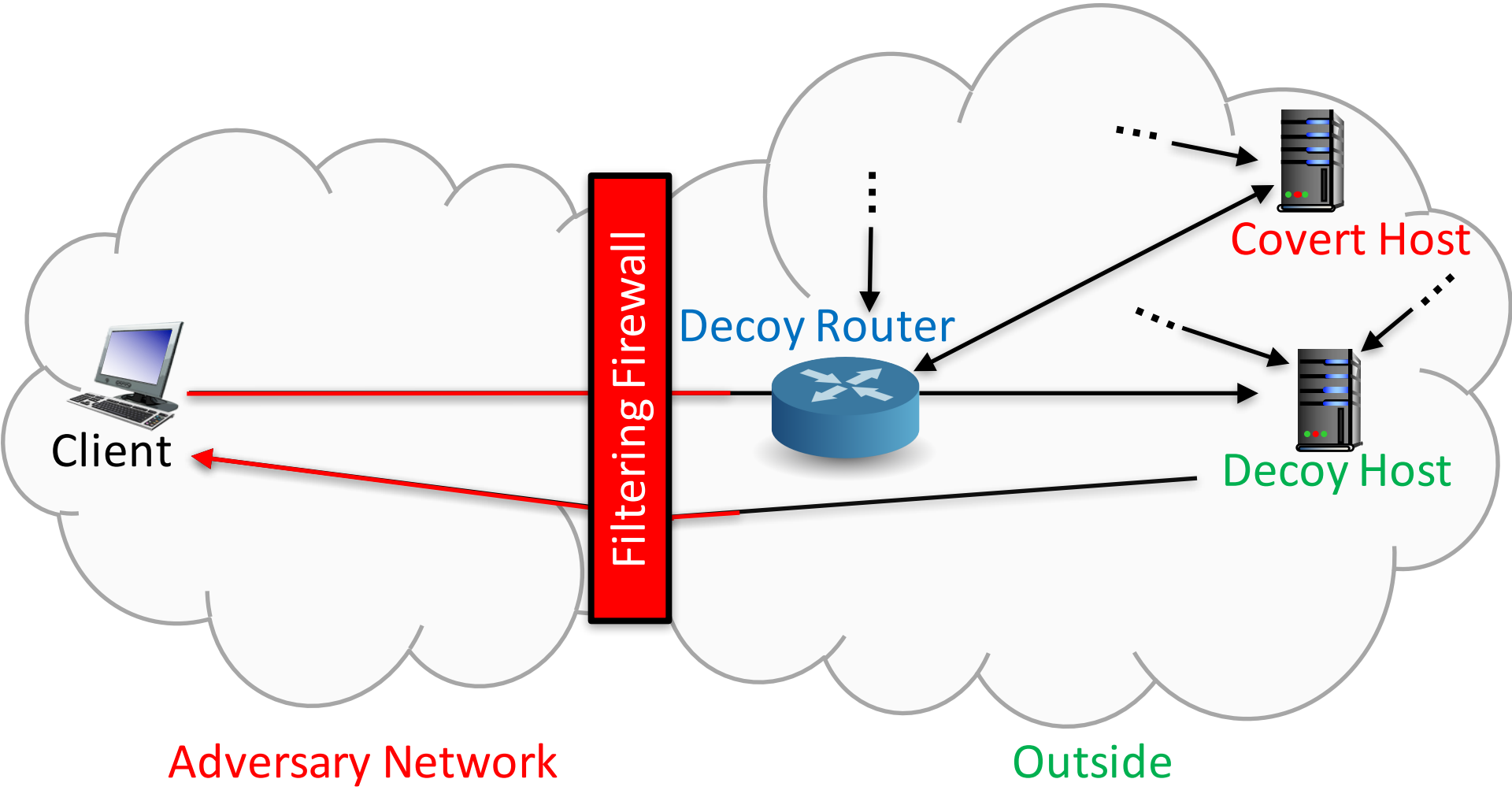
Assumptions

Route asymmetry: DR sees only one direction of Client-DH traffic



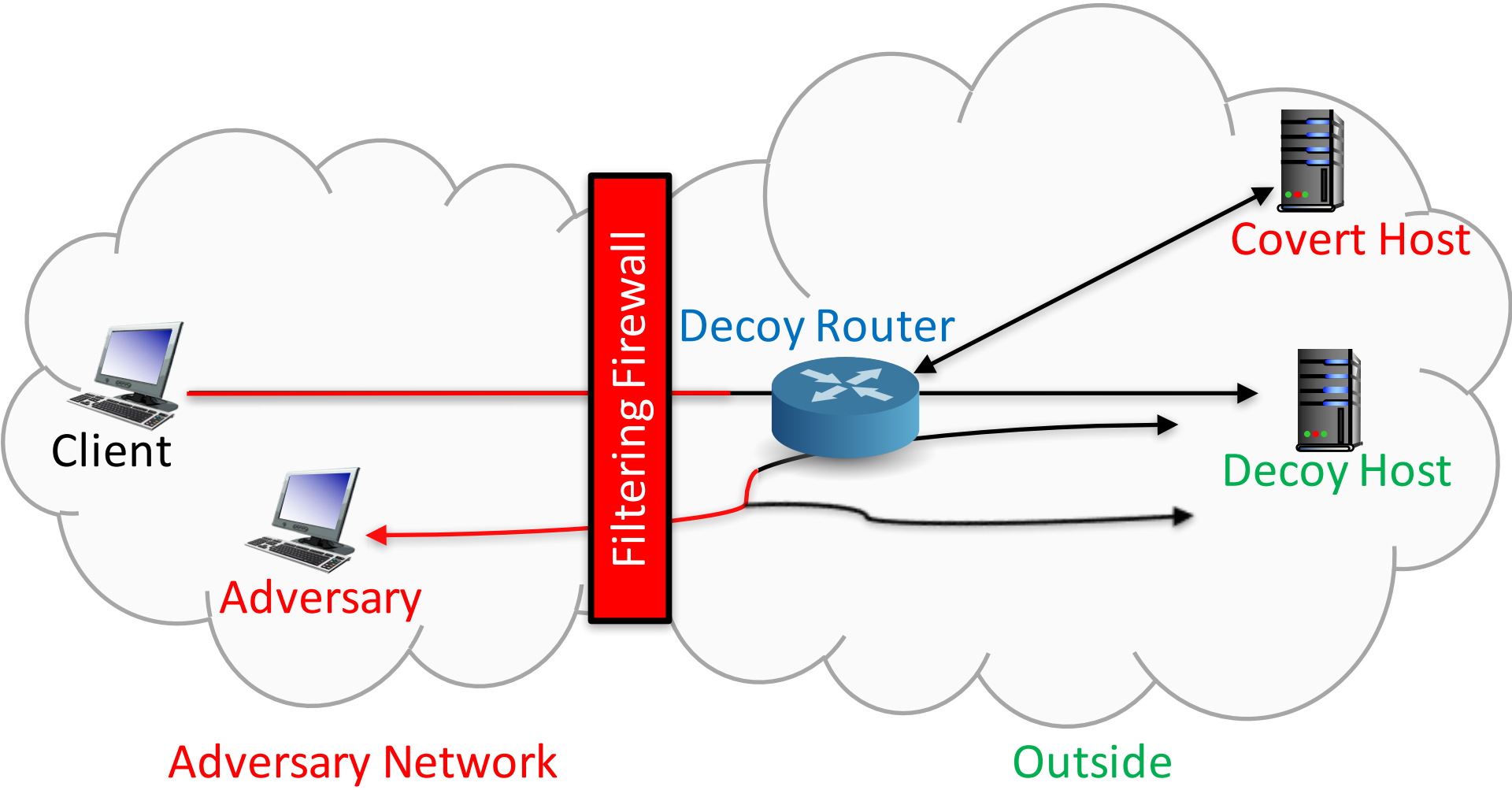
Assumptions

Adversary: can monitor, modify, block traffic to/from its network, but cannot see all traffic to/from DR, DH, CH



Assumptions

Active attacks: Adversary can replay client traffic and probe Decoy Host, but cannot monitor internal activity of Client



Are we done? Not quite …

Some problems

1. Inline blocking of traffic at DR is problematic for ISPs

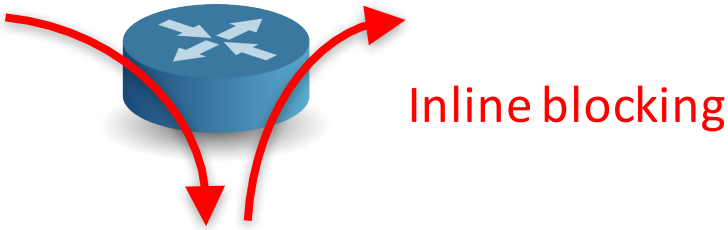
2. DR forging/rewriting traffic provides attack surface

3. TLS termination attack affects only decoy routing connections

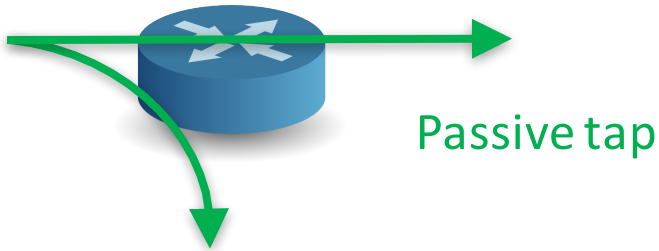
4. Rise of TLS 1.3

Some problems

1. Inline blocking of traffic at DR is problematic for ISPs



2. DR forging/rewriting traffic provides attack surface



3. TLS termination attack affects only decoy routing connections

TapDance [Wustrow et al, 2014]: passive tap but doesn't detect replay

How to thwart replay attack with passive tap and asymmetric routes?

4. Rise of TLS 1.3

Some problems

1. Inline blocking of traffic at DR is problematic for ISPs

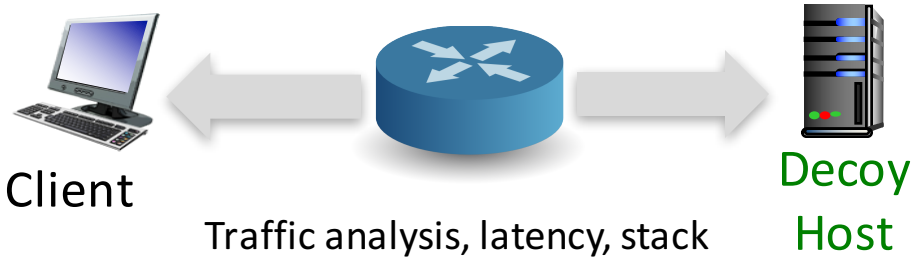
2. DR forging/rewriting traffic provides attack surface

3. TLS termination attack affects only decoy routing connections

4. Rise of TLS 1.3

Forge: Cirripede [Houmansadr et al, 2011]
Telex [Wustrow et al, 2011]
Curveball [Karlin et al, 2011]
TapDance [Wustrow et al, 2014]

Rewrite: Rebound [Ellard et al, 2015]
Slitheen [Bocovich and Goldberg, 2016]
Waterfall [Nasr et al, 2017]



Traffic analysis, latency, stack fingerprinting, probing attacks …

How to eliminate forging/rewriting by DR even with asymmetric routes?

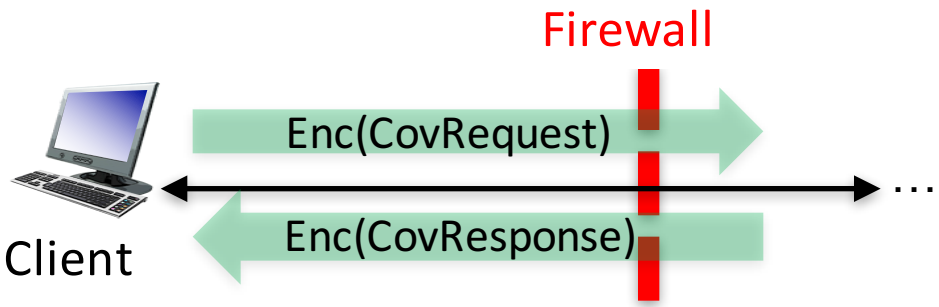
Some problems

1. Inline blocking of traffic at DR is problematic for ISPs

2. DR forging/rewriting traffic provides attack surface

3. TLS termination attack affects only decoy routing connections

4. Rise of TLS 1.3



*What if adversary terminates connection **<u>after</u>** data exchanged on normal TLS connection?*

Some problems

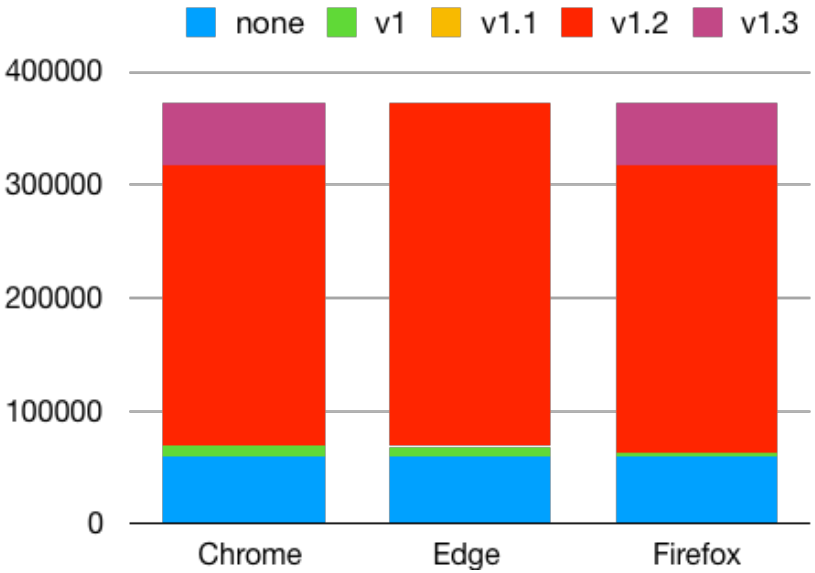
1. Inline blocking of traffic at DR is problematic for ISPs

2. DR forging/rewriting traffic provides attack surface

3. TLS termination attack affects only decoy routing connections

4. Rise of TLS 1.3

March 2018: IETF approves TLS 1.3



Acknowledgements: data collected by Daniel Ellard

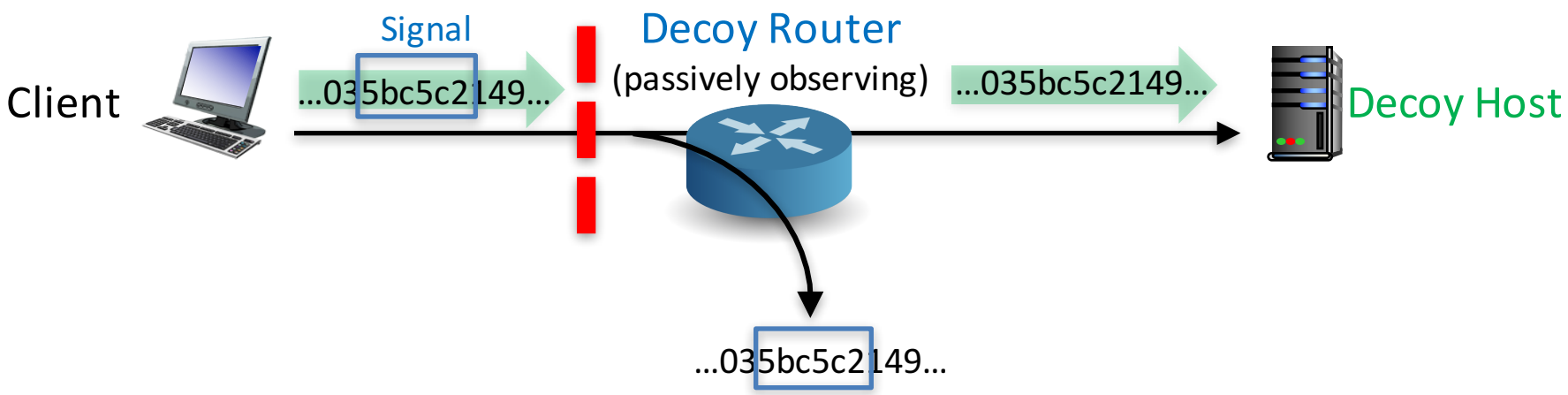
Browser support of TLS 1.3

MultiFlow: another decoy routing protocol

MultiFlow protocol

Step 1: Client signals to Decoy Router by setting TLS ClientRandom to appropriate string

Handshake

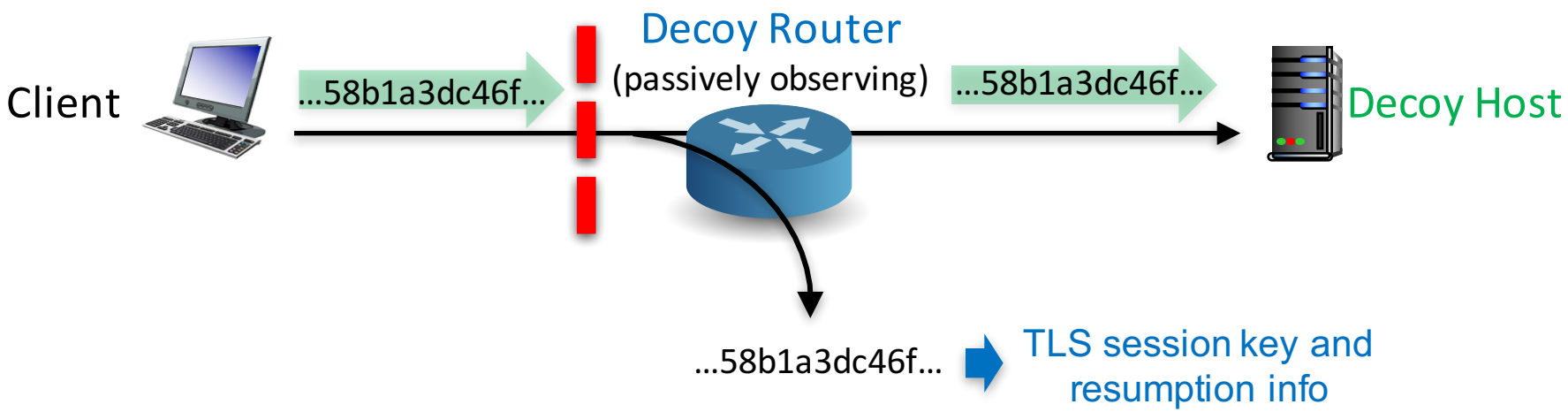


Benefit: Lets Decoy Router authenticate Client as valid (but not necessarily live) user

MultiFlow protocol

Step 2: Client exfiltrates its TLS session key and resumption info to Decoy Router

Handshake

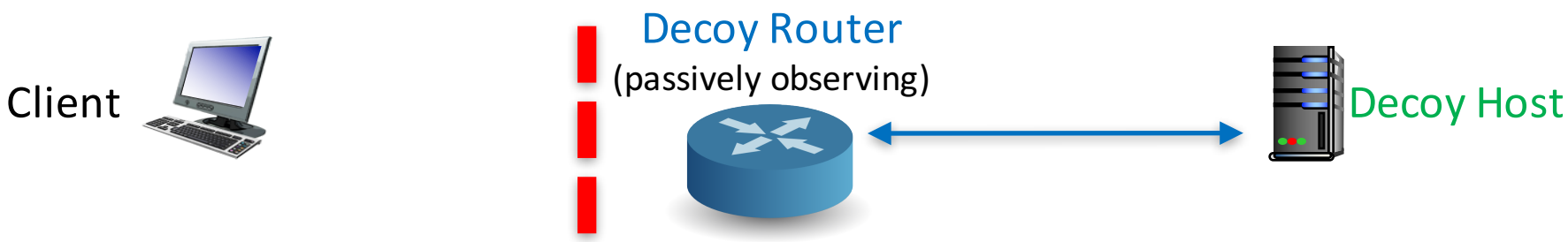


Benefit: Decoy Router can fully impersonate Client with Decoy Host

MultiFlow protocol

Step 3: Decoy Router resumes Client's session, opening new connection to Decoy Host

Handshake



Benefit: If resumable, then there exists a live Client even if adversary is replaying Client traffic

MultiFlow protocol

Step 3:
Handshake

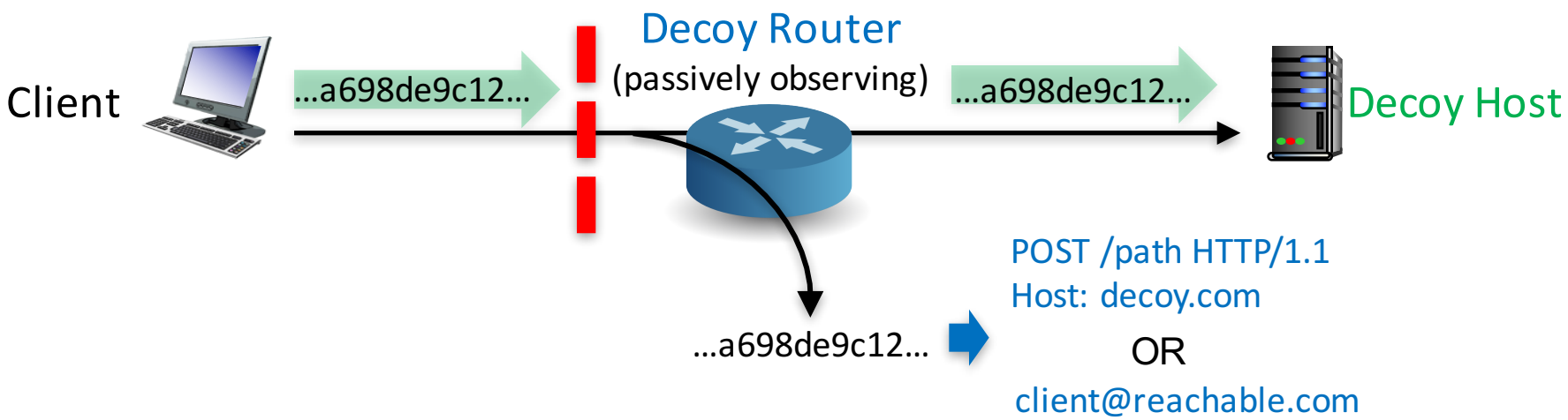
Decoy Router resumes Clients session, opening new connection to Decoy Host

Handshake is complete!

Benefit: If resumable, then there exists a live Client even if adversary is replaying Client traffic

MultiFlow protocol

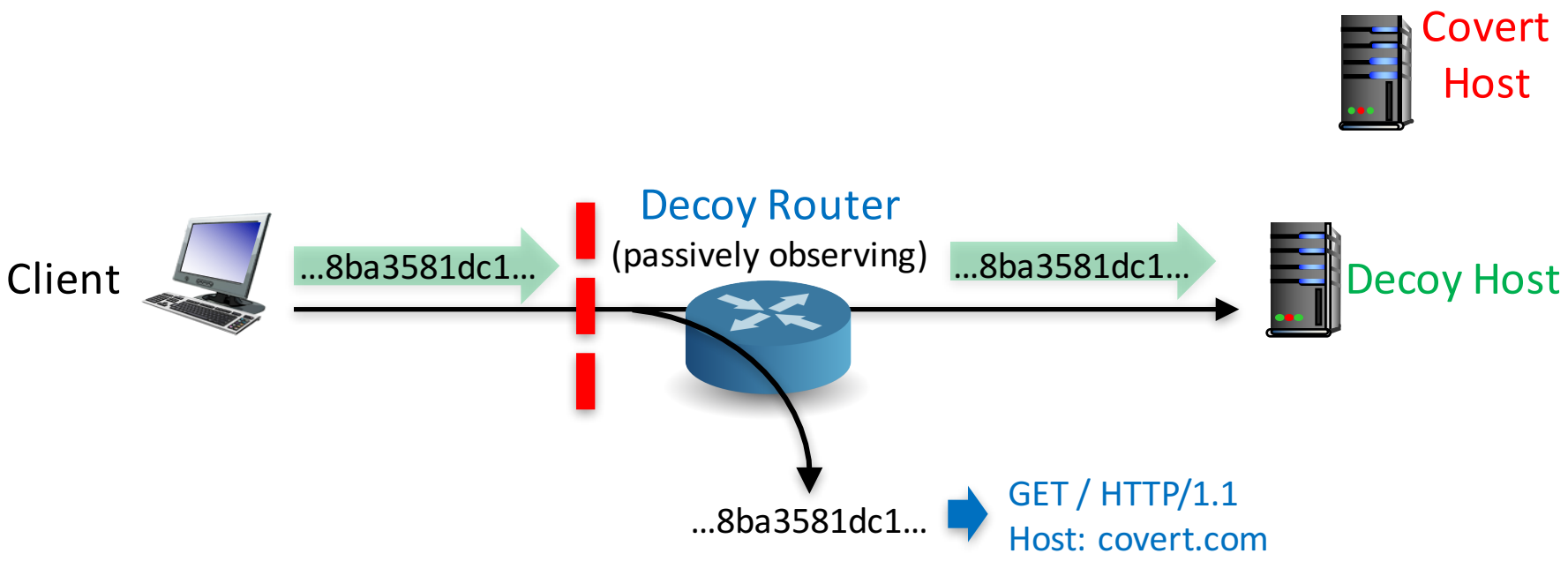
Step 4: Client exfiltrates "virtual" message board info:
Tunnel e.g., HTTP POST to replay or email address



MultiFlow protocol

Step 5: Client exfiltrates **request** for Covert Host to Decoy Router on Client-Decoy Host connection

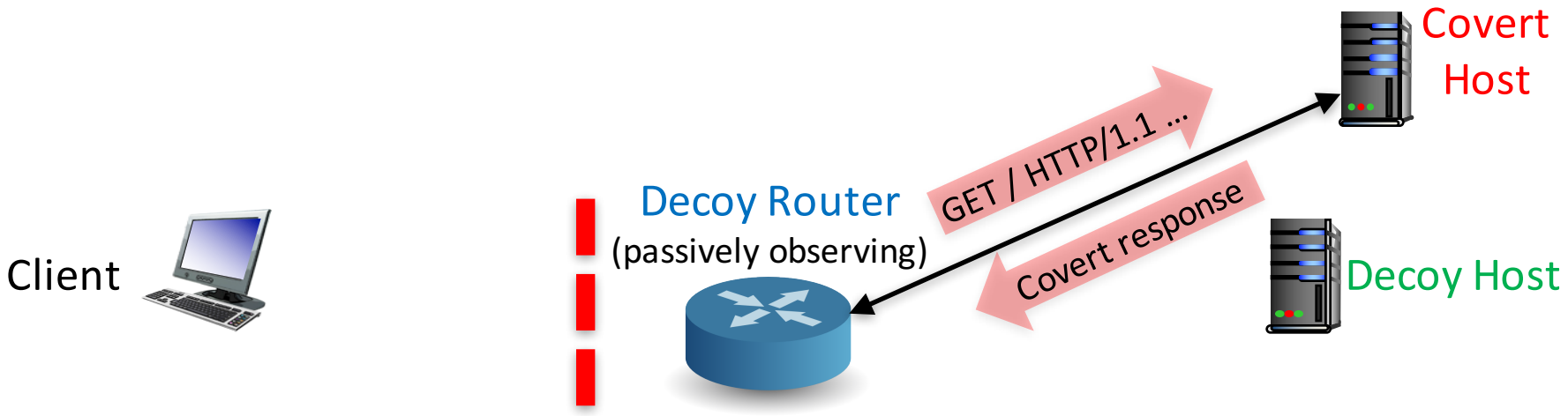
HTTP POST Tunnel



MultiFlow protocol

Step 6: Decoy Router connects to Covert Host, sends Client request, gets response

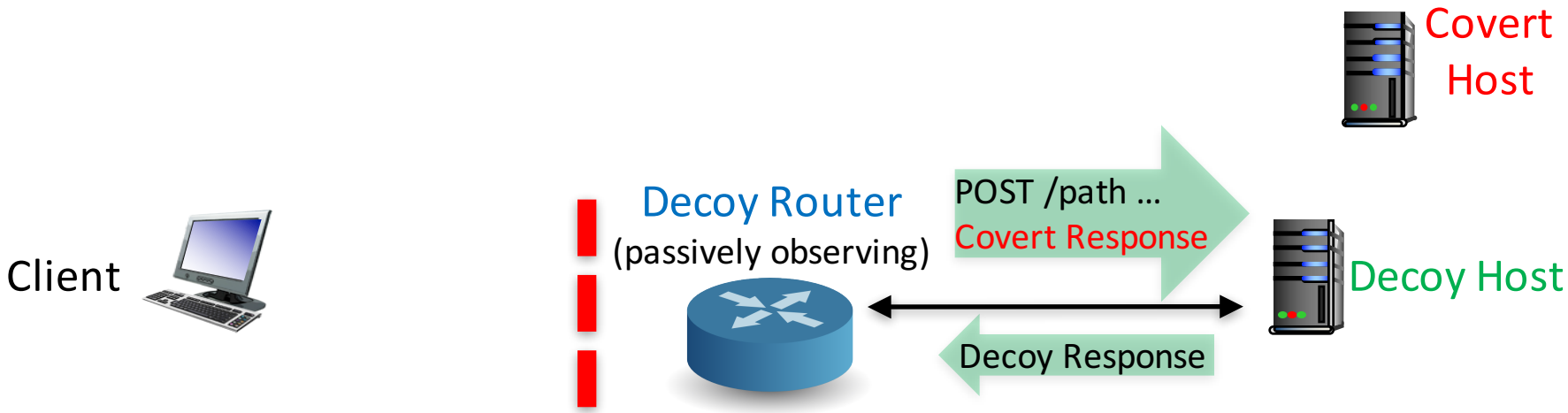
HTTP POST Tunnel



MultiFlow protocol

Step 7: Decoy Router **replays** Client POST but now (encrypted) covert response is payload

HTTP POST Tunnel

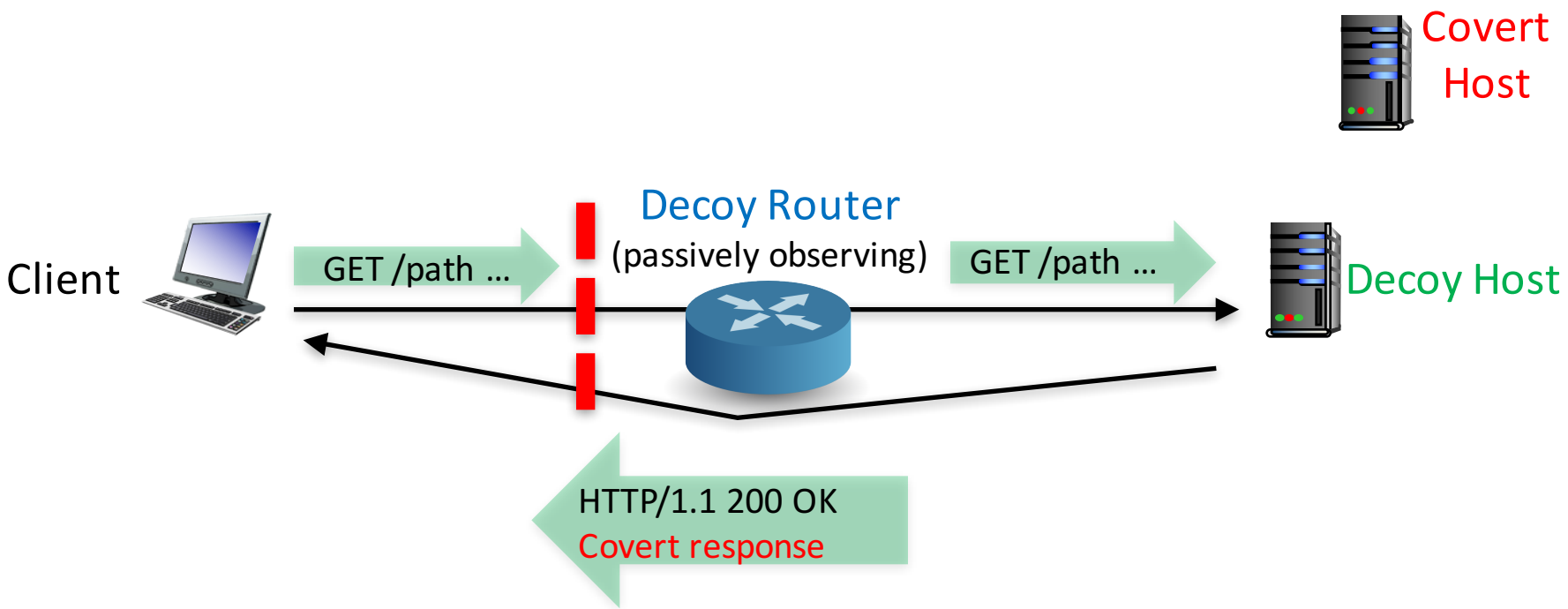


Benefit: Adversary never sees Decoy Router traffic, mitigates traffic analysis attacks

MultiFlow protocol

Step 8: Client re-downloads info from POST path on its own connection

HTTP POST Tunnel



Benefit: Decoy Router does not forge or rewrite Client-Decoy Host traffic, mitigates probing attacks

MultiFlow protocol

Step 8: Client re-downloads info from POST path on its own connection

HTTP POST Tunnel

Tunnel is complete!

Benefit: Decoy Router does not forge or rewrite Client-Decoy Host traffic, mitigates probing attacks

Issues, ideas, summary

MultiFlow's (potential) implementation issues

… aka possible bad things I didn't talk about

Information leakage

- Session resumption
- HTTP–posted data
- Message board info

Feasibility

- Replaying posts
- DR sending emails
- Decoy host "collusion"

Other ways to use MultiFlow protocol

… aka possible good things I didn't talk about

Asynchronous comm.
- Virtual symmetric routes
- Cross-server decoy routing

Combine with other DR protocols
- Tap-based authentication
- Virtual message-board

MultiFlow summary

Decoy routing protocol design to address deployment issues

Key ideas: Decoy Router resumes Client's TLS session, communicate via virtual message board

Benefits:

1. Tap-based but checks client liveness
2. No DR forging/rewriting of traffic
3. Resists TLS termination attack
4. Designed for TLS 1.3

vumanfredi@wesleyan.edu
https://victoriamanfredi.com