Spectrum

High-Bandwidth Anonymous Broadcast

Zachary Newman (zjn@mit.edu) & Sacha Servan-Schreiber (3s@mit.edu)

Joint work with Srini Devadas (MIT)

Elevator Pitch

Previous systems: anonymous Twitter.

Us: anonymous Twitch.

Highlights:

- 3–140x faster; overhead ~200 bytes compared to non-private broadcast
- Security against active attacks
- Share full-length documentary in 1.5 hours (w/ 10K users)

Whistleblowers expose corruption

The Washington Post
Nixon Resigns
Ford Assumes Presidency Today

“I’M THE GUY THEY CALLED DEEP THROAT”

Despite three decades of intense speculation, the identity of “Deep Throat”—the source who leaked key details of Nixon’s Watergate cover-up to *Washington Post* reporters Bob Woodward and Carl Bernstein—has never been revealed. Now, at age 91, W. Mark Felt, number two at the F.B.I. in the early 70s, is finally admitting to that historic, anonymous role. In an exclusive, JOHN D. O’CONNOR puts a name and face to one of American democracy’s heroes, learning about the struggle between honor and duty that nearly led Felt to take his secret to the grave

On a sunny California morning in August 1999, Joan Felt, a busy college Spanish professor and single mother, was completing chores before leaving for class. She stopped when she heard an unexpected knock at the front door. Upon answering it, she was met by a courteous, 50-ish man, who introduced himself as a journalist from *The Washington Post*. He asked if he could see her father, W. Mark Felt, who lived with her in her suburban Santa Rosa home. The man said his name was Bob Woodward.

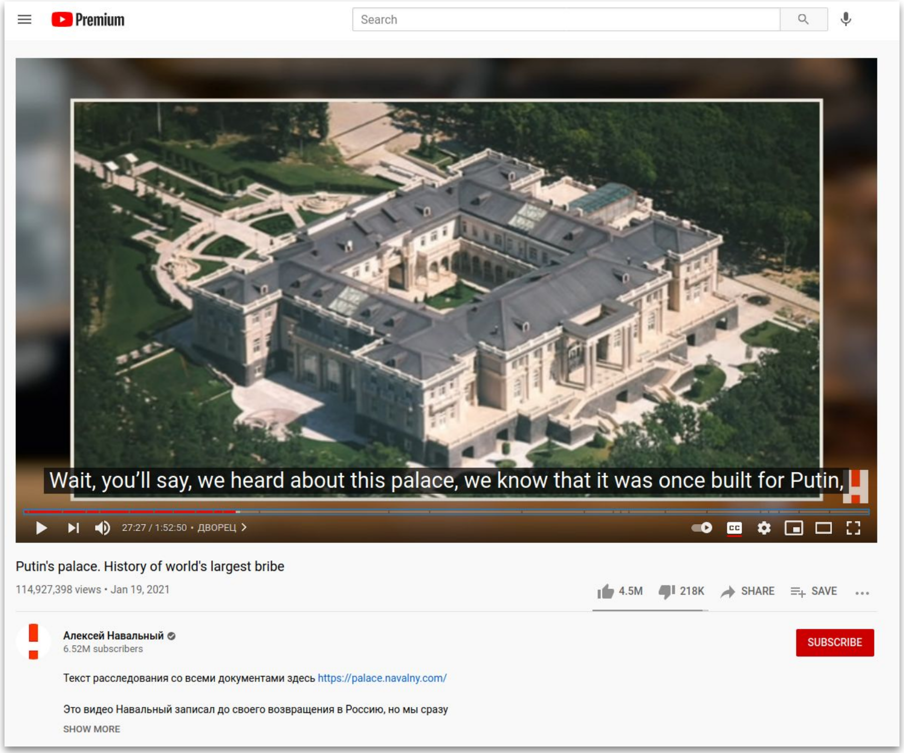
Woodward’s name did not register with Joan, and she assumed he was no different from a number of other reporters, who had called that week. This was, after all, the 25th anniversary of the resignation of President Richard Nixon, disgraced in the scandal known as Watergate, and hounded from office in 1974. The journalists had all

PHOTOGRAPH BY GASPER TRINGALE

JULY 2005

GRANDPA G-MAN

Former F.B.I. official W. Mark Felt, 91, is now a retiree living in Santa Rosa, California. He has told friends and intimates that he was the confidential inside source of Watergate fame.



But often at great cost

Whistleblowing—*Is It Really Worth the Consequences?*

by Kayla L. Delk, JD, BSN, RN

This article explores general principles of state whistleblower laws and alerts nurses to considerations when deciding whether to report an employer violation. Reporting an employer violation can be difficult for any employee but especially for nurses because nurses are torn between their desire and duty to advocate for clients' safety and their desire to maintain employment. The author suggests questions to consider when deciding to "blow the whistle" and alerts nurses to statutes of limitations that may affect when nurses must report violations. The article also illuminates policy and procedural issues for various states that affect how and to whom nurses report violations to protect themselves under whistleblower protection laws. Finally, this article explores personal and professional consequences that nurses should consider before reporting violations.

This article explores when and how to "blow the whistle" and alerts nurses to the consequences of blowing the whistle. However, this is merely a general overview of state whistleblower laws; nurses who intend to blow the whistle should seek legal advice from an attorney knowledgeable about each state's specific statutes regarding whistleblowing.

BEFORE BLOWING THE WHISTLE

Because whistleblowing can have deleterious effects on nurses' professional and personal lives, they

Putin critic Navalny jailed in Russia despite protests

2 February

alny says the latest case against him was fabricated

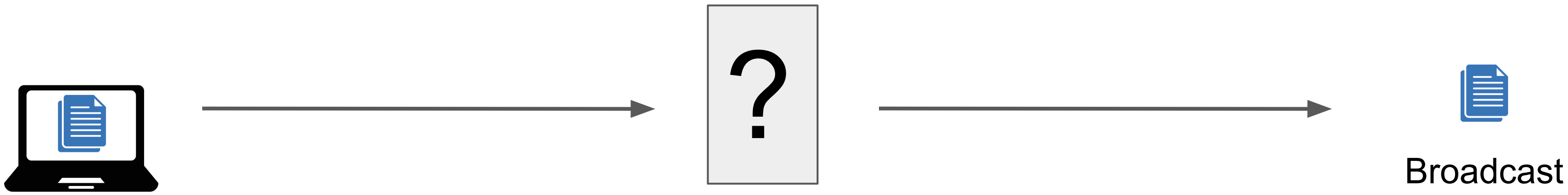
MEDIA

Panama Papers journalist killed by car bomb

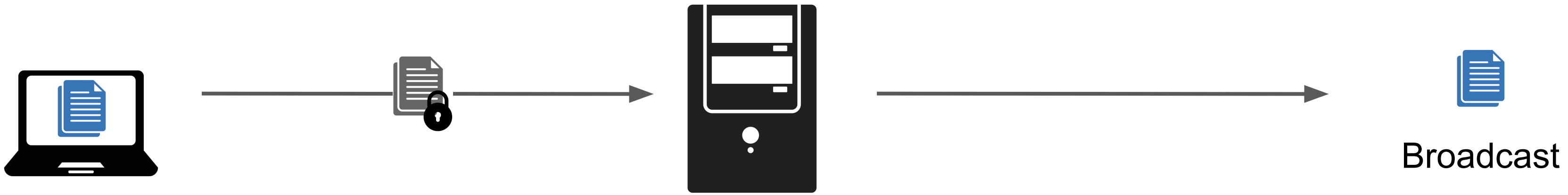
By Yaron Steinbuch

October 16, 2017 | 3:24pm | Updated

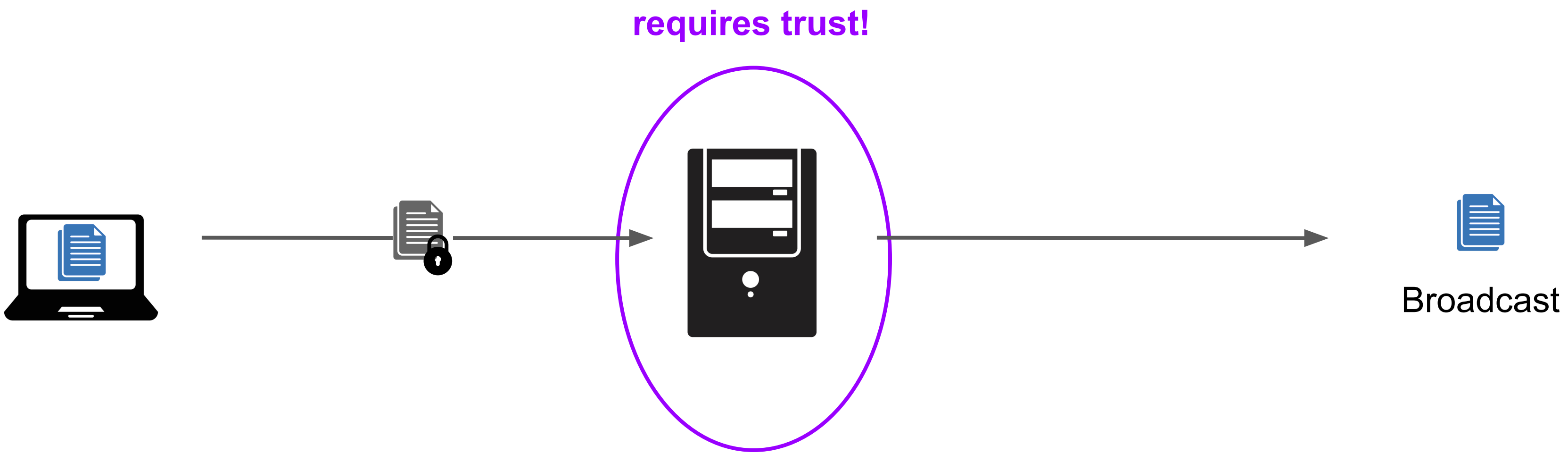
Can they publish anonymously?



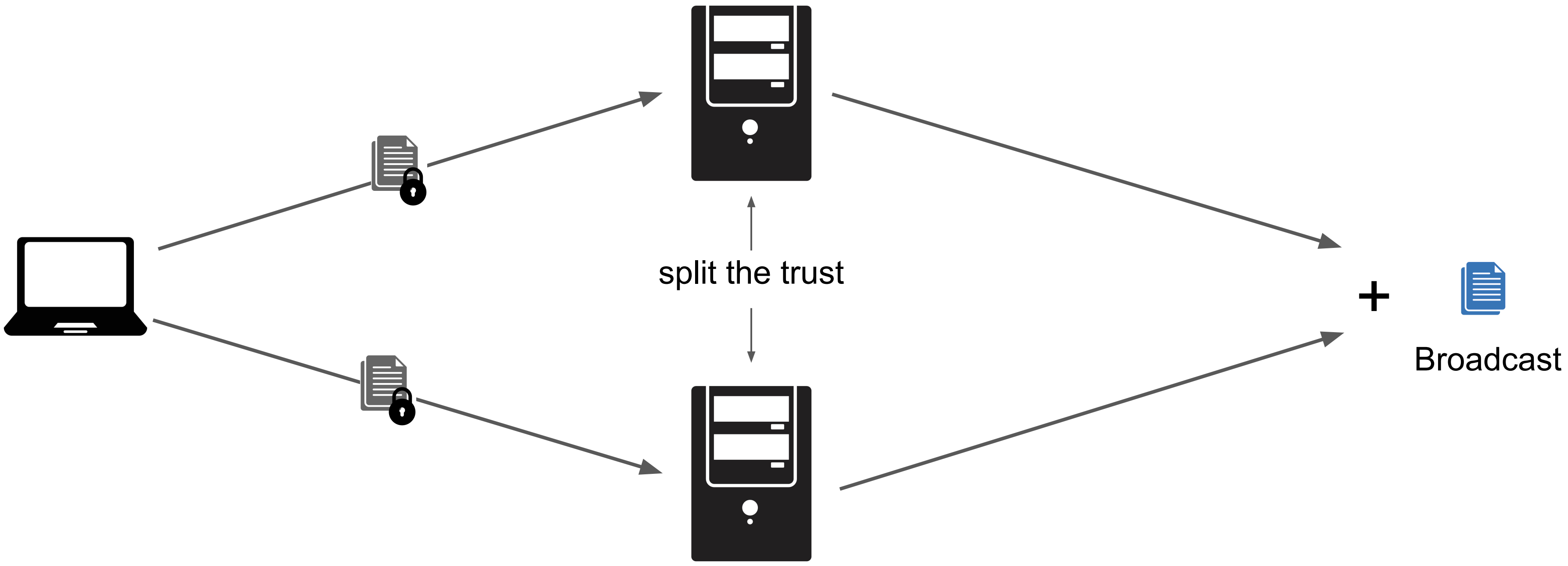
Can they publish anonymously?



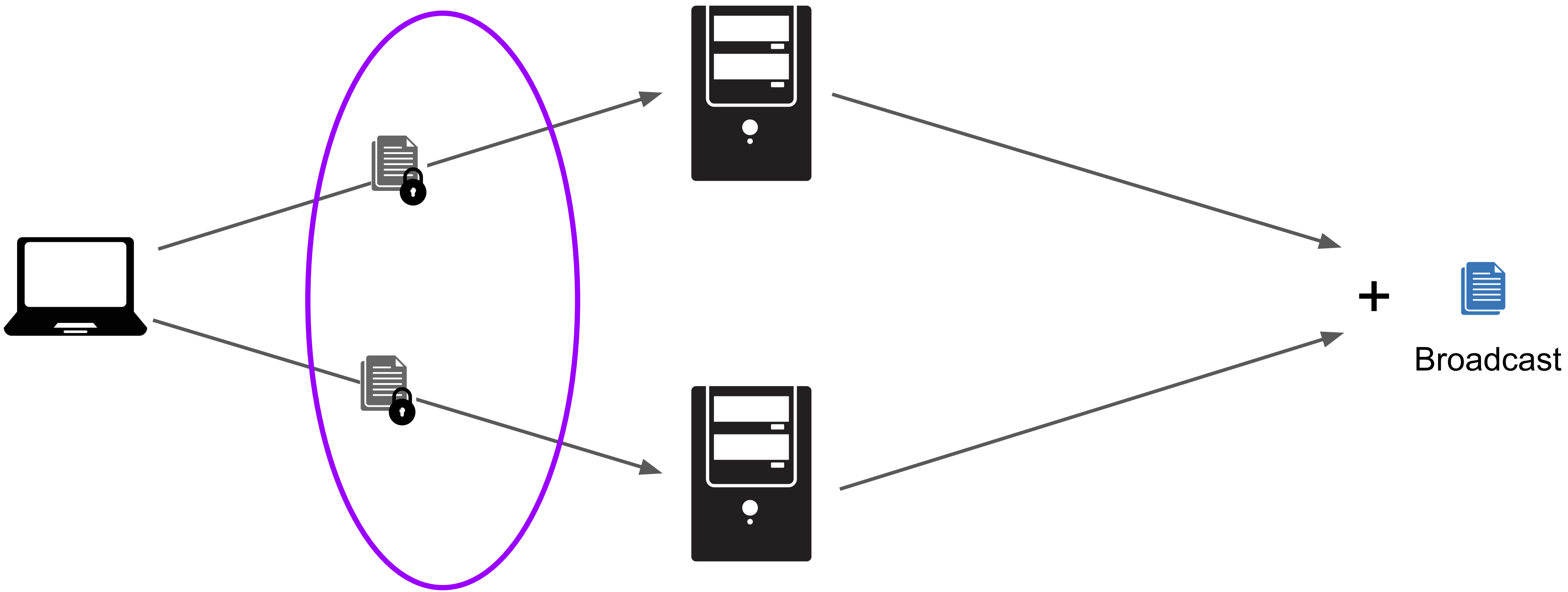
Can they publish anonymously?



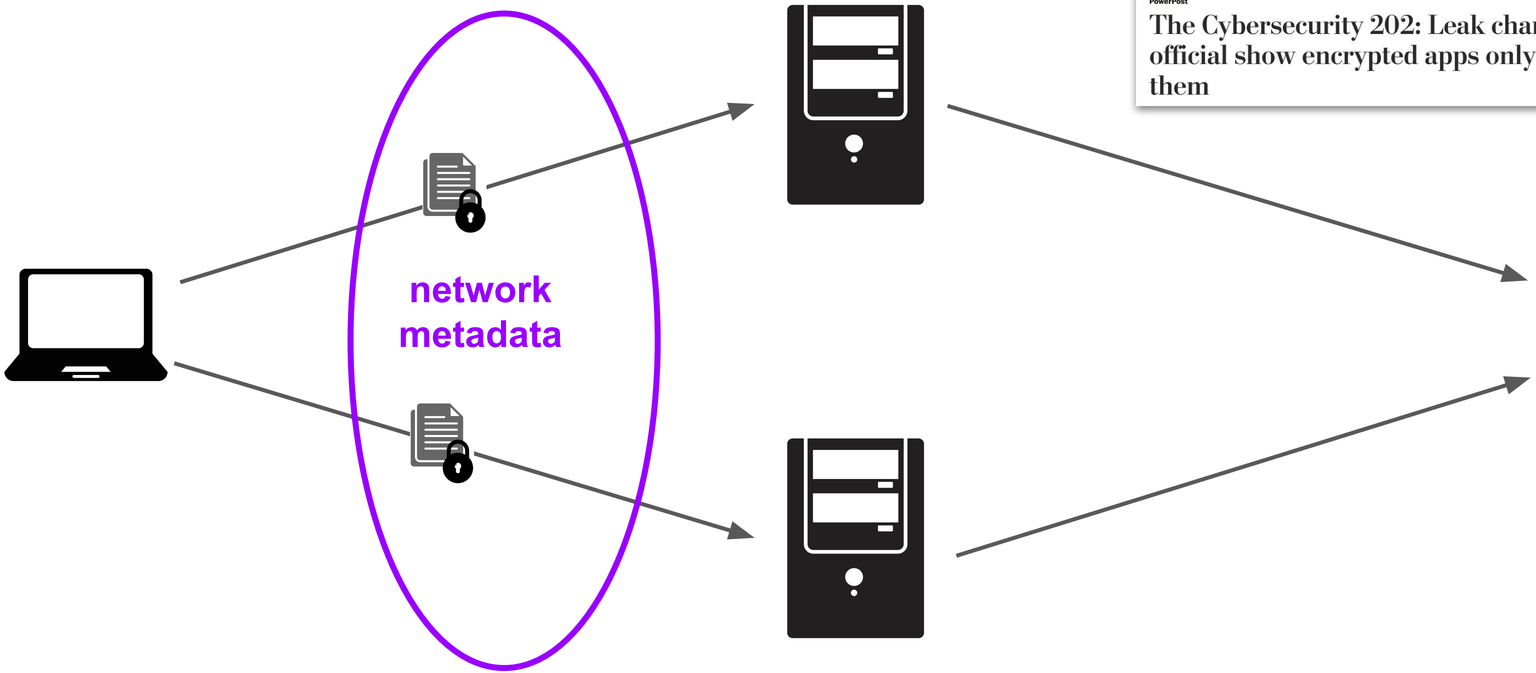
Can they publish anonymously?



Can they publish anonymously?



Can they publish anonymously?



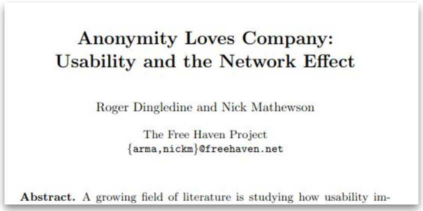
The Washington Post
Democracy Dies in Darkness

Politics | Live updates | Congress | Courts & Law | Fact Checker | The Fix | Monkey Cage | Polling | White House

PowerPost

The Cybersecurity 202: Leak charges against Treasury official show encrypted apps only as secure as you make them

Can they publish anonymously?

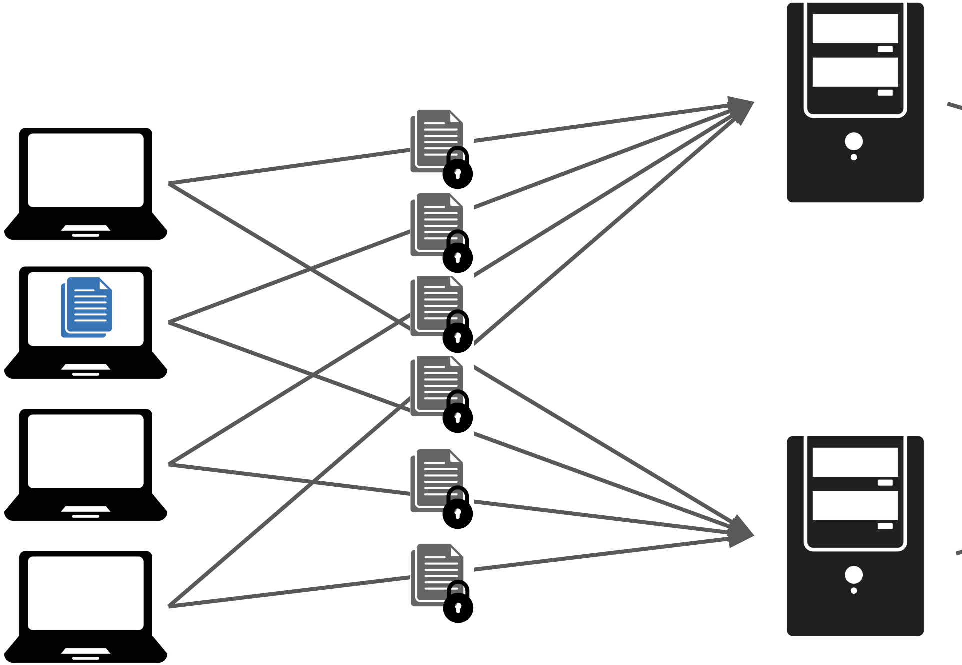


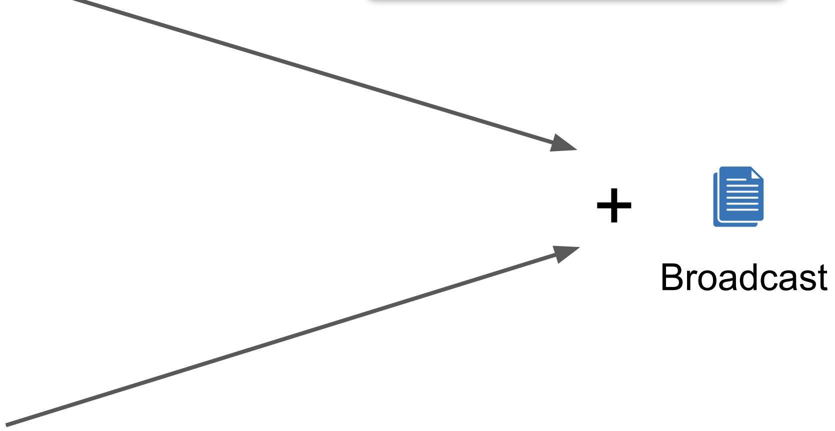
Anonymity Loves Company:
Usability and the Network Effect

Roger Dingledine and Nick Mathewson

The Free Haven Project
{arma,nickm}@freehaven.net

Abstract. A growing field of literature is studying how usability im-





Dining Cryptographer Nets (DC-nets)

Simple XOR-based scheme (DC-net)

$m_i = \boldsymbol{m}$ Broadcaster

Simple XOR-based scheme (DC-net)

$m_i = \boldsymbol{m}$ Broadcaster

$m_i = 0$ Other clients

Simple XOR-based scheme (DC-net)

$m_i = \boldsymbol{m}$ Broadcaster

$m_i = 0$ Other clients

$r_i \oplus m_i$ to Server A

r_i to Server B

Simple XOR-based scheme (DC-net)

$m_i = \boldsymbol{m}$ Broadcaster

$m_i = 0$ Other clients

$r_i \oplus m_i$ to Server A

r_i to Server B

$$\boldsymbol{m} = \overbrace{\bigoplus_{i}^{N} (r_i \oplus m_i)}^{\text{ServerA}} \oplus \overbrace{\bigoplus_{i}^{N} r_i}^{\text{ServerB}} = \overbrace{\bigoplus_{i}^{N} (\cancel{r_i \oplus r_i}) \oplus \bigoplus_{i}^{N} m_i}^{(commutativity\ of\ xor)}$$

$$= \underbrace{0 \oplus \ldots \oplus \boldsymbol{m} \oplus \ldots \oplus 0}_{origin\ of\ \boldsymbol{m}\ is\ hidden}.$$

Simple XOR-based scheme (DC-net)

$m_i = \boldsymbol{m}$ Broadcaster

$m_i = 0$ Other clients

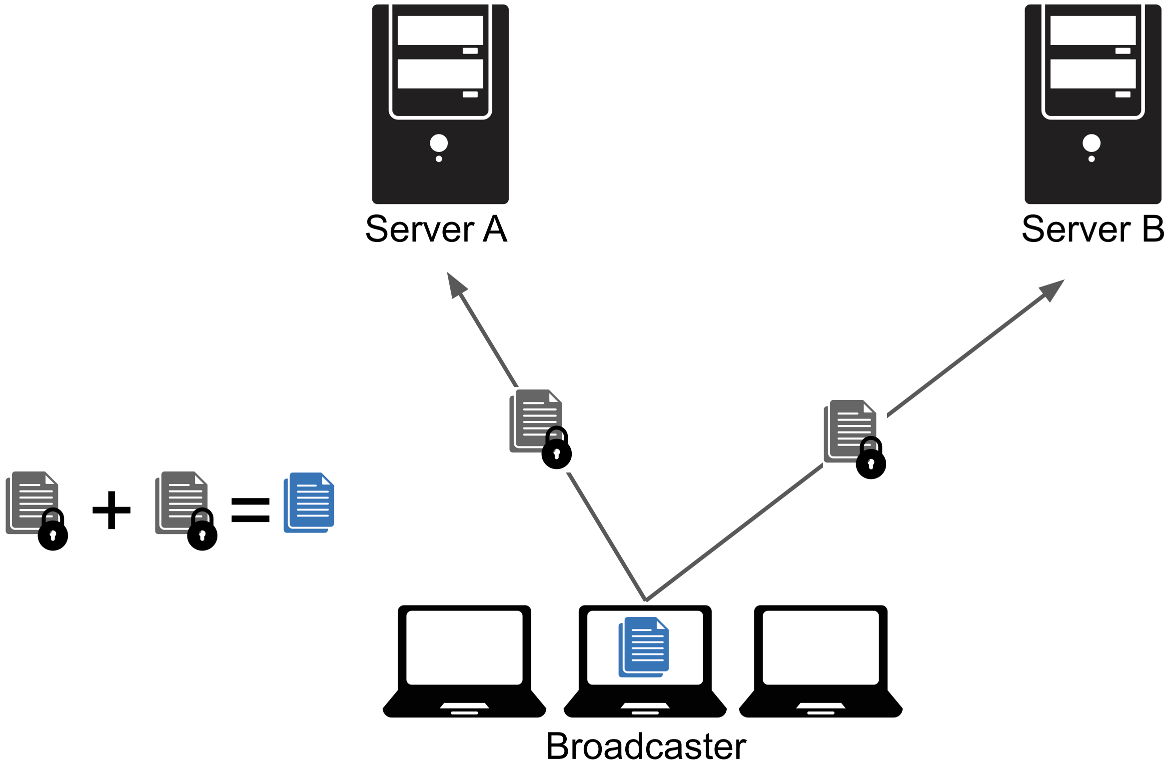
$r_i \oplus m_i$ to Server A

r_i to Server B

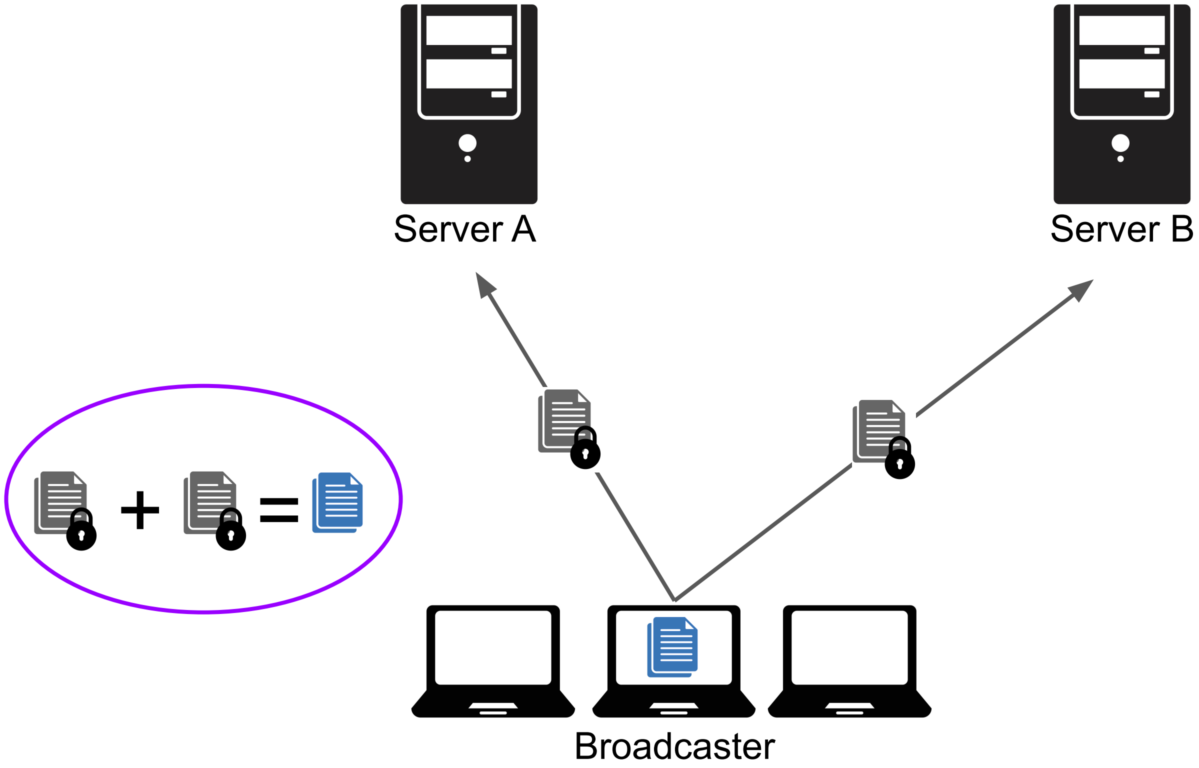
$$\boldsymbol{m} = \overbrace{\bigoplus_i^N (r_i \oplus m_i)}^{\text{ServerA}} \oplus \overbrace{\bigoplus_i^N r_i}^{\text{ServerB}} = \overbrace{\bigoplus_i^N (\cancel{r_i \oplus r_i}) \oplus \bigoplus_i^N m_i}^{(commutativity\ of\ xor)}$$

$$\boxed{= \underbrace{0 \oplus \ldots \oplus \boldsymbol{m} \oplus \ldots \oplus 0}_{origin\ of\ \boldsymbol{m}\ is\ hidden}.}$$

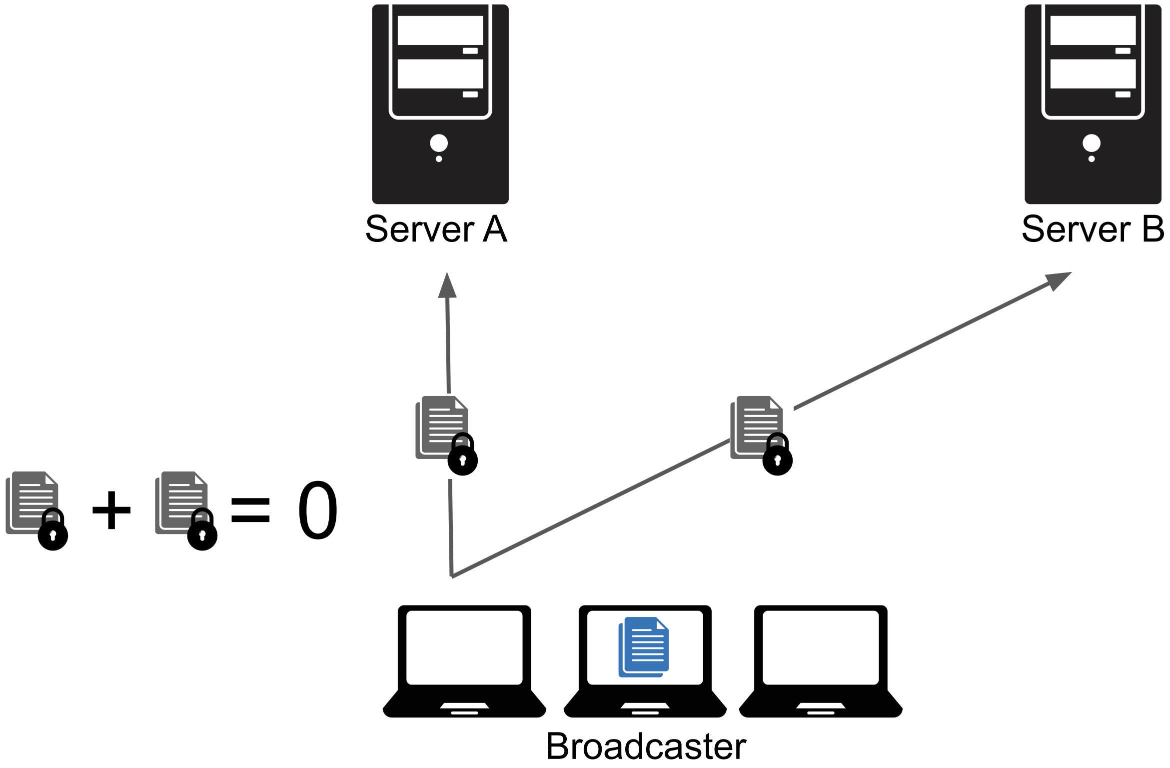
A toy protocol



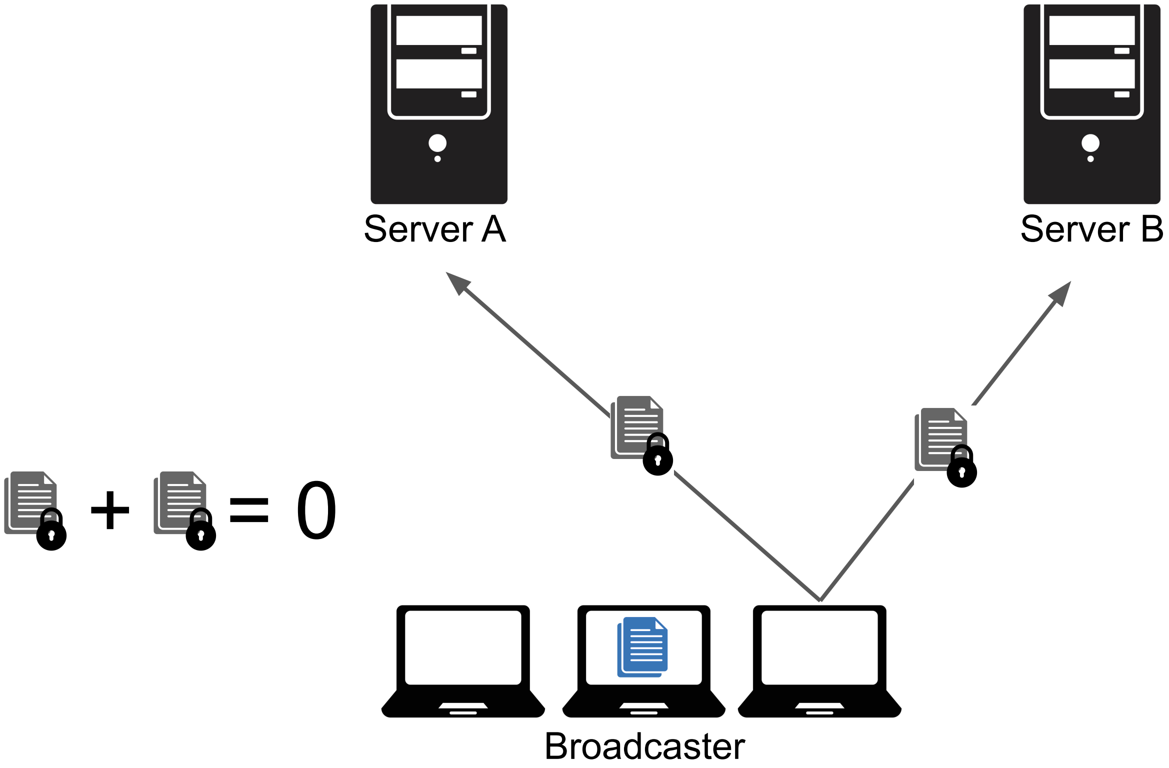
A toy protocol



A toy protocol

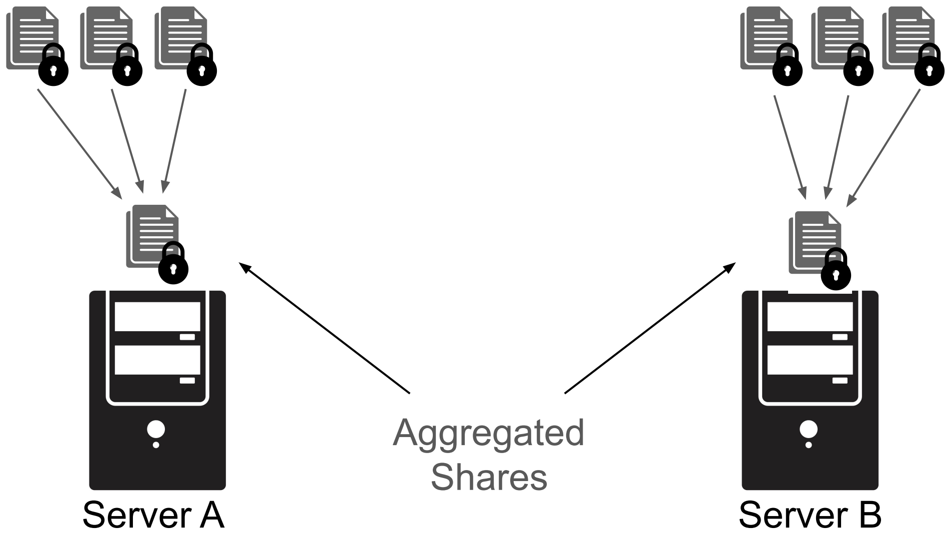


A toy protocol

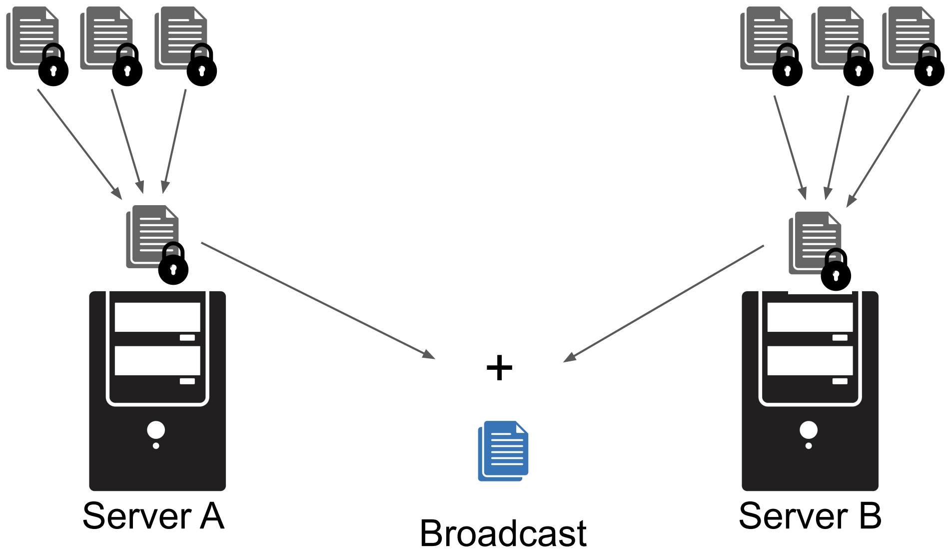


A toy protocol

A toy protocol



A toy protocol



Broadcast origin hidden

DC-Nets

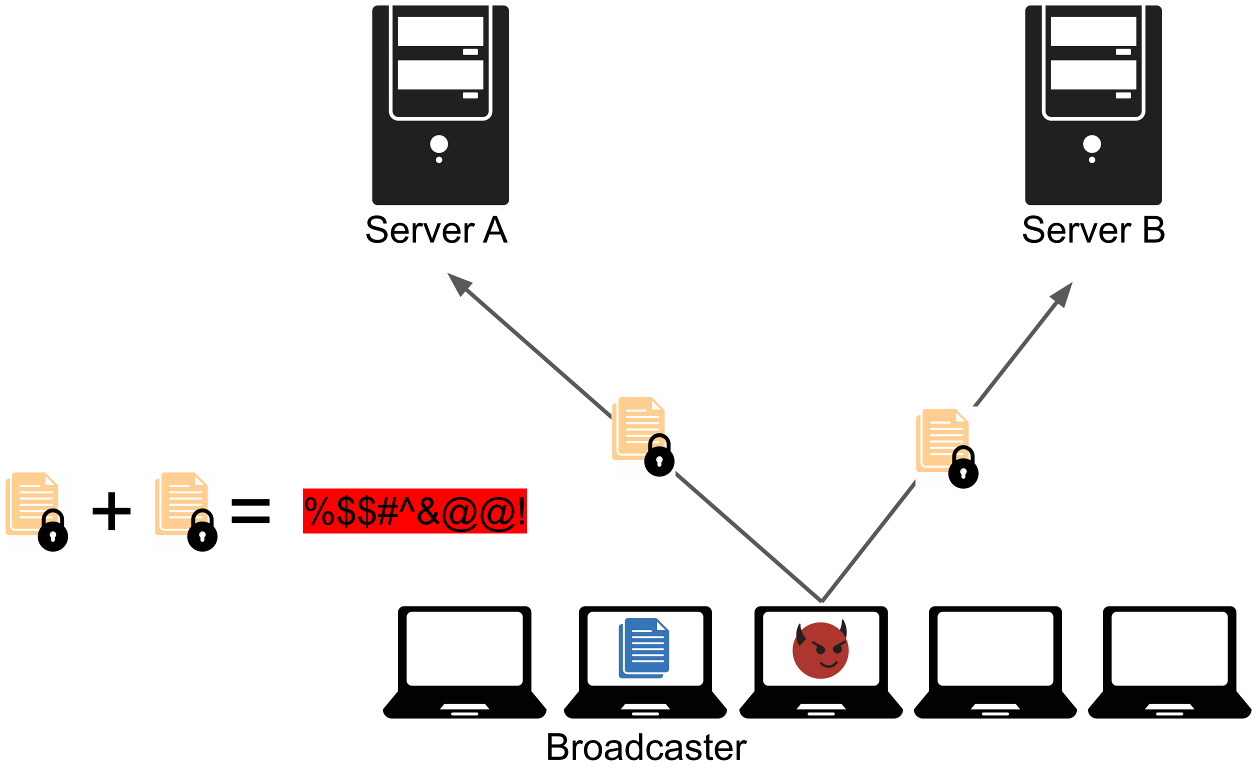
Pros

- **Fast:** server XORs requests
- **Metadata private:** fully hides broadcast source
- **High bandwidth:** throughput matches client upload bandwidth

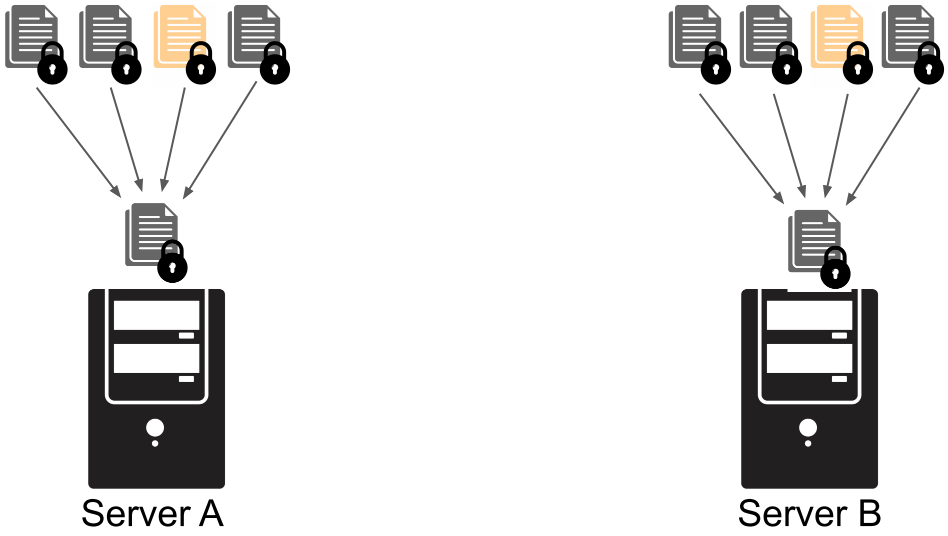
Cons

- **Not-scalable:** only one broadcaster (unless repeated ⇒ extra bandwidth)
- **Insecure:** client can *undetectably* disrupt the broadcast

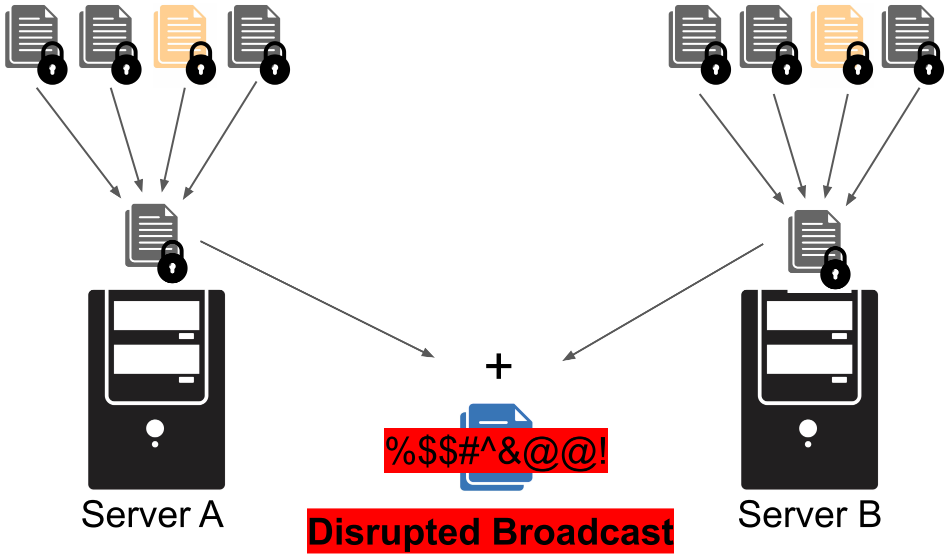
Broadcast Disruption



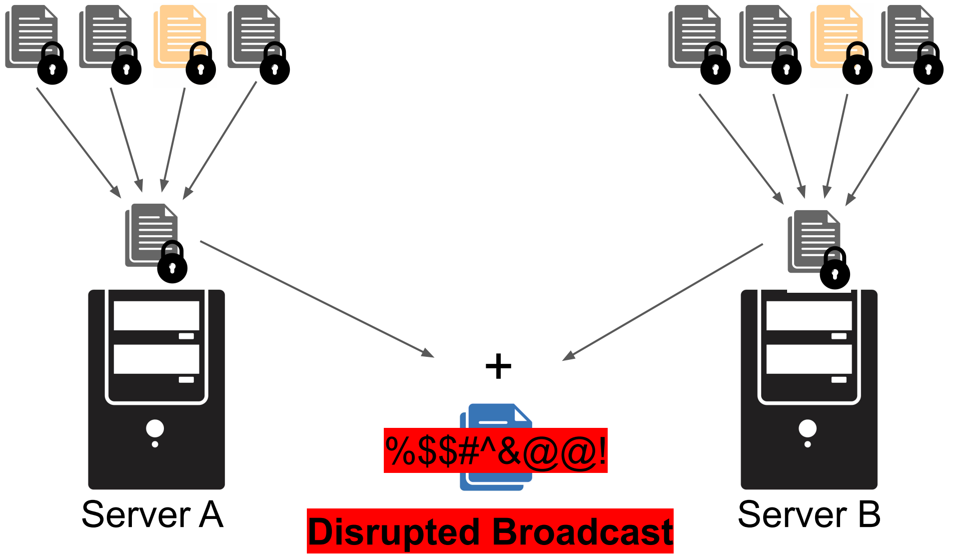
Broadcast Disruption



Broadcast Disruption



Broadcast Disruption

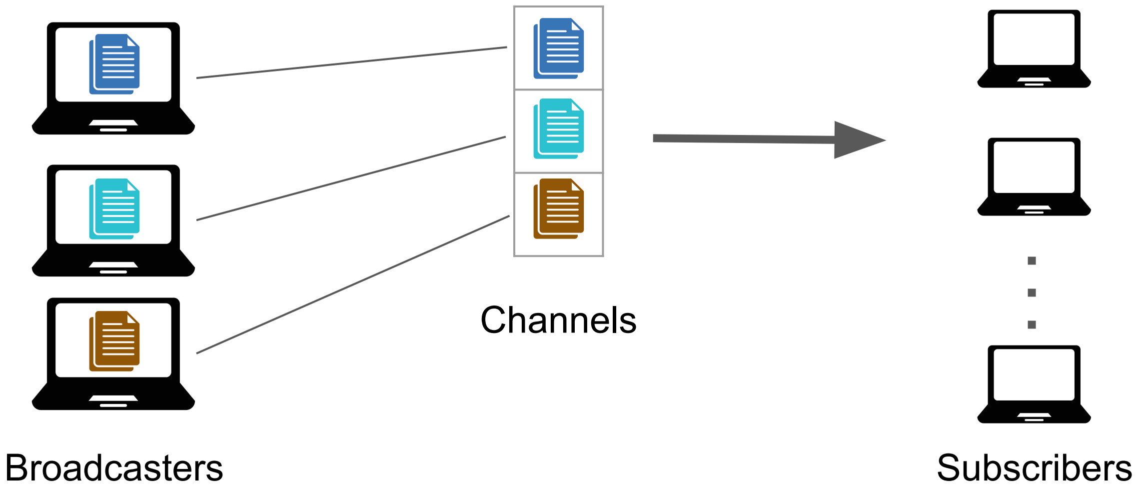


Malicious users can corrupt broadcasts = censorship

Definition: A Broadcast Channel

A **channel** is an allocated slot to which a message is written

- Multiple *simultaneous* broadcasts can be supported with multiple channels
  - A user can write to a channel without clobbering another message on another slot
- Servers can't tell which user is writing to which channel



Existing Approaches for Preventing Disruption

Riposte [CGBM15]

- Allocate Ω(# users) channel slots
- Clients pick **one random channel** to write to
  - Prevents collision with malicious writes w.h.p.
- Additional non-colluding "audit" server required to prevent malicious clients

Blinder [APY20]

- Allocate Ω(# users) channel slots
- Clients pick **one random channel** to write to (like Riposte)
- Honest-majority MPC instead of audit server to prevent malicious clients

Existing Approaches for Preventing Disruption

Common Theme: # channels ≫ # users

⇒ more server work than DC-net

⇒ only efficient with small messages

sts our Riposte
160-byte rows

and up to 1 million clients. Second, we show that Blinder scales to
large messages of up to 10 kilobytes (required by some applications

Can we do better?

Existing Approaches for Preventing Disruption

Common Theme: # channels ≫ # users

⇒ more server work than DC-net

⇒ only efficient with small messages

sts our Riposte
160-byte rows

and up to 1 million clients. Second, we show that Blinder scales to
large messages of up to 10 kilobytes (required by some applications

Can we do better?

1. One channel per broadcaster?

Existing Approaches for Preventing Disruption

Common Theme: # channels ≫ # users

⇒ more server work than DC-net

⇒ only efficient with small messages

sts our Riposte
160-byte rows

and up to 1 million clients. Second, we show that Blinder scales to
large messages of up to 10 kilobytes (required by some applications

Can we do better?

1. One channel per broadcaster?
2. Optimize for few broadcasters, many users?

Our approach

- One channel per broadcaster ⇒ need *authorized* writes
    - Otherwise, malicious clients can disrupt the broadcast

Our approach

- One channel per broadcaster ⇒ need *authorized* writes
  - Otherwise, malicious clients can disrupt the broadcast
- Solution: **Blind** message authentication for access control
  - To write to a channel, must know its secret key

Our approach

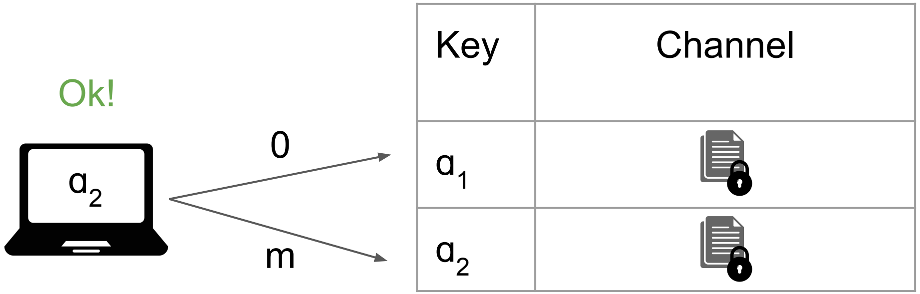
- One channel per broadcaster ⇒ need *authorized* writes
  - Otherwise, malicious clients can disrupt the broadcast
- Solution: **Blind** message authentication for access control
  - To write to a channel, must know its secret key
  - Servers do not know which client is the broadcaster

Our approach

- One channel per broadcaster ⇒ need *authorized* writes
  - Otherwise, malicious clients can disrupt the broadcast
- Solution: **Blind** message authentication for access control
  - To write to a channel, must know its secret key
  - Servers do not know which client is the broadcaster
  - Server work: $O(\text{ \# channels })$, instead of $\Omega(\text{ \# users })$

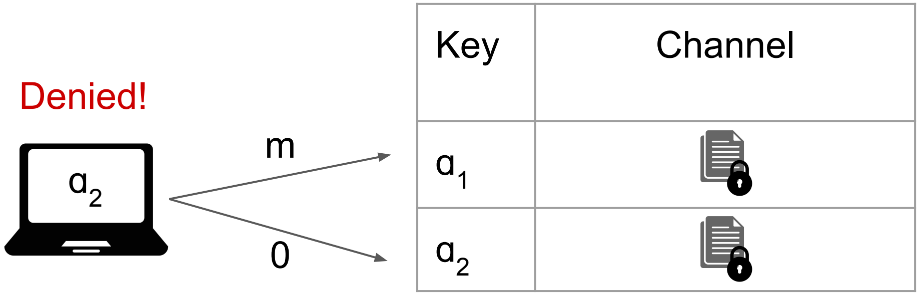
Our approach

- One channel per broadcaster ⇒ need *authorized* writes
  - Otherwise, malicious clients can disrupt the broadcast
- Solution: **Blind** message authentication for access control
  - To write to a channel, must know its secret key
  - Servers do not know which client is the broadcaster
  - Server work: O(# channels), instead of Ω(# users)



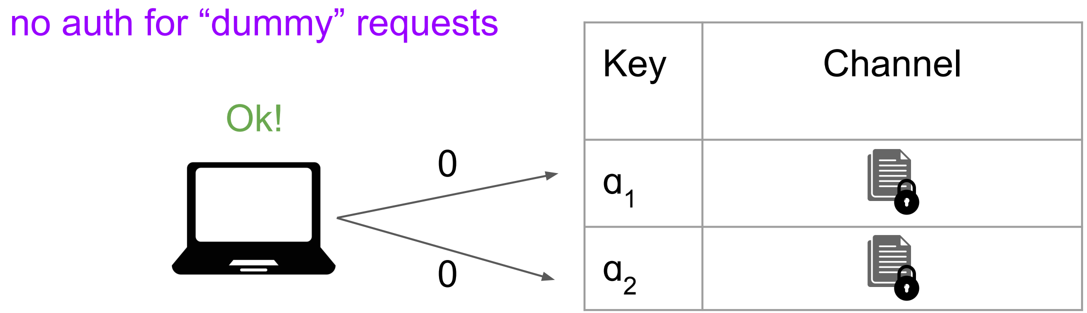
Our approach

- One channel per broadcaster ⇒ need *authorized* writes
  - Otherwise, malicious clients can disrupt the broadcast
- Solution: **Blind** message authentication for access control
  - To write to a channel, must know its secret key
  - Servers do not know which client is the broadcaster
  - Server work: O(# channels), instead of Ω(# users)



Our approach

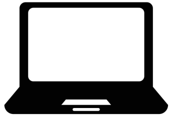
- One channel per broadcaster ⇒ need *authorized* writes
  - Otherwise, malicious clients can disrupt the broadcast
- Solution: **Blind** message authentication for access control
  - To write to a channel, must know its secret key
  - Servers do not know which client is the broadcaster
  - Server work: O(# channels), instead of Ω(# users)



Our approach

- One channel per broadcaster ⇒ need *authorized* writes
  - Otherwise, malicious clients can disrupt the broadcast
- Solution: **Blind** message authentication for access control
  - To write to a channel, must know its secret key
  - Servers do not know which client is the broadcaster
  - Server work: O(# channels), instead of Ω(# users)

servers can’t tell what key was used (if any)



Key	Channel
a_1	
a_2	

Protocol

Outline

1. Supporting multiple channels
2. Blind message authentication
3. Preventing de-anonymization attacks

Multiple channels (almost) for free

- Example scenario:
  - 5 broadcasters, each wants to share a 1MB message
  - If repeated in parallel then *every user* must upload 5 MB
  - Quickly saturates upload bandwidth

Multiple channels (almost) for free

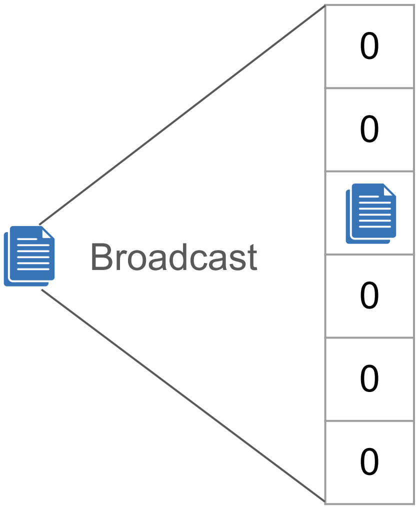
- Example scenario:
  - 5 broadcasters, each wants to share a 1MB message
  - If repeated in parallel then *every user* must upload 5 MB
  - Quickly saturates upload bandwidth
- Want: each user only uploads ~1MB!
  - Servers must still extract 5 “real” messages
  - From many random-looking messages

Multiple channels (almost) for free

- Example scenario:
  - 5 broadcasters, each wants to share a 1MB message
  - If repeated in parallel then *every user* must upload 5 MB
  - Quickly saturates upload bandwidth
- Want: each user only uploads ~1MB!
  - Servers must still extract 5 “real” messages
  - From many random-looking messages
- Must write to *every* channel for *every* message
  - Otherwise the servers learn who is broadcasting

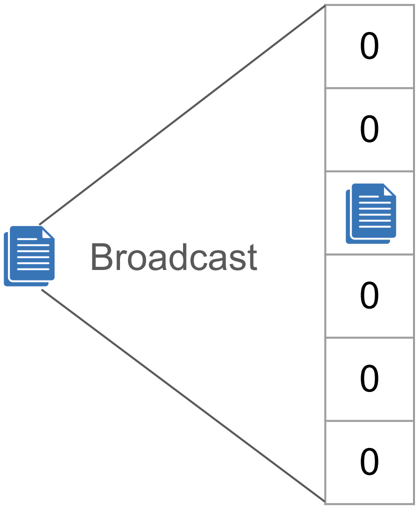
Multiple channels (almost) for free

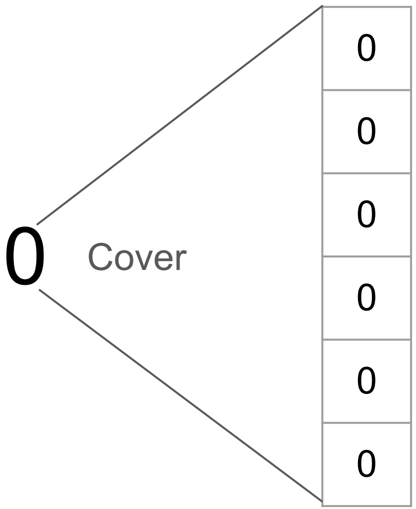
- Valid writes
  - *Point functions*



Multiple channels (almost) for free

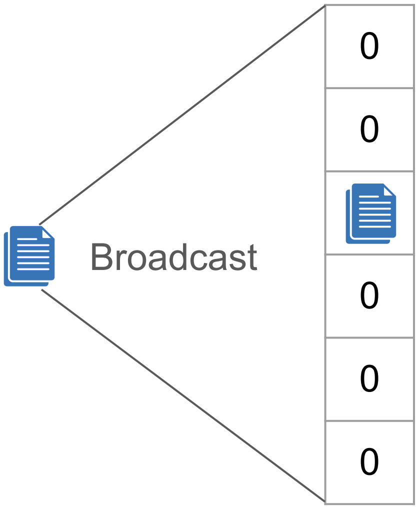
- Valid writes
  - *Point functions*
  - Or all-0 messages

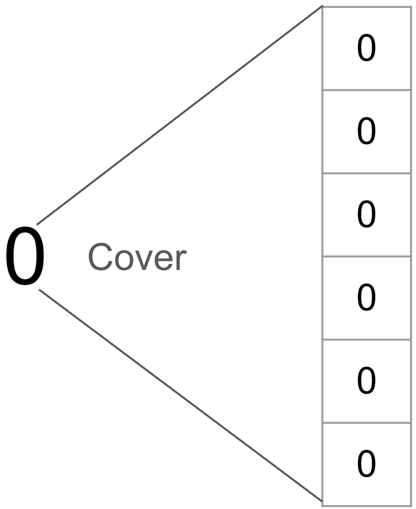




Multiple channels (almost) for free

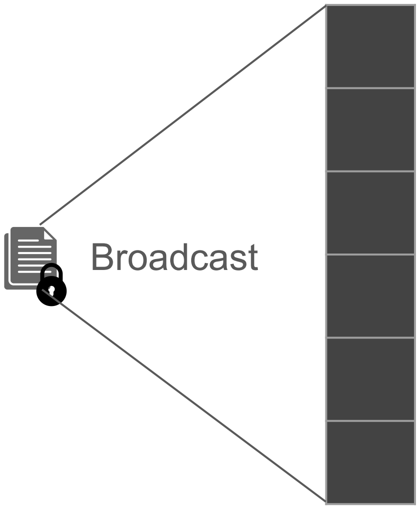
- Valid writes
  - *Point functions*
  - Or all-0 messages
- Want to send *short secret-shares* of our write

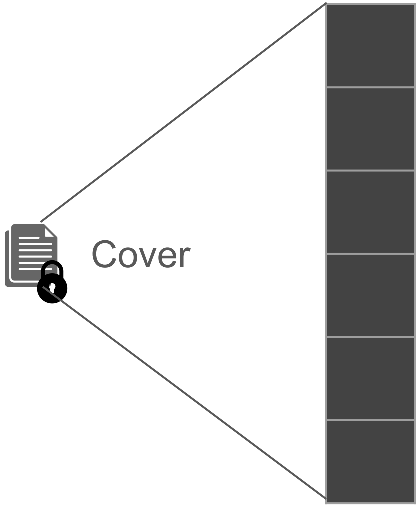




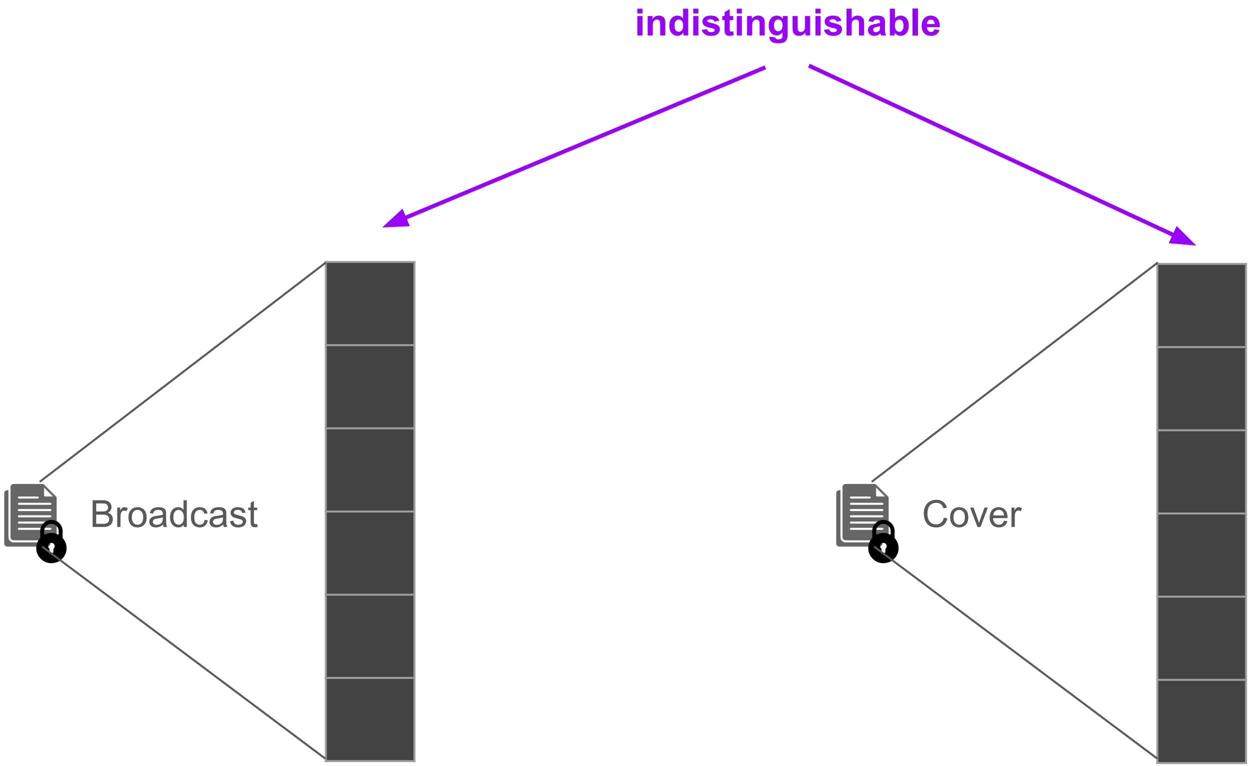
Distributed point function (DPF) [GI14]

- Want: short representation
- Want: secret shares



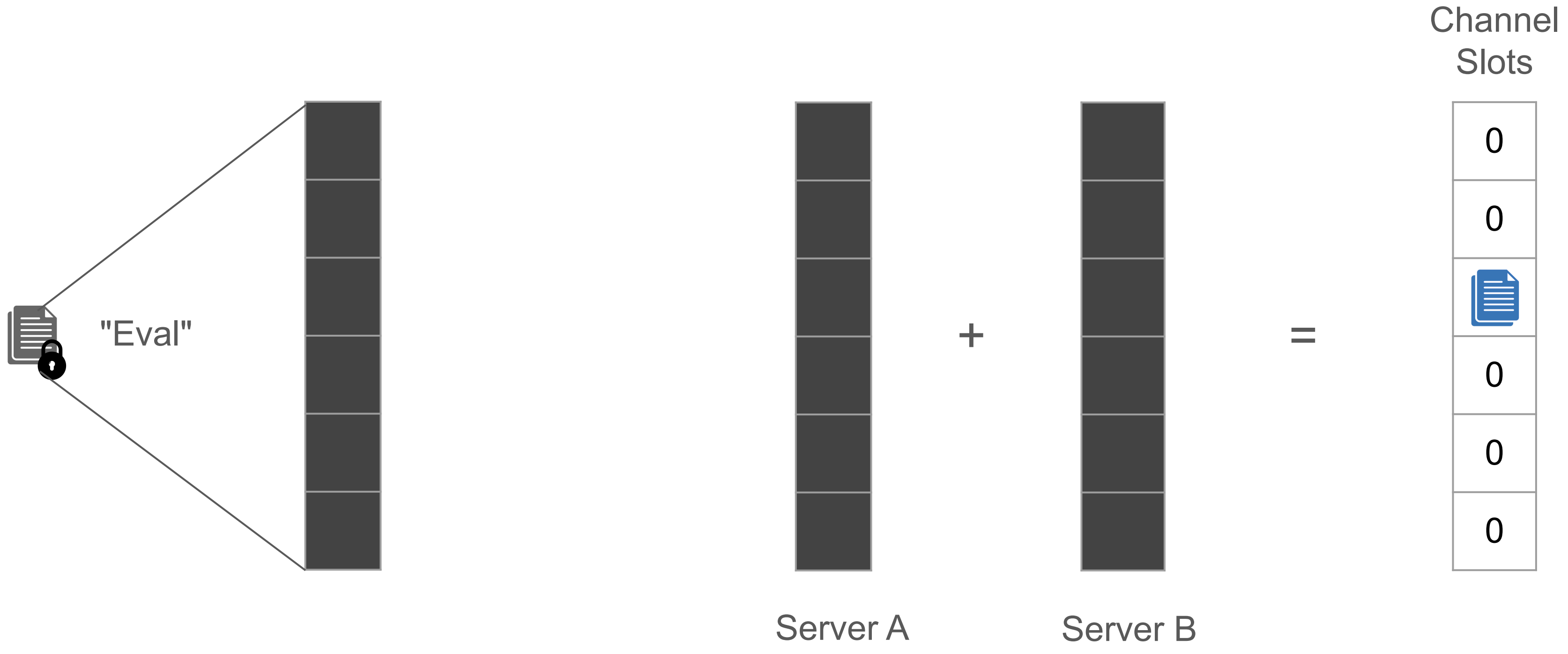


Distributed point function (DPF) [GI14]



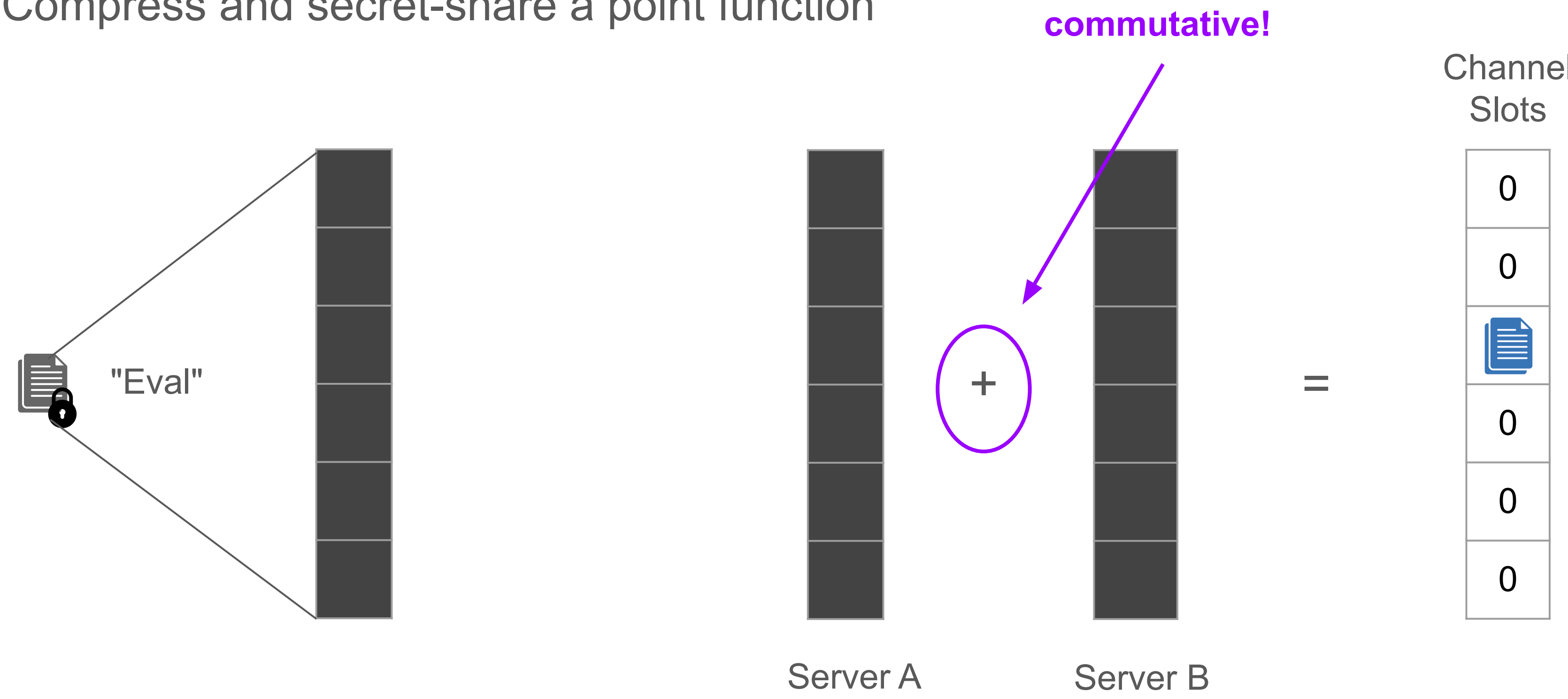
Distributed point function (DPF) [GI14]

Compress and secret-share a point function



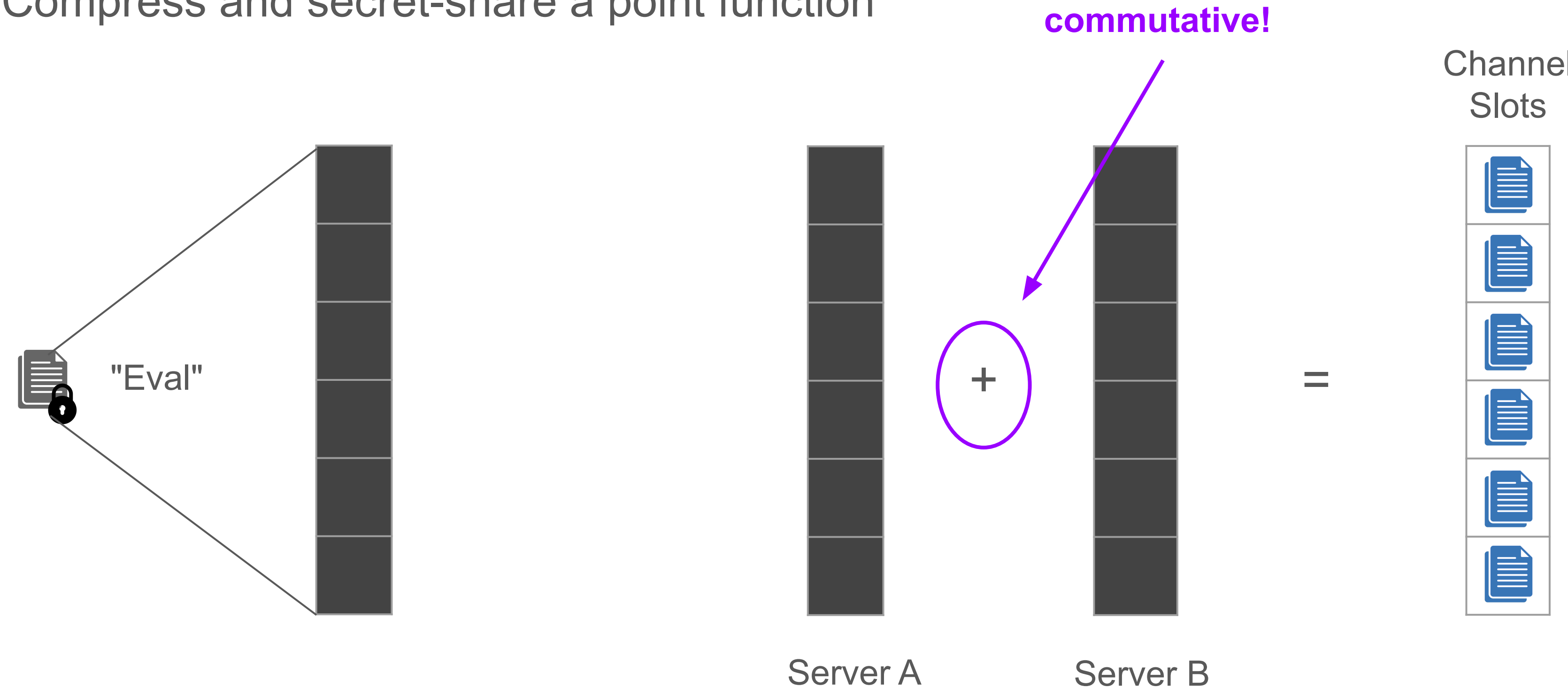
Distributed point function (DPF) [GI14]

Compress and secret-share a point function



Distributed point function (DPF) [GI14]

Compress and secret-share a point function



Blind message authentication

- For DPF, want to check (each channel): “is 0 **OR** authorized”
  - must perform check on secret shares
  - minimize interaction
  - must allow 0 messages, but they should look the same
  - malicious server shouldn’t be able to impersonate malicious client

Blind message authentication

- For DPF, want to check (each channel): “is 0 **OR** authorized”
- MAC the message [CW79,WC81]
  - can check over secret shares
  - verify that tagger knows a secret
  - *anybody* can tag empty messages

Blind message authentication

- For DPF, want to check: “every channel is 0 **OR** client knows the secret”
- MAC the message [CW79,WC81]

MAC tag

$$t = \alpha \cdot m$$

Blind message authentication

- For DPF, want to check: “every channel is 0 **OR** client knows the secret”
- MAC the message [CW79,WC81]

MAC tag

$$t = \alpha \cdot m$$

if m = 0, t = 0!

Blind message authentication

- For DPF, want to check: “every channel is 0 **OR** client knows the secret”
- MAC the message [CW79,WC81]
- Can compute *locally* on additive secret-shares

MAC tag

known to servers (for now)

$$\alpha \cdot [\![m]\!]_1 + \alpha \cdot [\![m]\!]_2 = t = \alpha \cdot m$$

Blind message authentication

- For DPF, want to check: “every channel is 0 **OR** client knows the secret”
- MAC the message [CW79,WC81]
- Can compute *locally* on additive secret-shares

secret shares of MAC tag

known to servers
(for now)

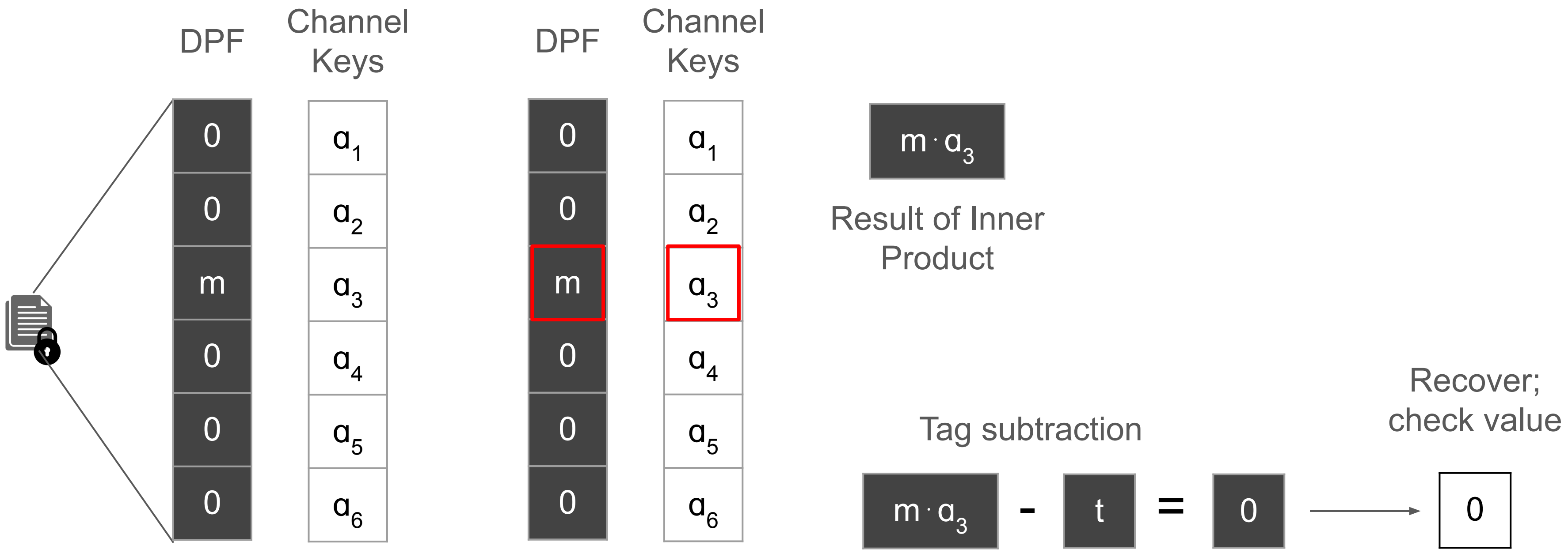
$$\begin{aligned} &\alpha \cdot [\![m]\!]_1 - [\![t]\!]_1 \\ &\quad + \qquad\qquad = 0 \\ &\alpha \cdot [\![m]\!]_2 - [\![t]\!]_2 \end{aligned}$$

Blind message authentication

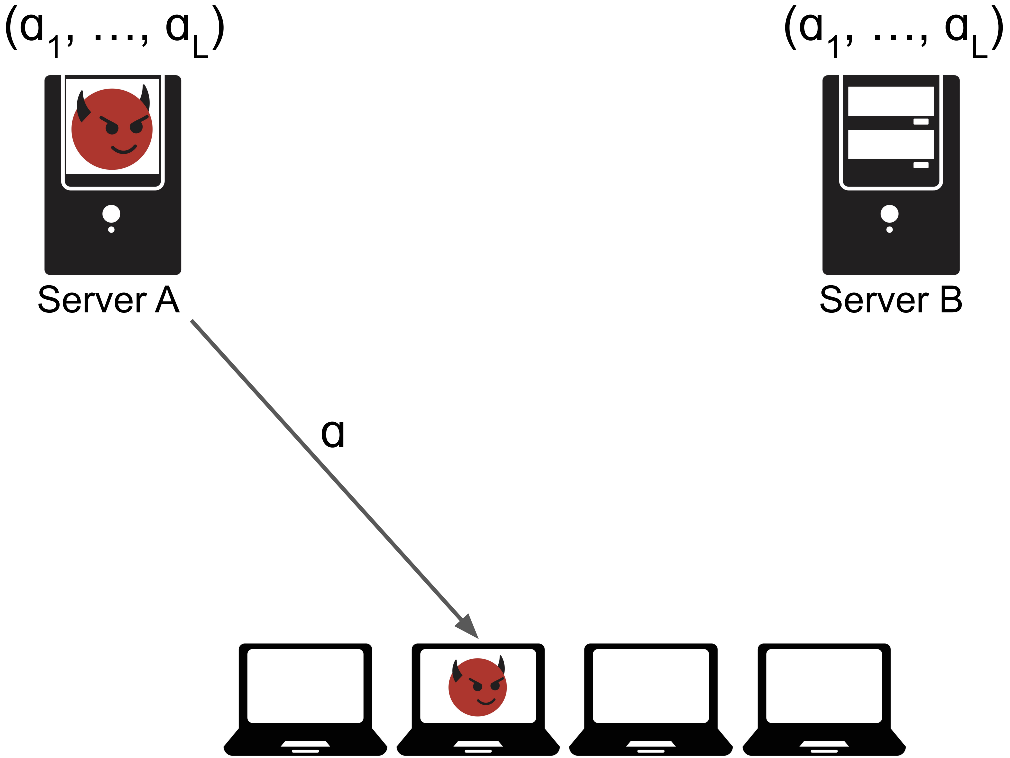
- For DPF, want to check: “every channel is 0 **OR** client knows the secret”
- MAC the message [CW79,WC81]
- Can compute *locally* on additive secret-shares
- Multiple channels: inner product

$$\left\langle (a_1, \ldots, a_L), ([\![m_1]\!]_1, \ldots, [\![m_L]\!]_1) \right\rangle - [\![t]\!]_1 + \left\langle (a_1, \ldots, a_L), ([\![m_1]\!]_2, \ldots, [\![m_L]\!]_2) \right\rangle - [\![t]\!]_2 = 0$$

Blind message authentication



Problem: Client-server collusion



Blind message authentication

- For DPF, want to check: “every channel is 0 **OR** client knows the secret”
- MAC the message [CW79,WC81]
- Can compute *locally* on additive secret-shares
- Multiple channels: inner product

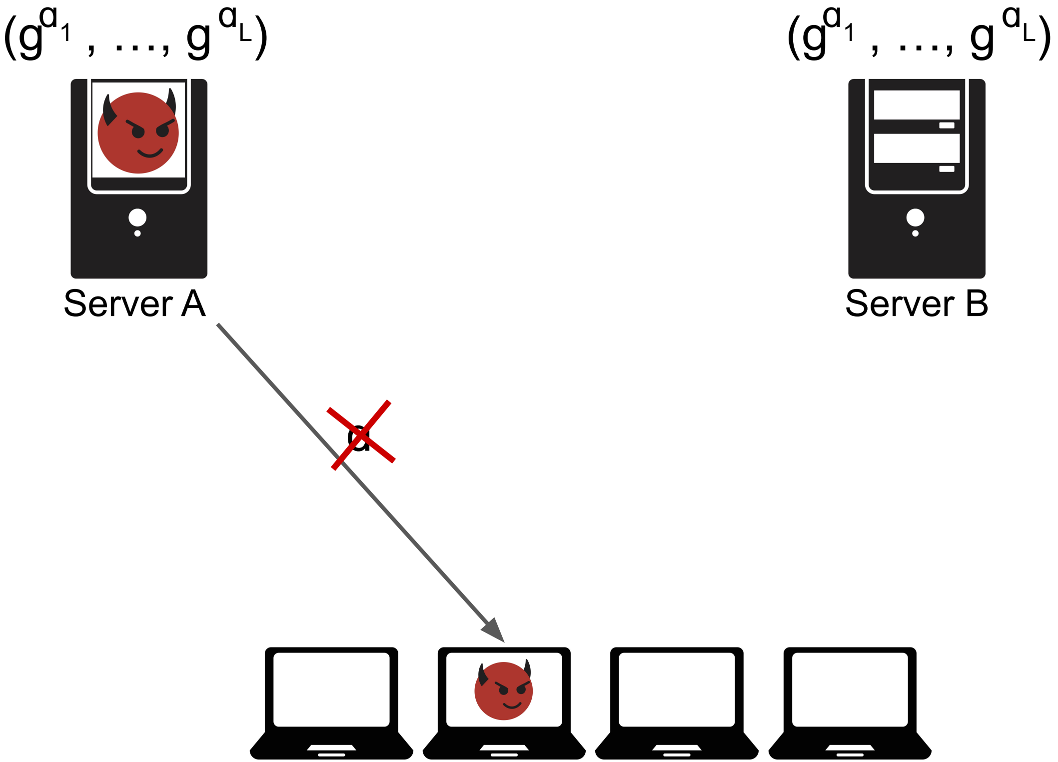
$$\Big\langle (a_1, \ldots, a_L), ([\![m_1]\!]_1, \ldots, [\![m_L]\!]_1) \Big\rangle - [\![t]\!]_1$$
$$+ \qquad\qquad\qquad = \quad 0$$
$$\Big\langle (a_1, \ldots, a_L), ([\![m_1]\!]_2, \ldots, [\![m_L]\!]_2) \Big\rangle - [\![t]\!]_2$$

Blind message authentication

- For DPF, want to check: “every channel is 0 **OR** client knows the secret”
- MAC the message [CW79,WC81]
- Can compute *locally* on additive secret-shares
- Multiple channels: inner product
- In the exponent: hide keys from servers!

$$g^{\langle (\alpha_1, \ldots, \alpha_L), ([\![m_1]\!]_1, \ldots, [\![m_L]\!]_1) \rangle - [\![t]\!]_1} \cdot g^{\langle (\alpha_1, \ldots, \alpha_L), ([\![m_1]\!]_2, \ldots, [\![m_L]\!]_2) \rangle - [\![t]\!]_2} = g^0$$

Client-server collusion

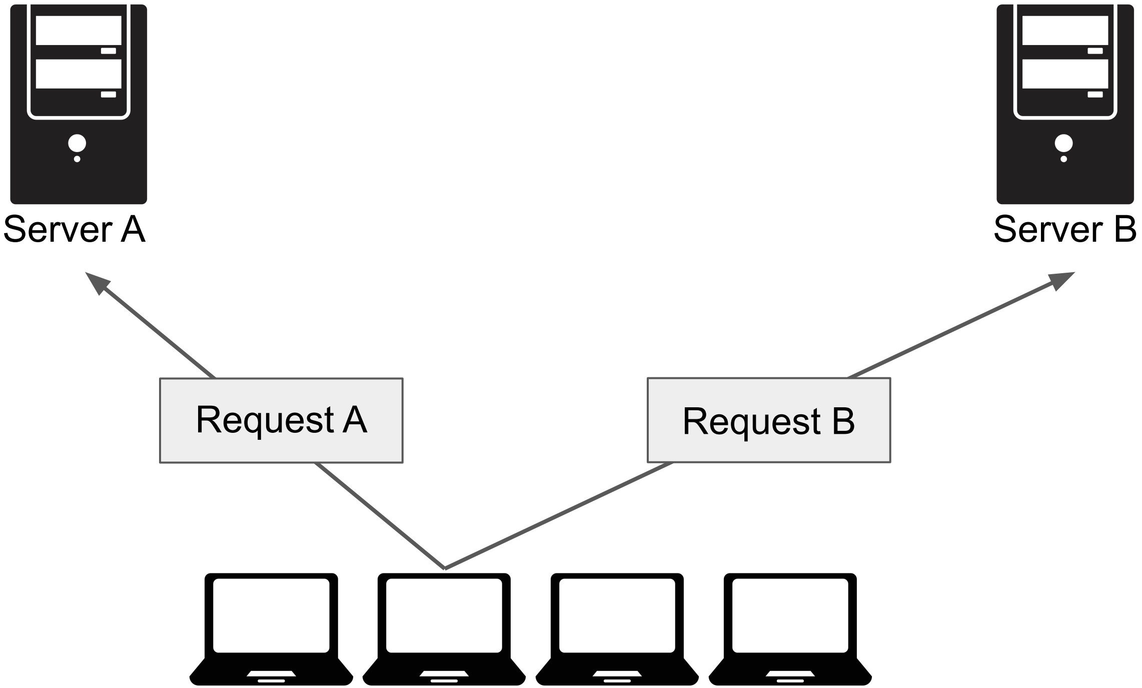


Deanonymization Attack

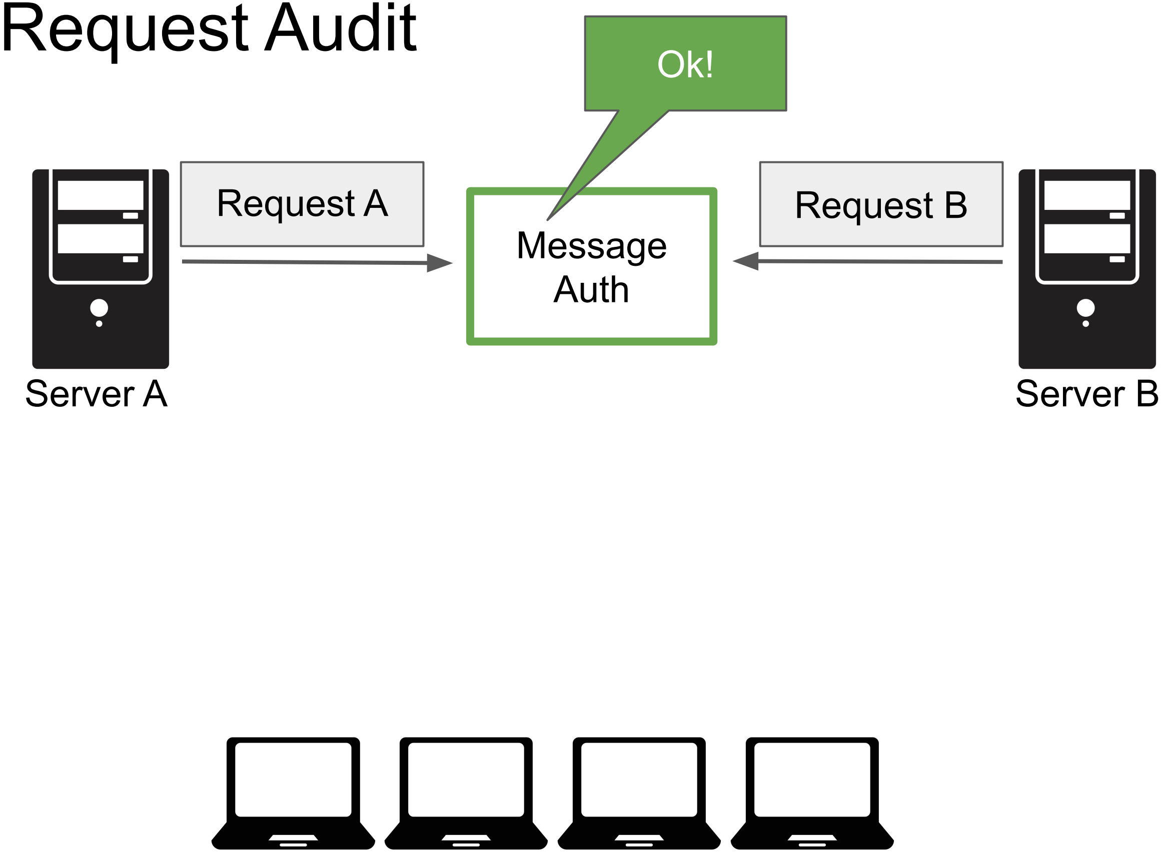
Availability vs. Privacy

- In anonymous broadcast Availability ⇒ Privacy
  - Denying a client removes them from the anonymity set
- Prior work: disruptive clients ignored
  - Can't have availability and privacy at the same time!
  - Makes Riposte / Spectrum susceptible to an “audit attack”
  - A malicious server can artificially shrink the anonymity set

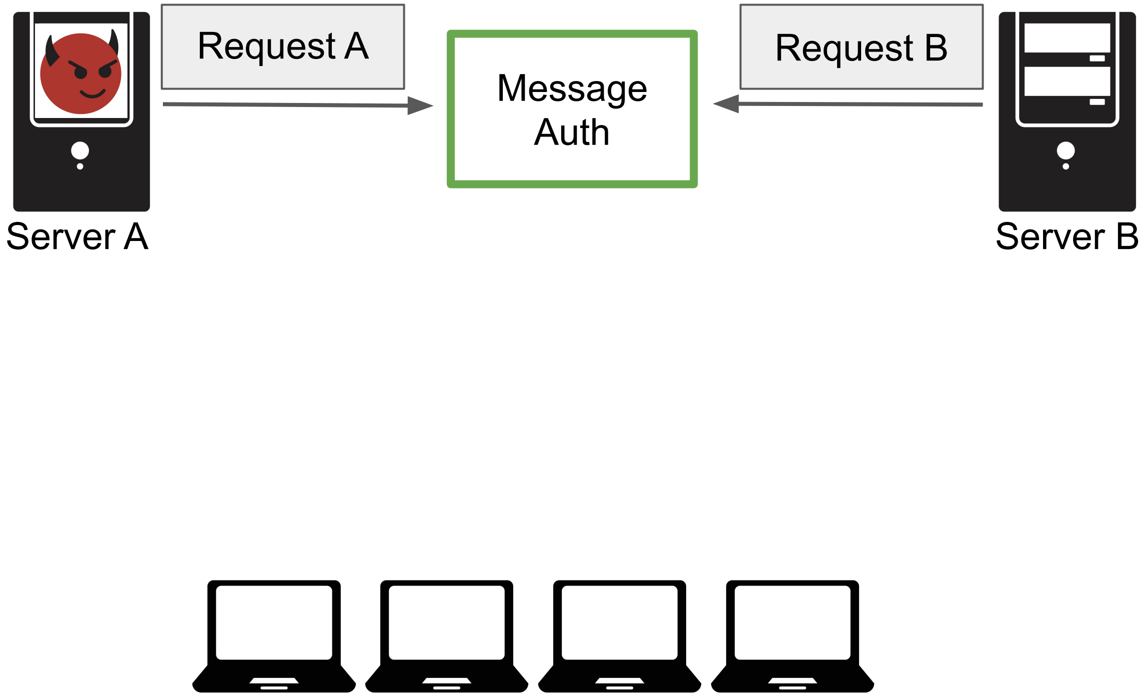
Recap: Request Audit



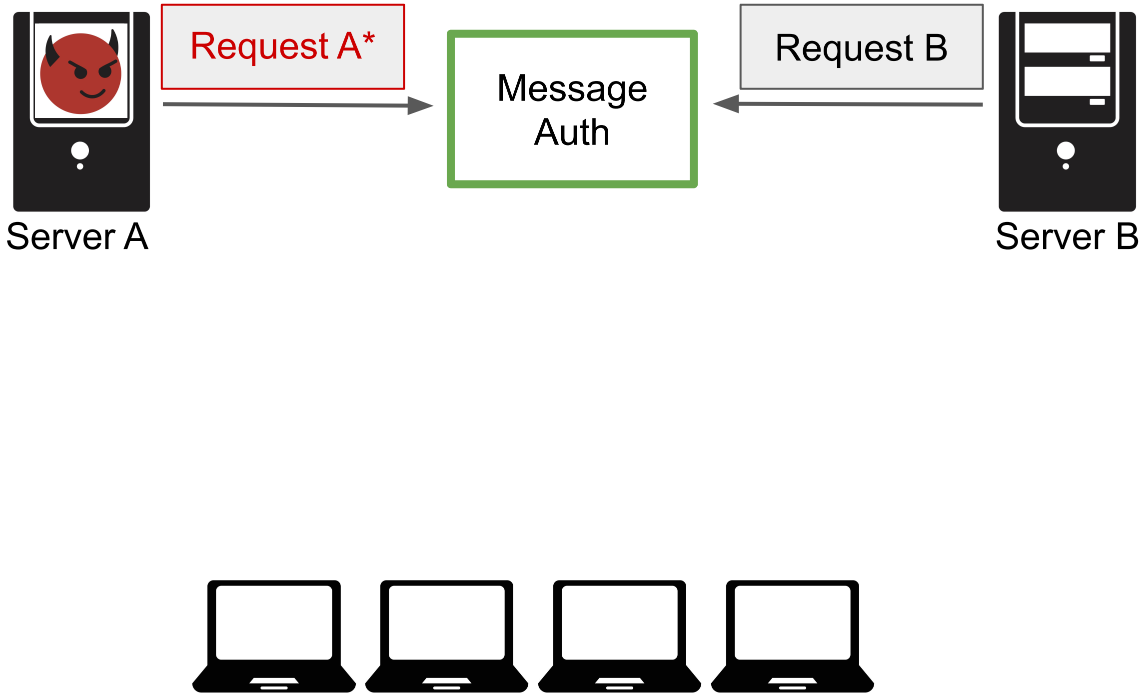
Recap: Request Audit



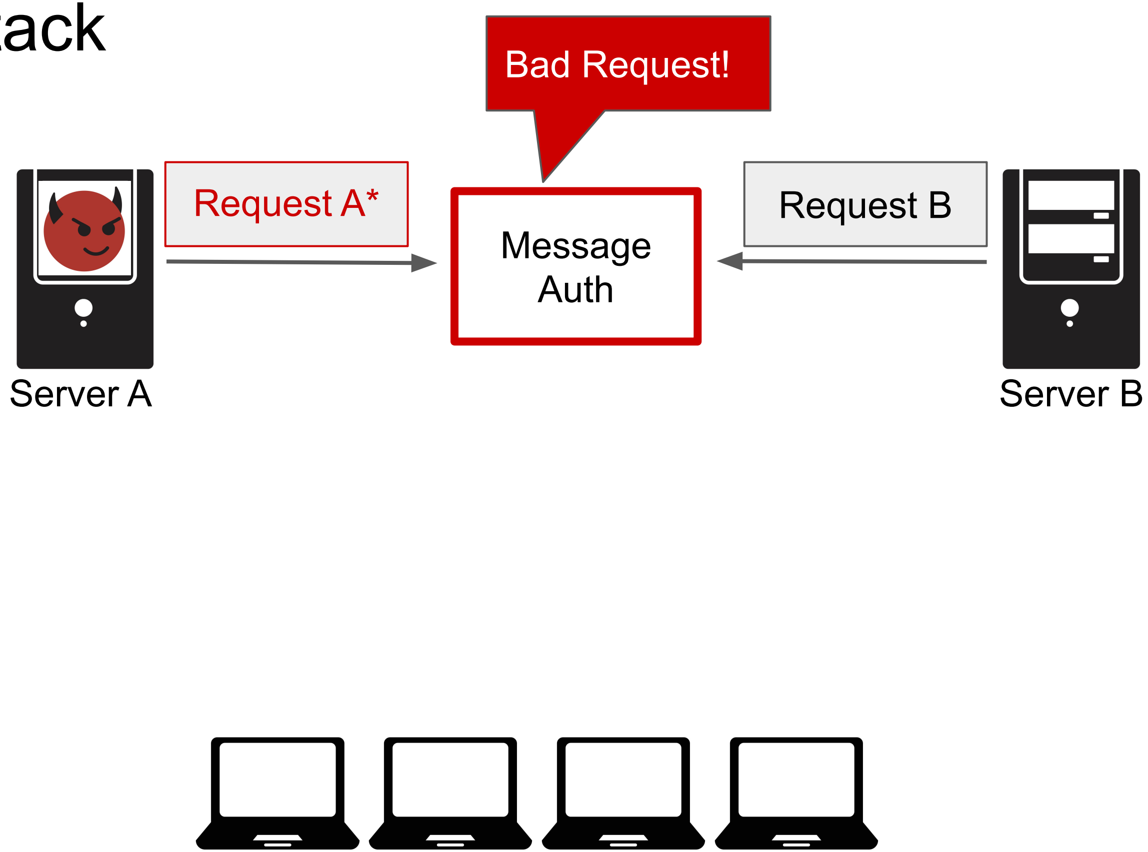
Audit Attack



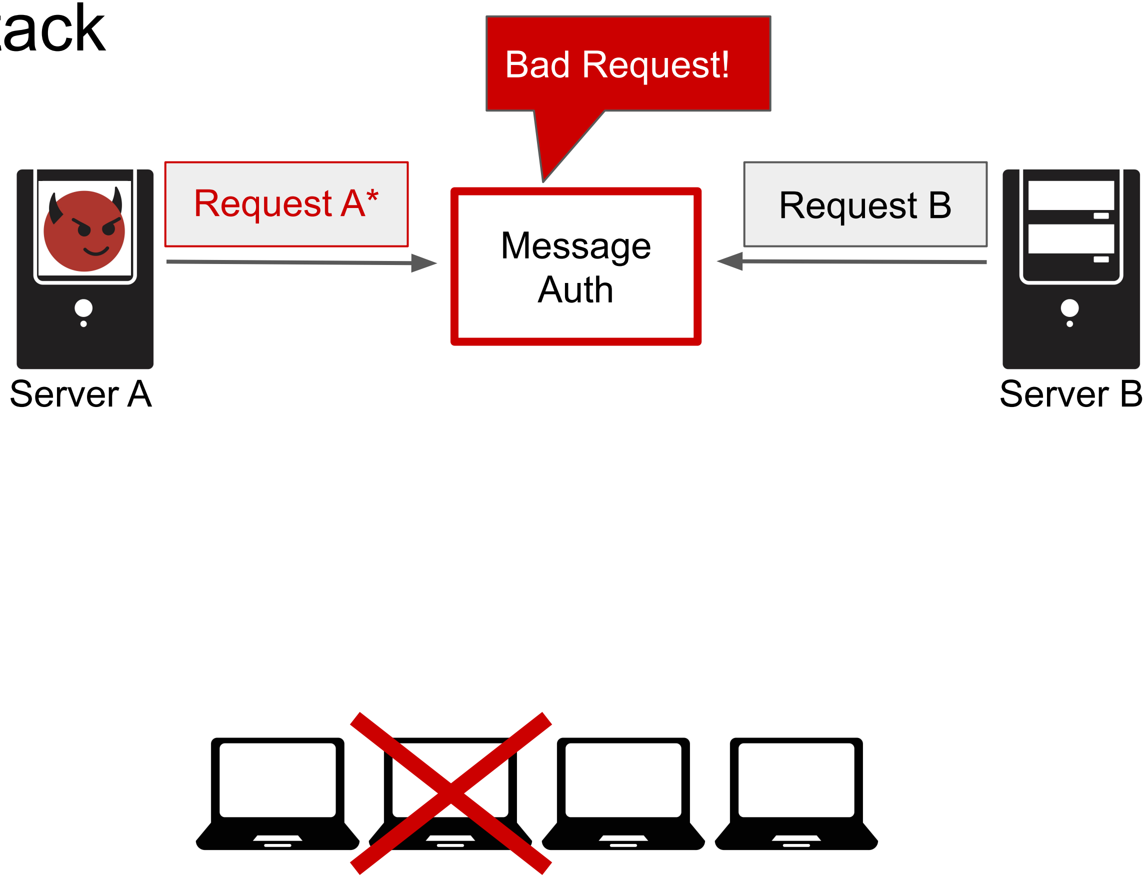
Audit Attack



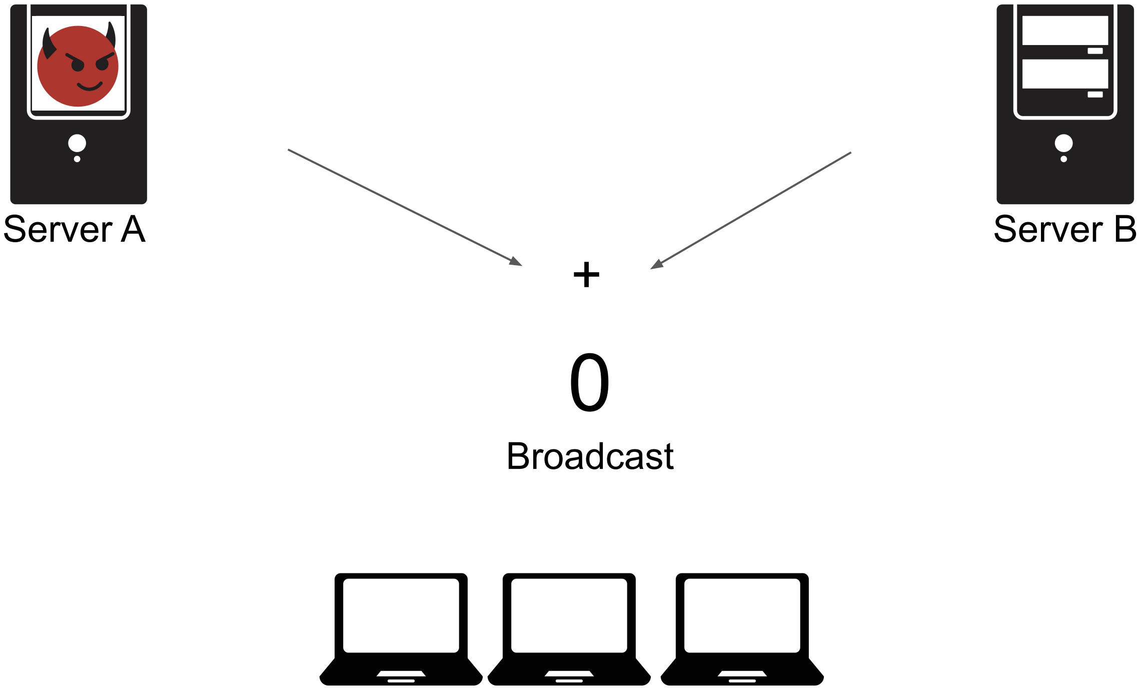
Audit Attack



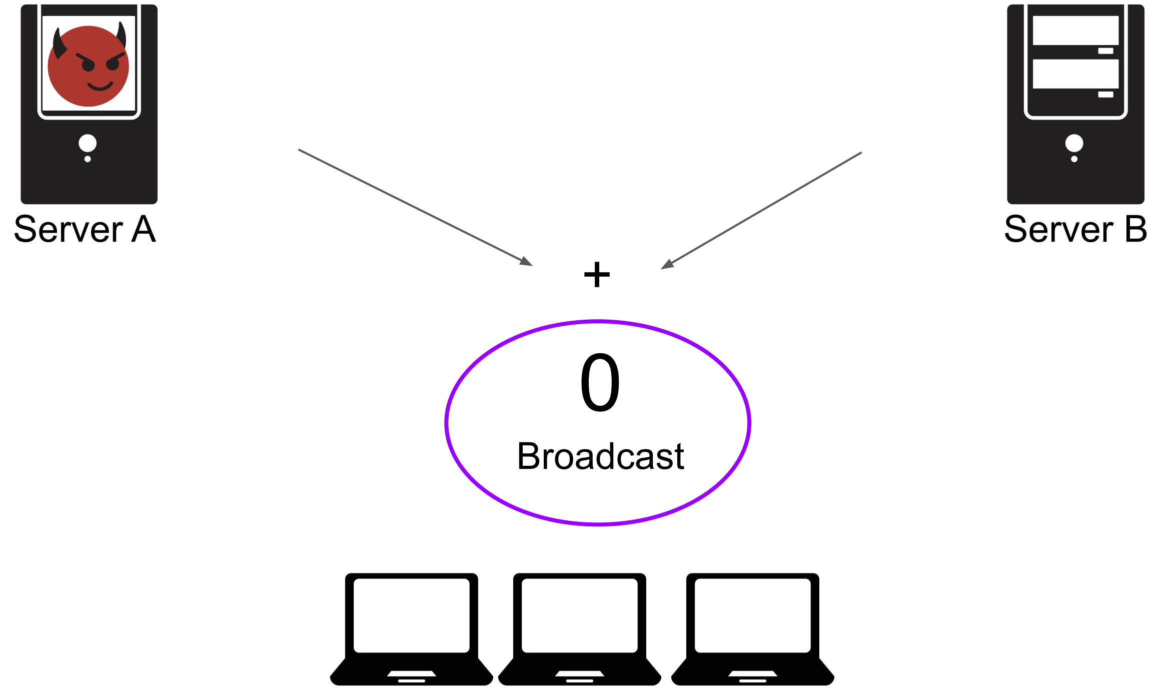
Audit Attack



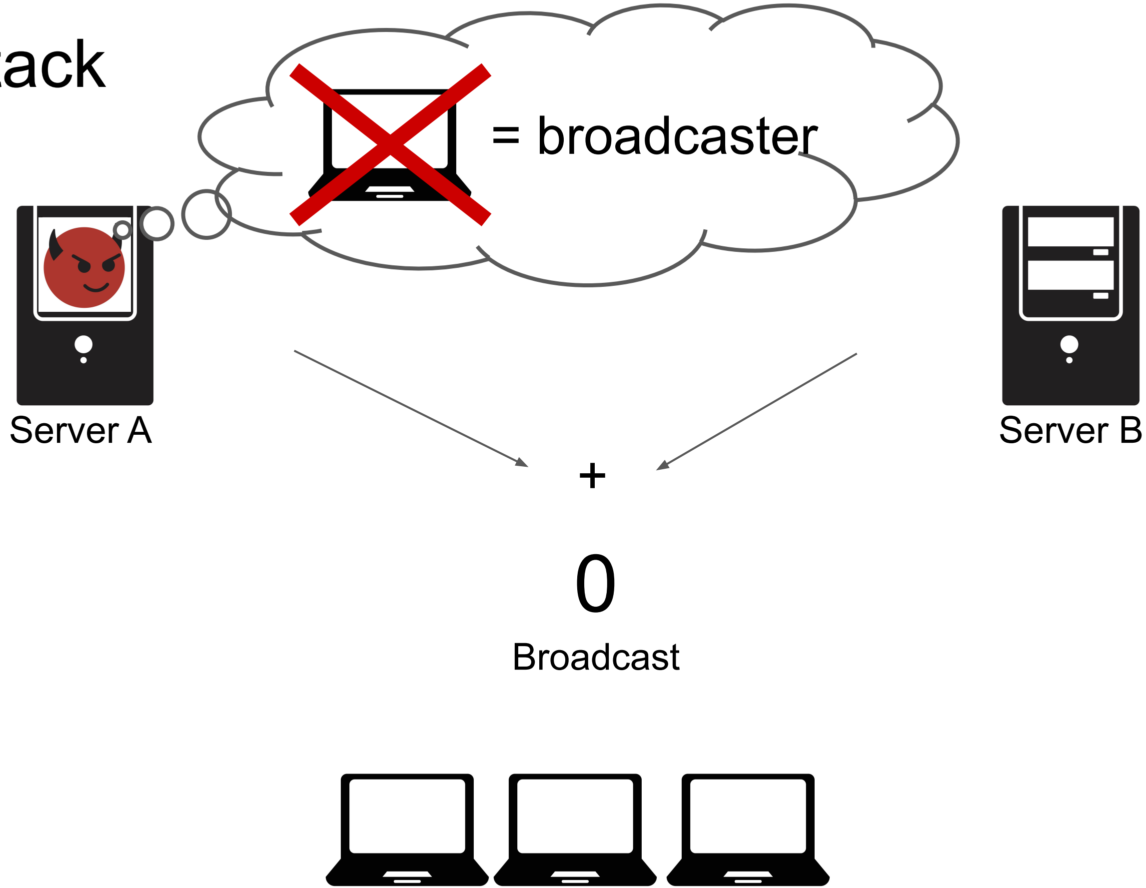
Audit Attack



Audit Attack



Audit Attack



Why this is challenging

Can’t trust any client

- client could be attempting disruption

Can’t trust any server

- “other” server could have caused the failure

BlameGame: Stopping the Audit Attack

Who’s cheating? Is it the client? Is it the server?

Want: abort Spectrum if a server is malicious.

BlameGame: Stopping the Audit Attack

Key idea: “commit” each server to its (private) audit input

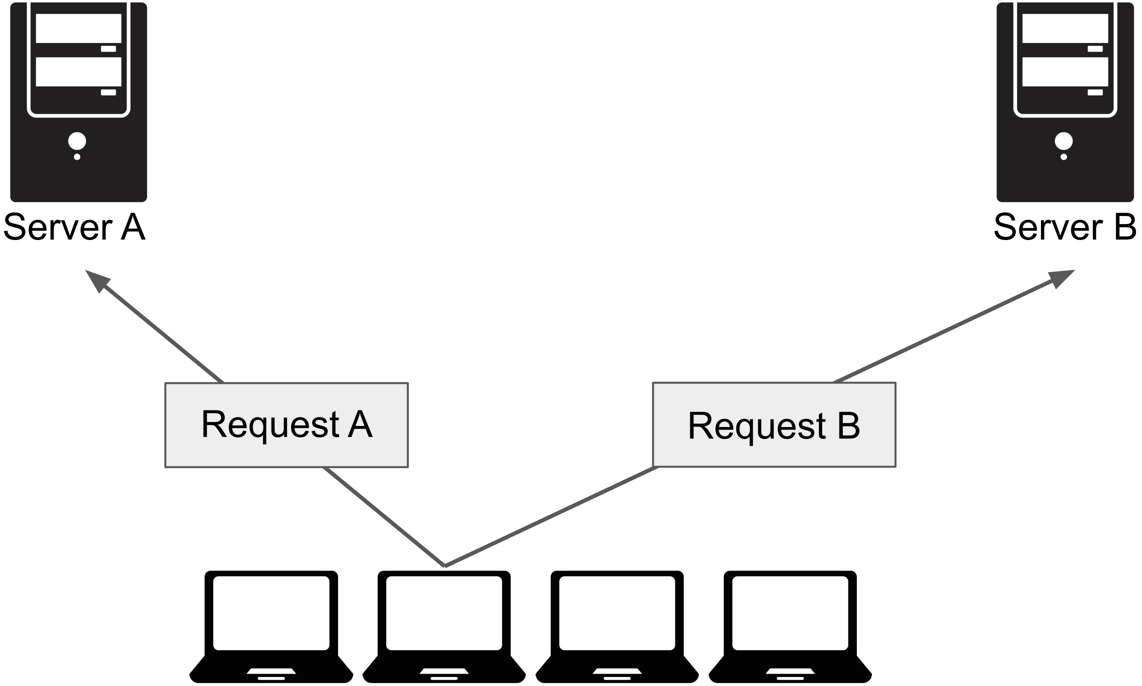
BlameGame: Stopping the Audit Attack

Key idea: “commit” each server to its (private) audit input

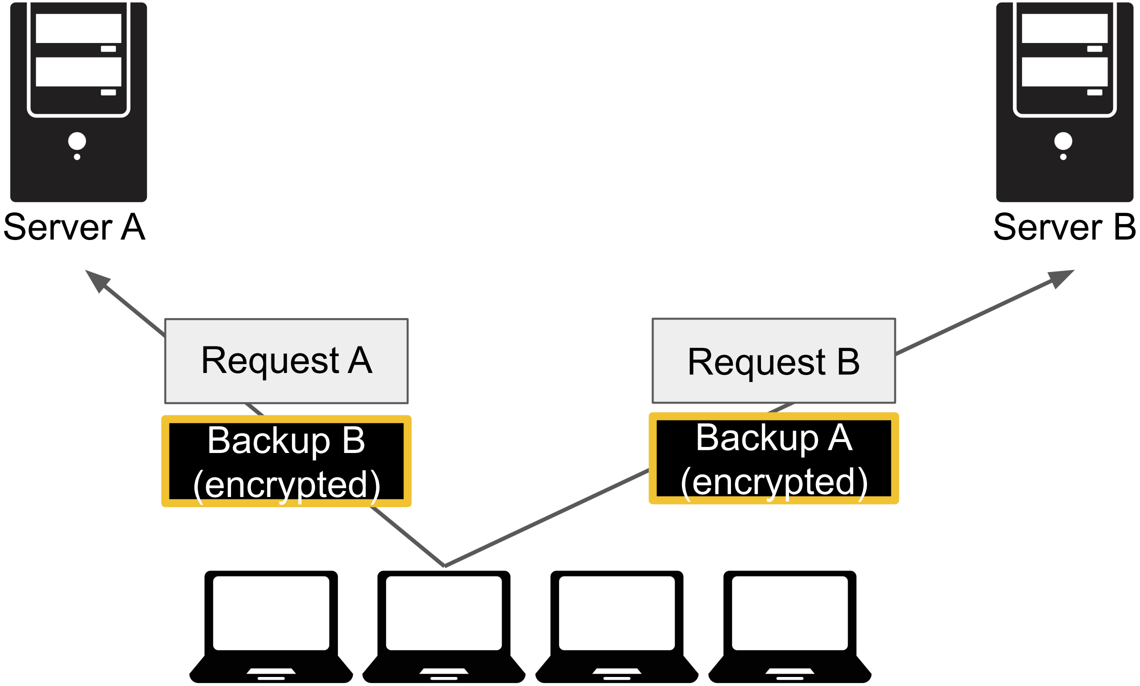
In a nutshell:

1. Clients send “backup” requests encrypted under other server’s public key
2. **If audit fails**: swap backups, decrypt, and try again
3. **If backup fails**: servers prove compliance
4. Blames malicious *server* or *client*

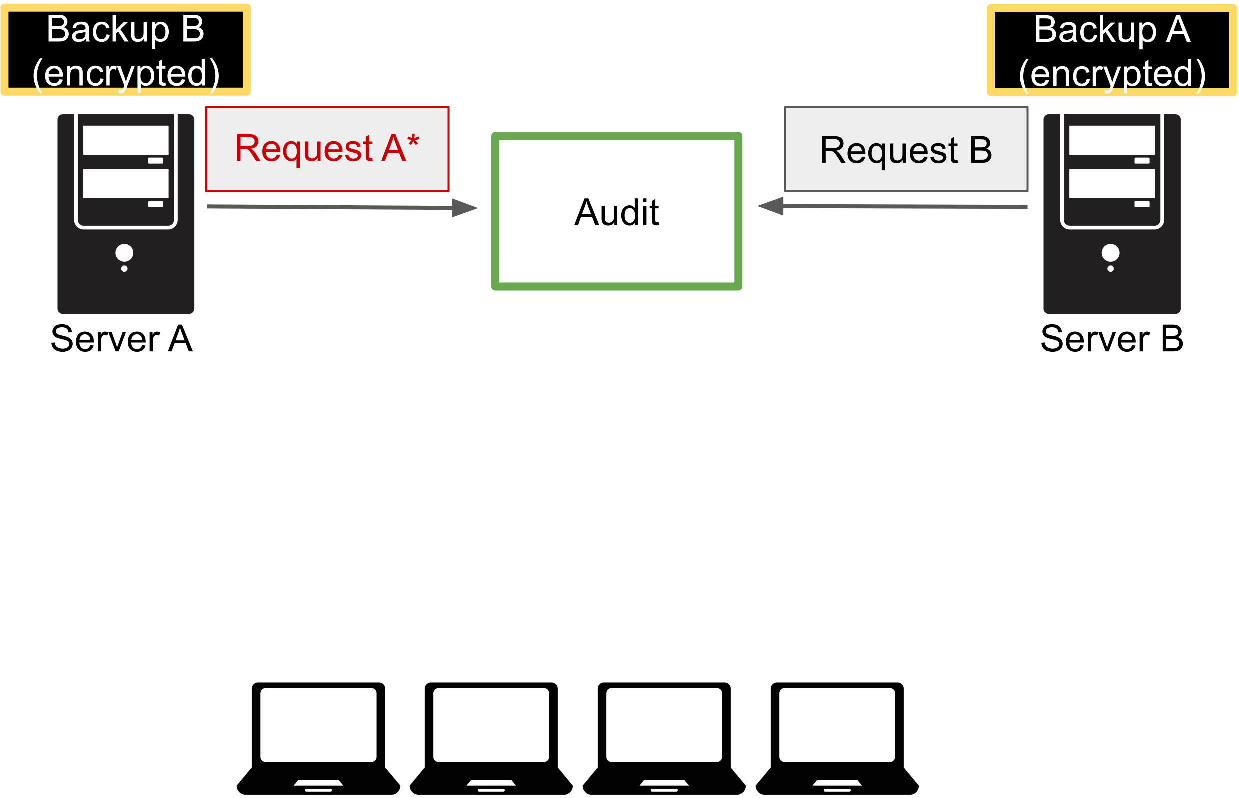
Step 1: Do regular request audit



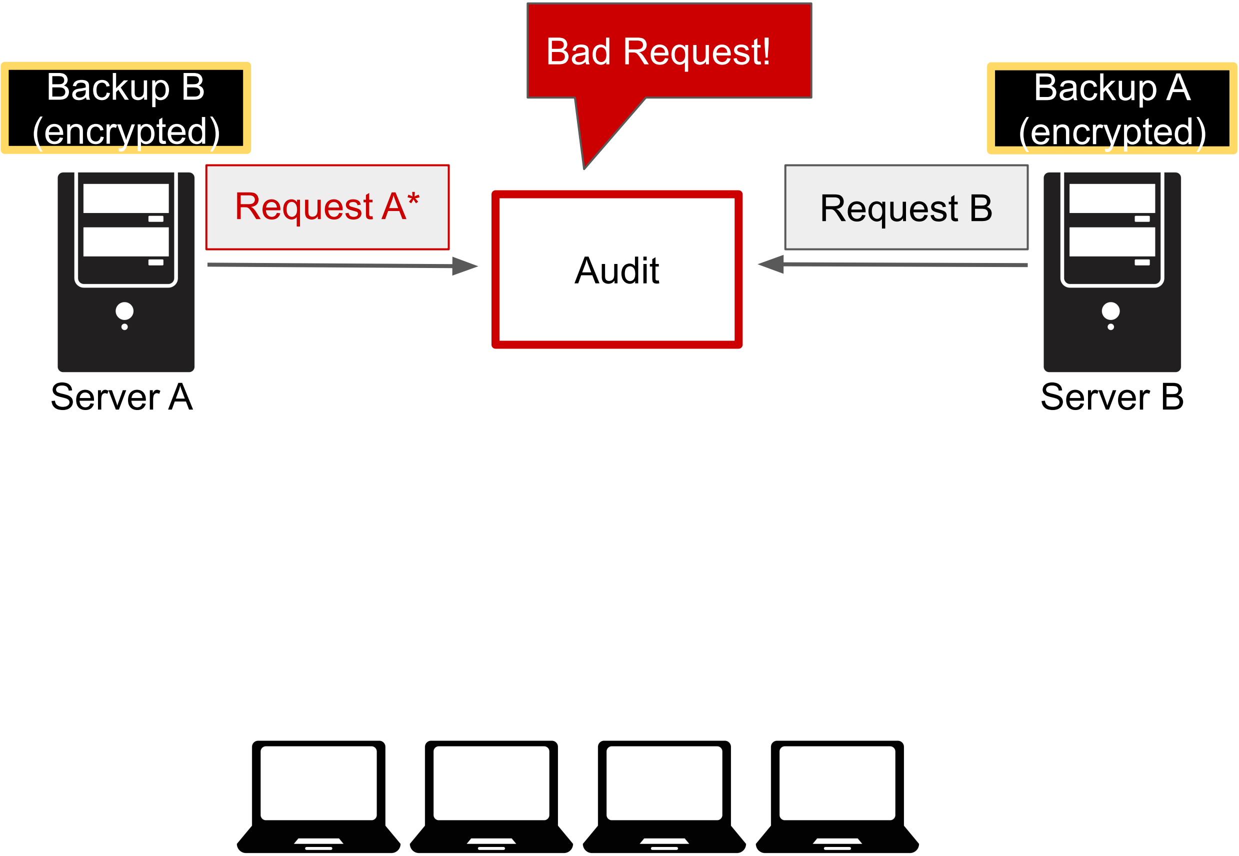
Step 1: Do regular request audit



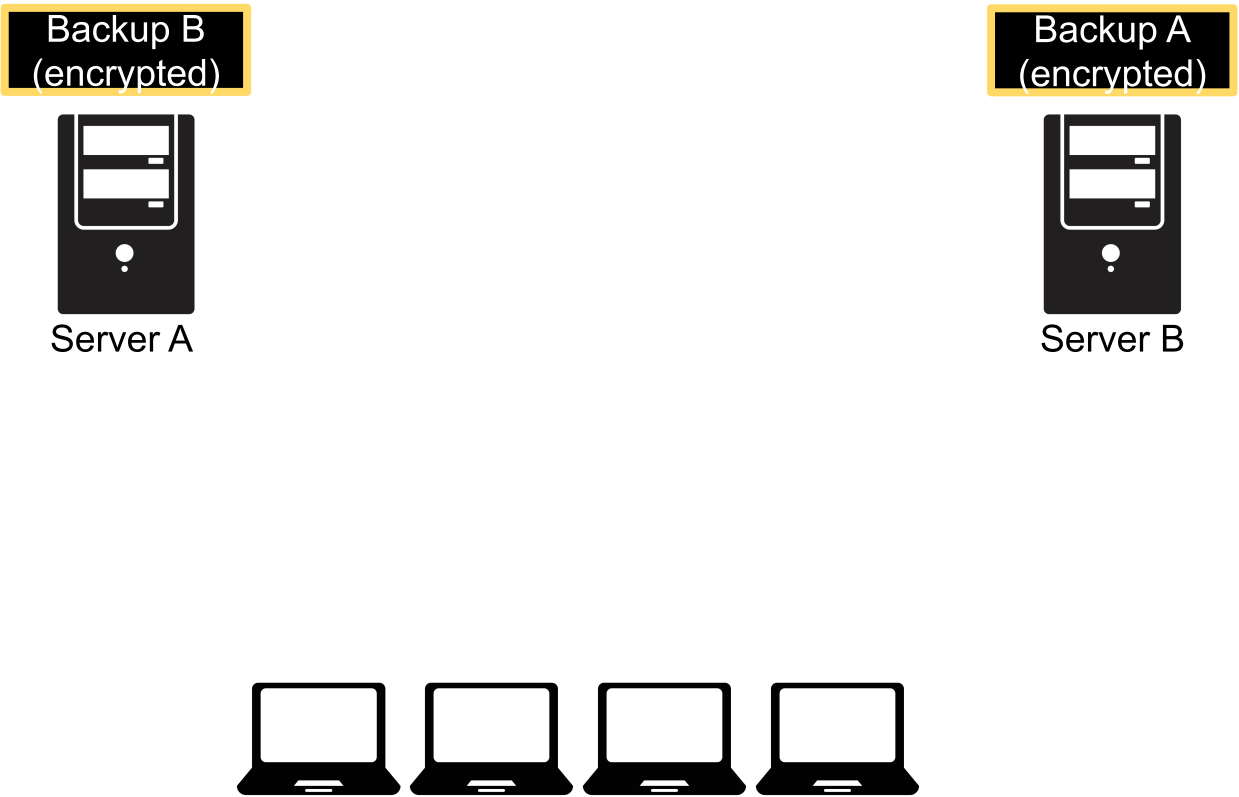
Step 1: Do regular request audit



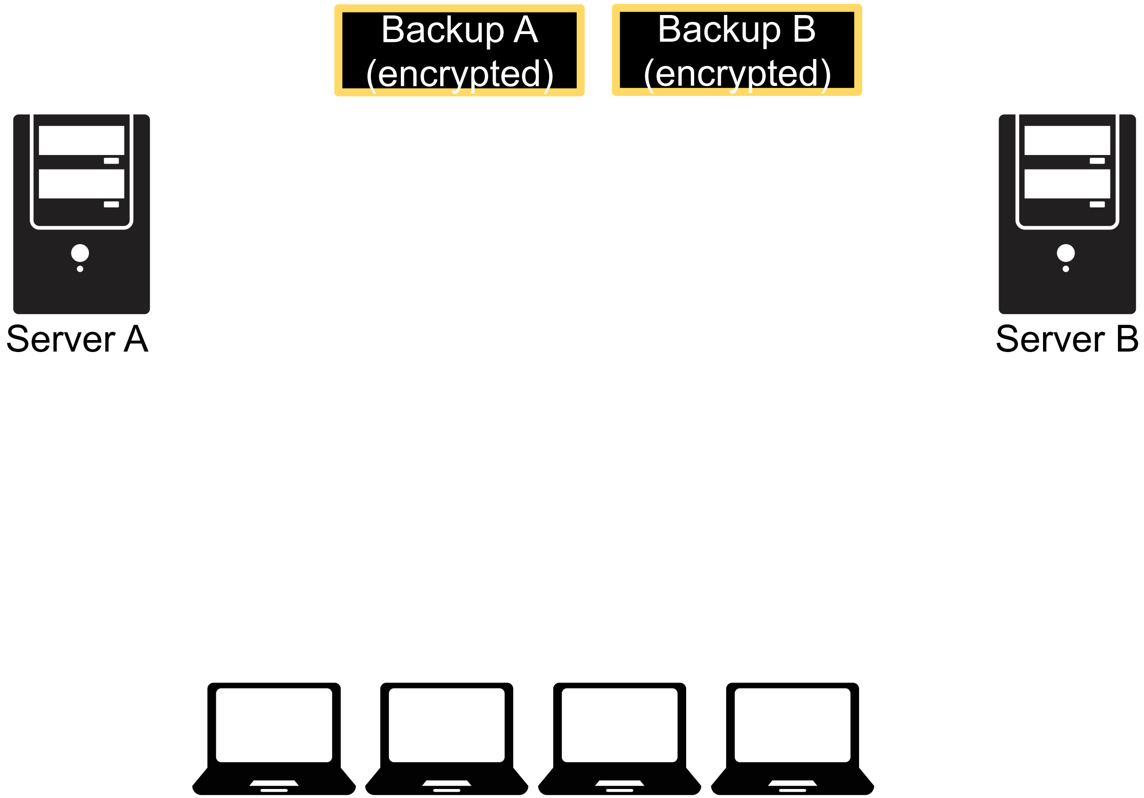
Step 1: Do regular request audit



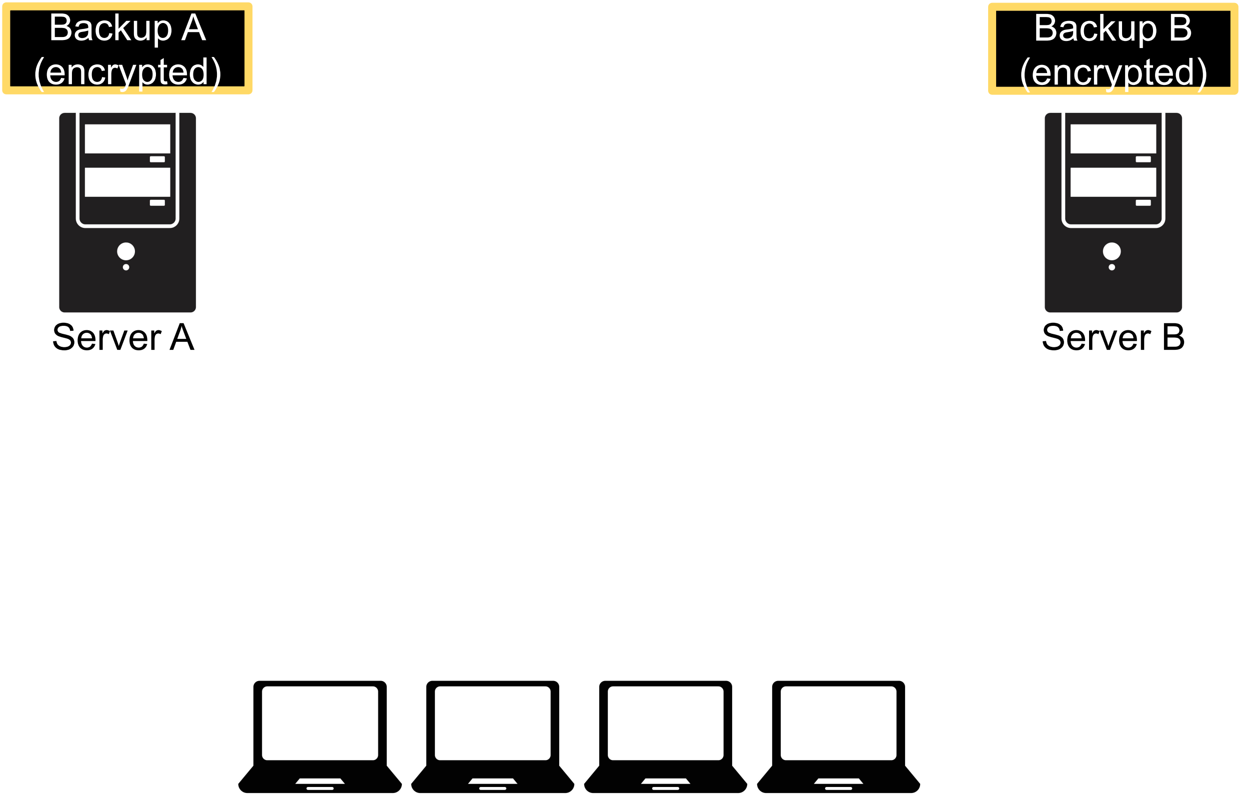
Step 2: Do backup audit



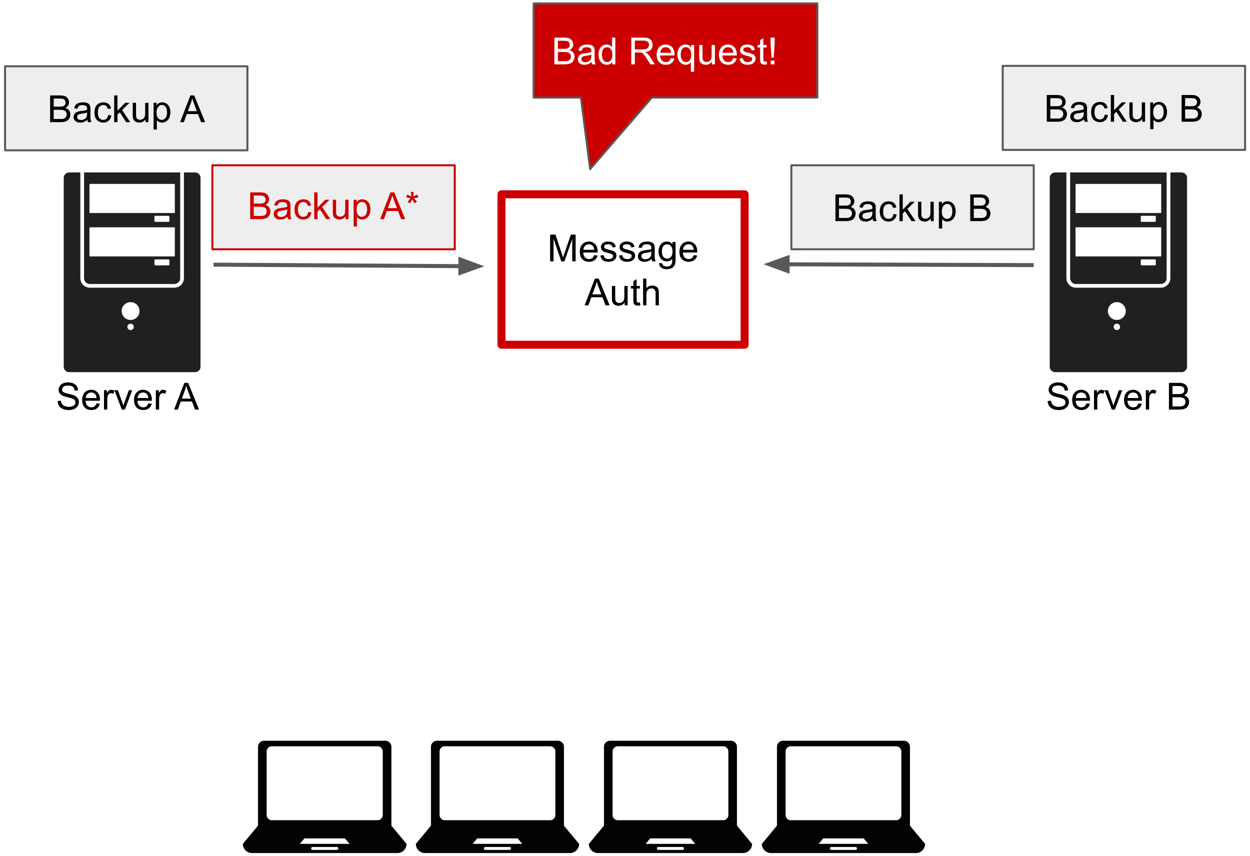
Step 2: Do backup audit



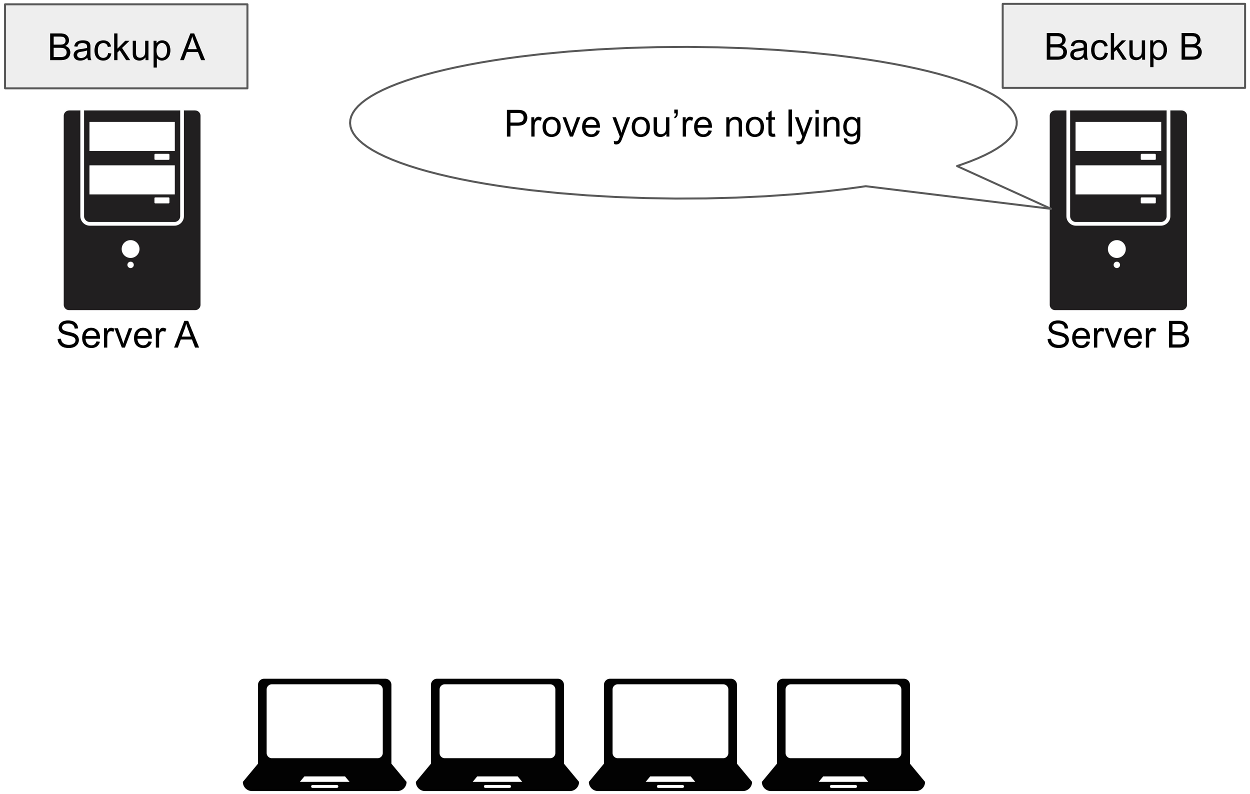
Step 2: Do backup audit



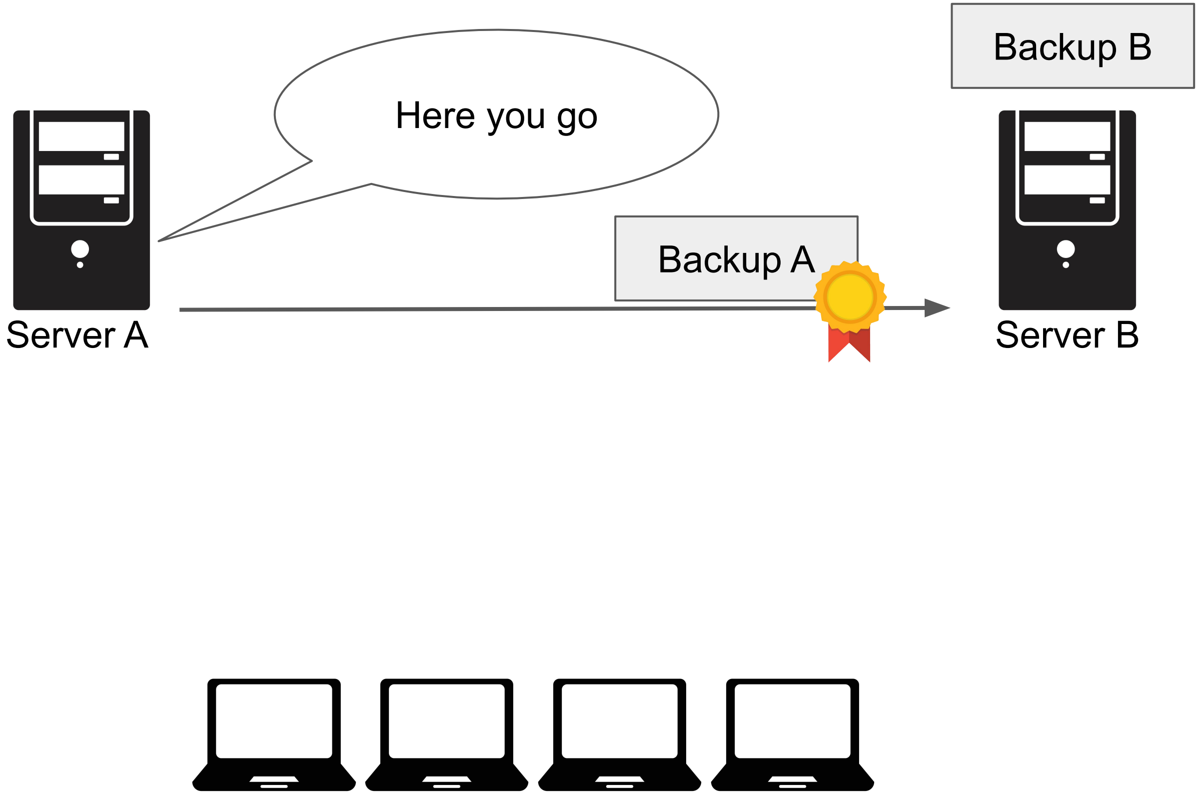
Step 2: Do backup audit



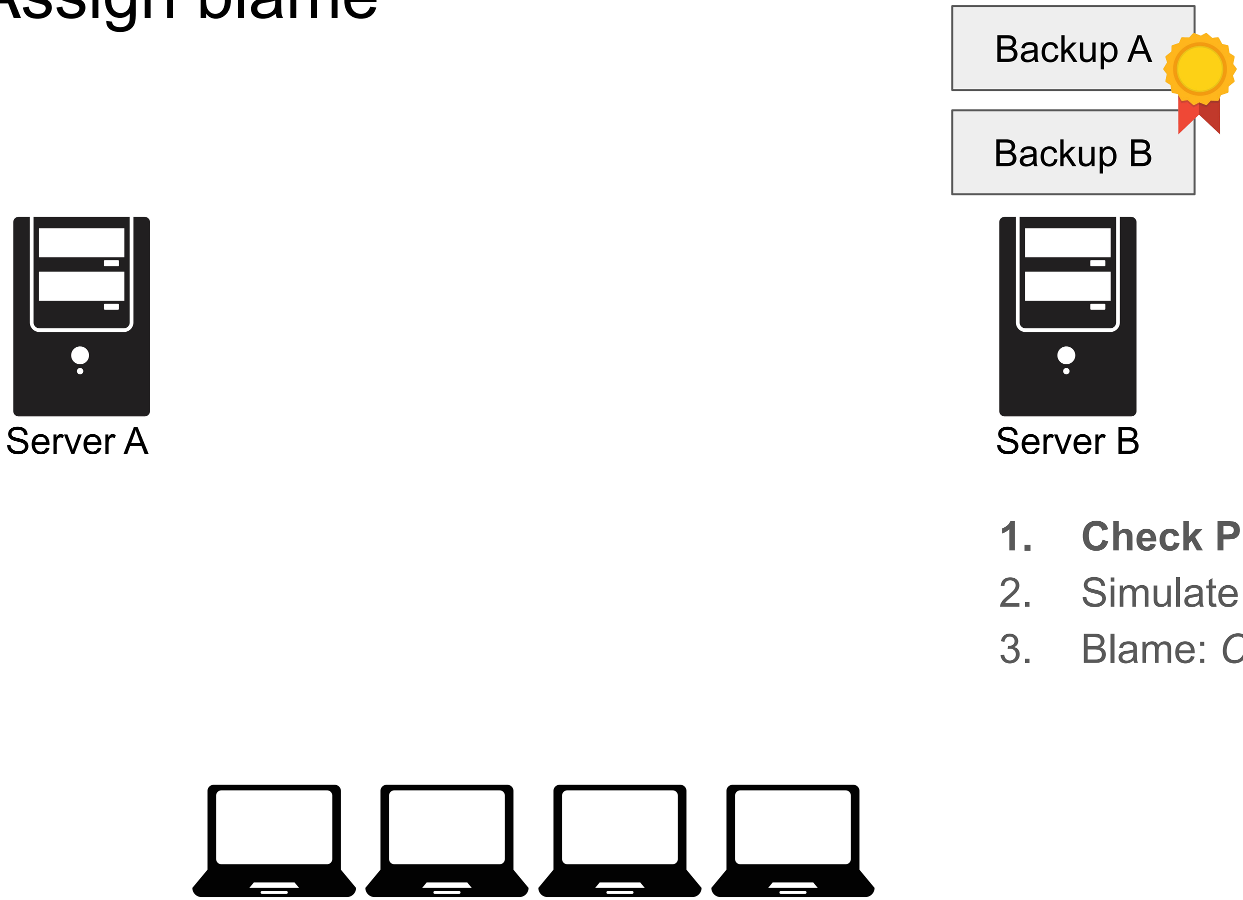
Step 3: Prove innocence



Step 3: Prove innocence

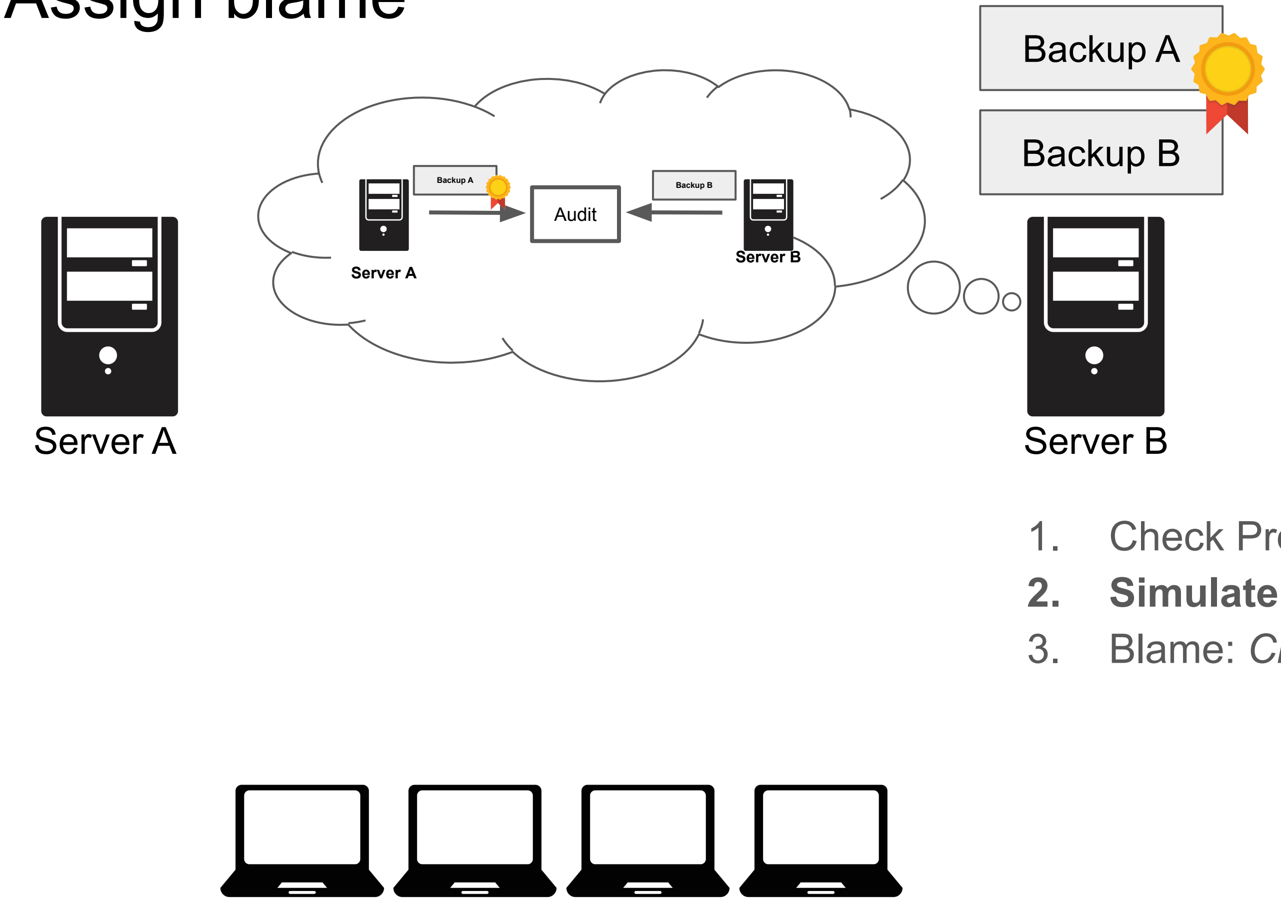


Step 4: Assign blame



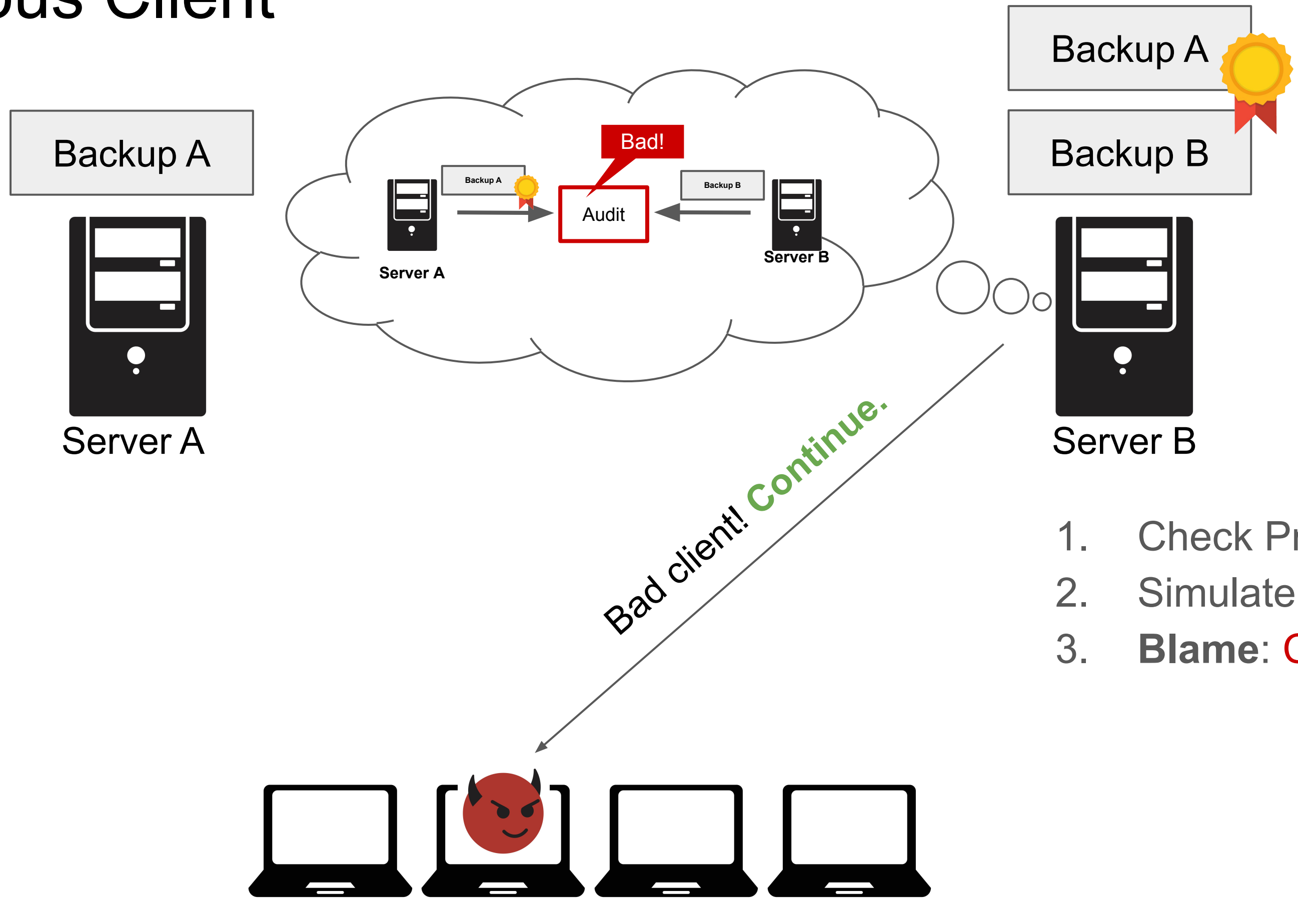
1. **Check Proof**
2. Simulate Audit
3. Blame: *Client* or *Server*

Step 4: Assign blame



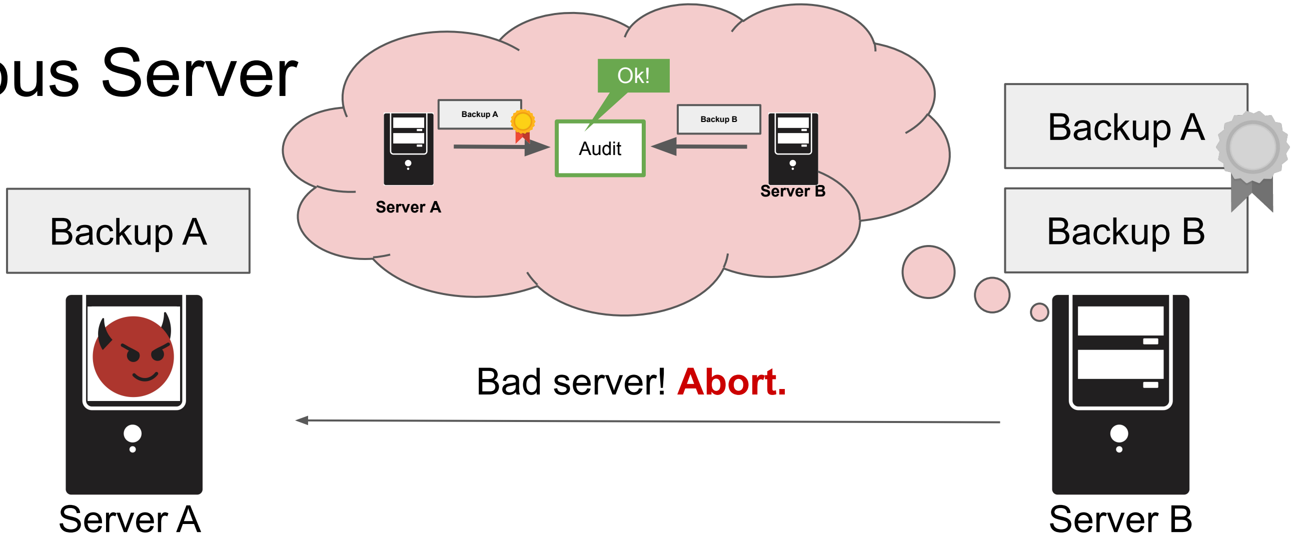
1. Check Proof
2. **Simulate Audit**
3. Blame: *Client* or *Server*

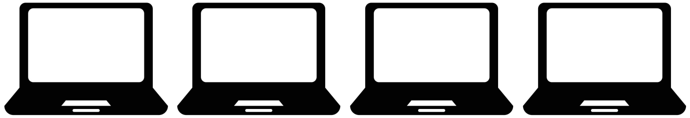
Malicious Client



1. Check Proof
2. Simulate Audit
3. **Blame**: Client or Server

Malicious Server





1. Check Proof
2. Simulate Audit
3. **Blame**: Client or Server

BlameGame in practice

- Now only one shot at de-anonymization before Spectrum is aborted
- Only invoked following a failed audit of a request (few in practice)
- Backup request size independent of message length
- Also applies to other anonymous broadcast protocols

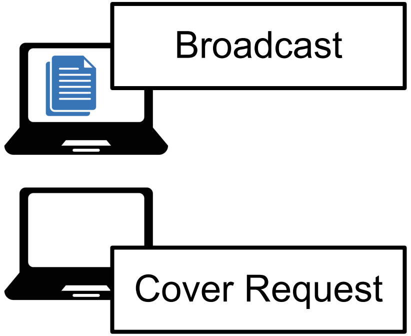
BlameGame in practice

- Now only one shot at de-anonymization before Spectrum is aborted
- Only invoked following a failed audit of a request (few in practice)
- **Backup request size independent of message length**
- Also applies to other anonymous broadcast protocols

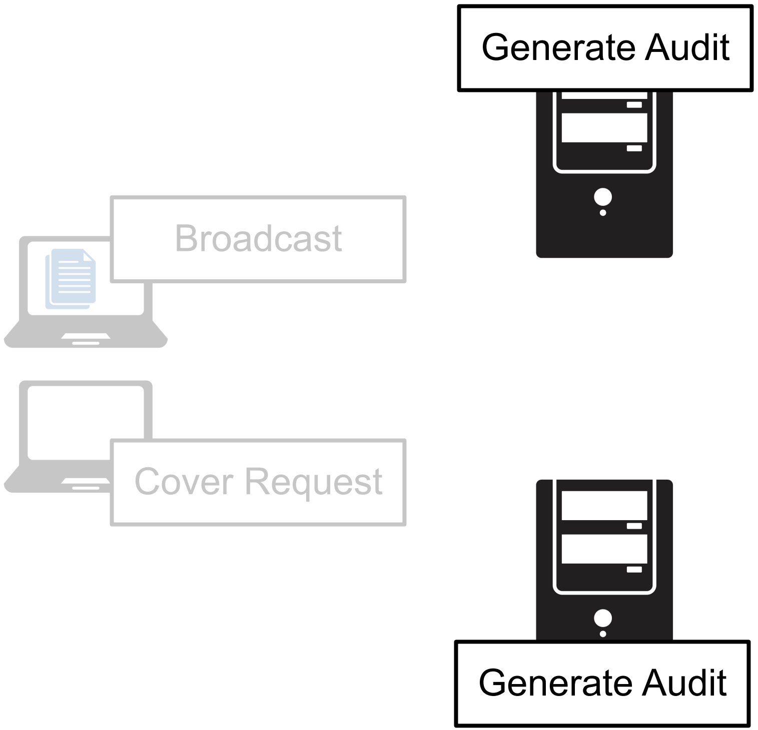
BlameGame (per failed audit)	**Backup Request** per client	**Audit** per client	**Decryption** once per client
	140 bytes	200 bytes	10 μs

Spectrum

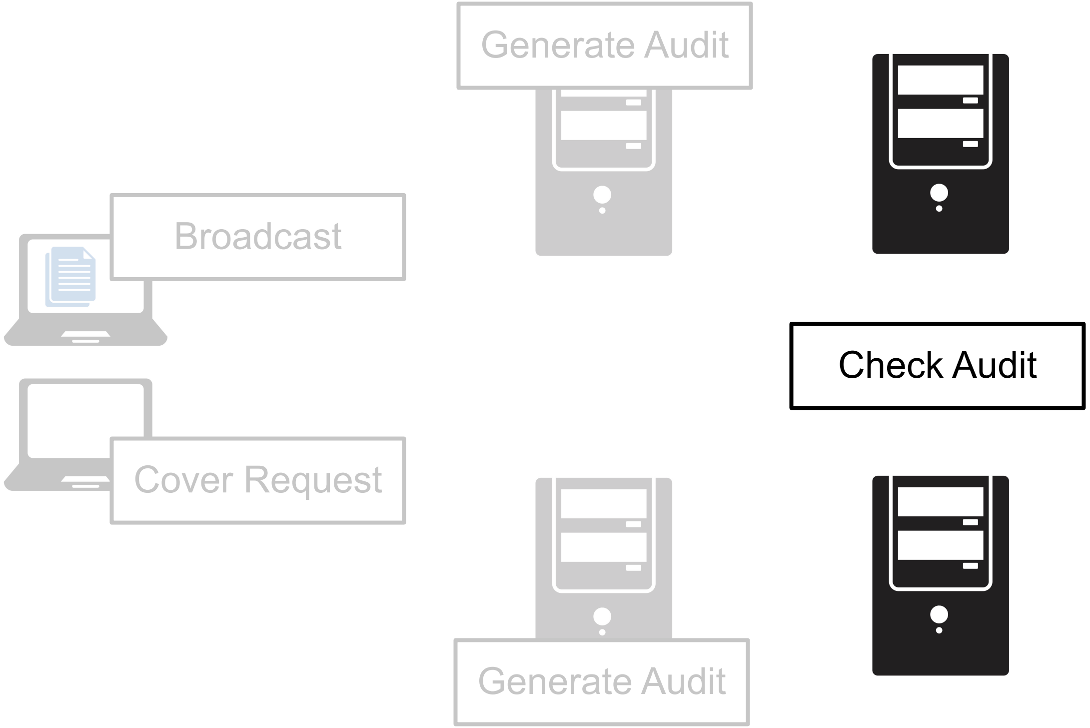
Putting it all together



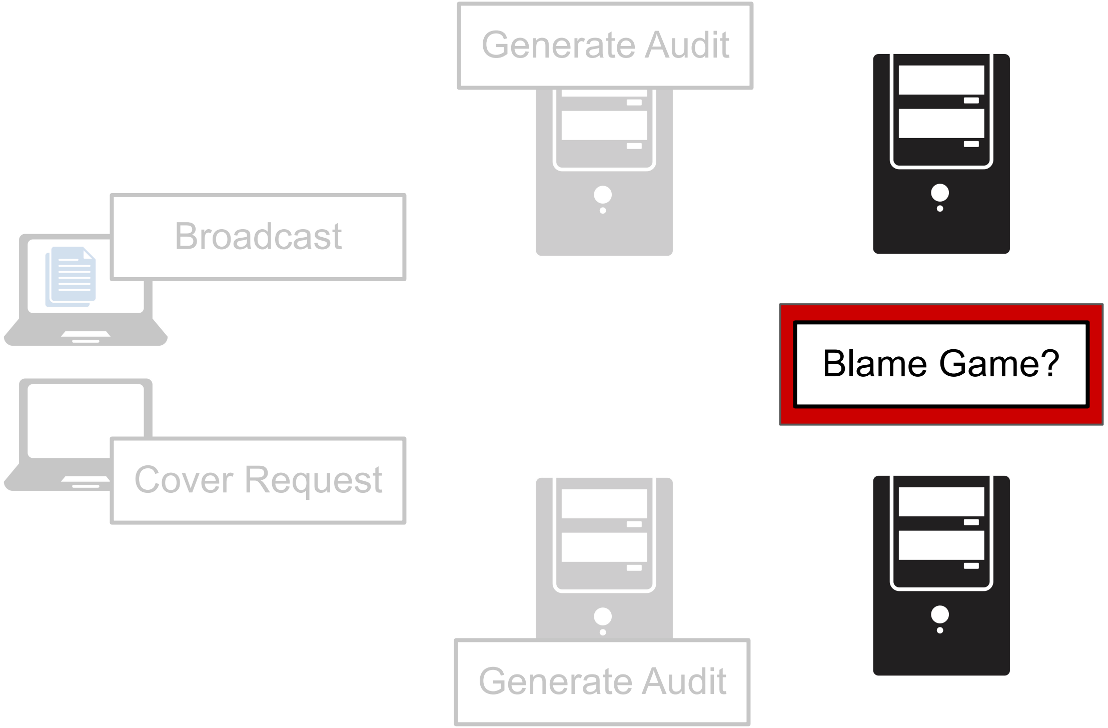
Putting it all together



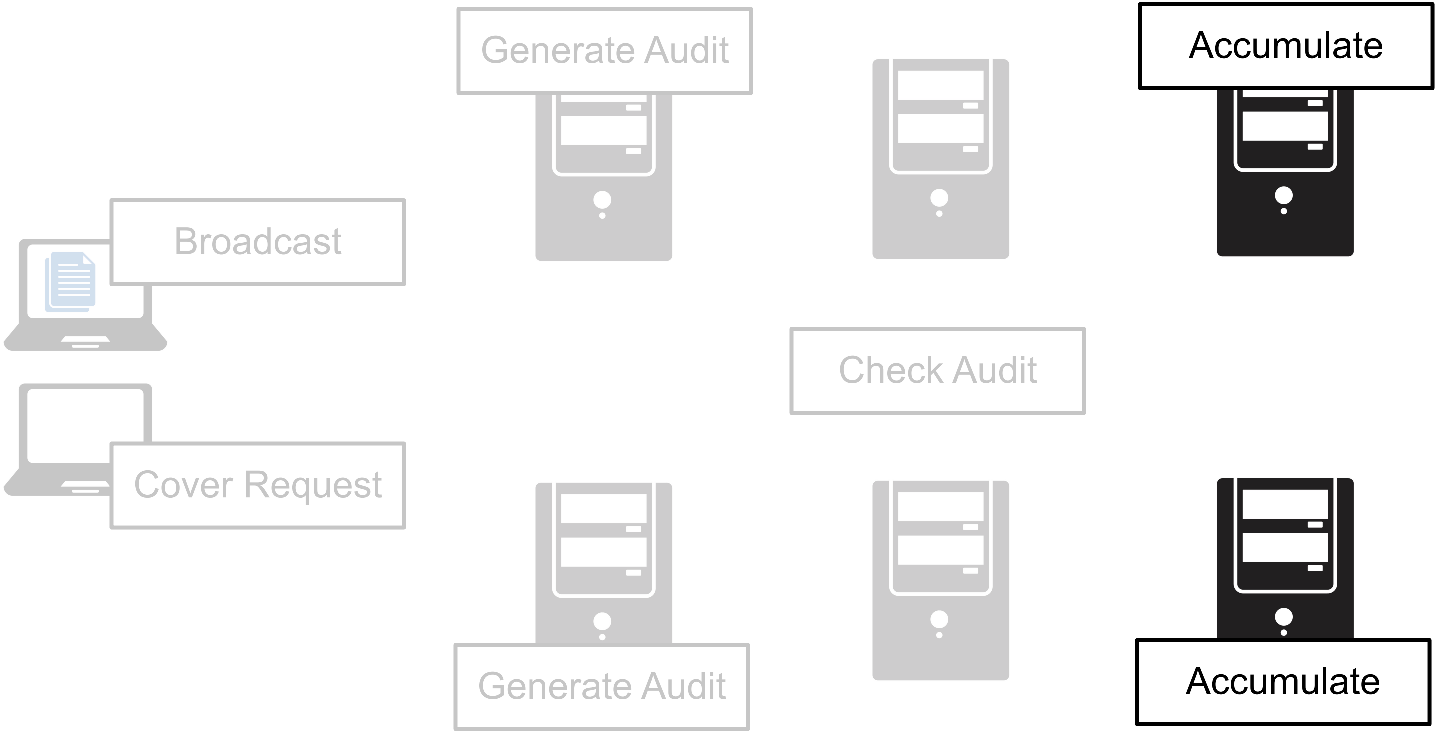
Putting it all together



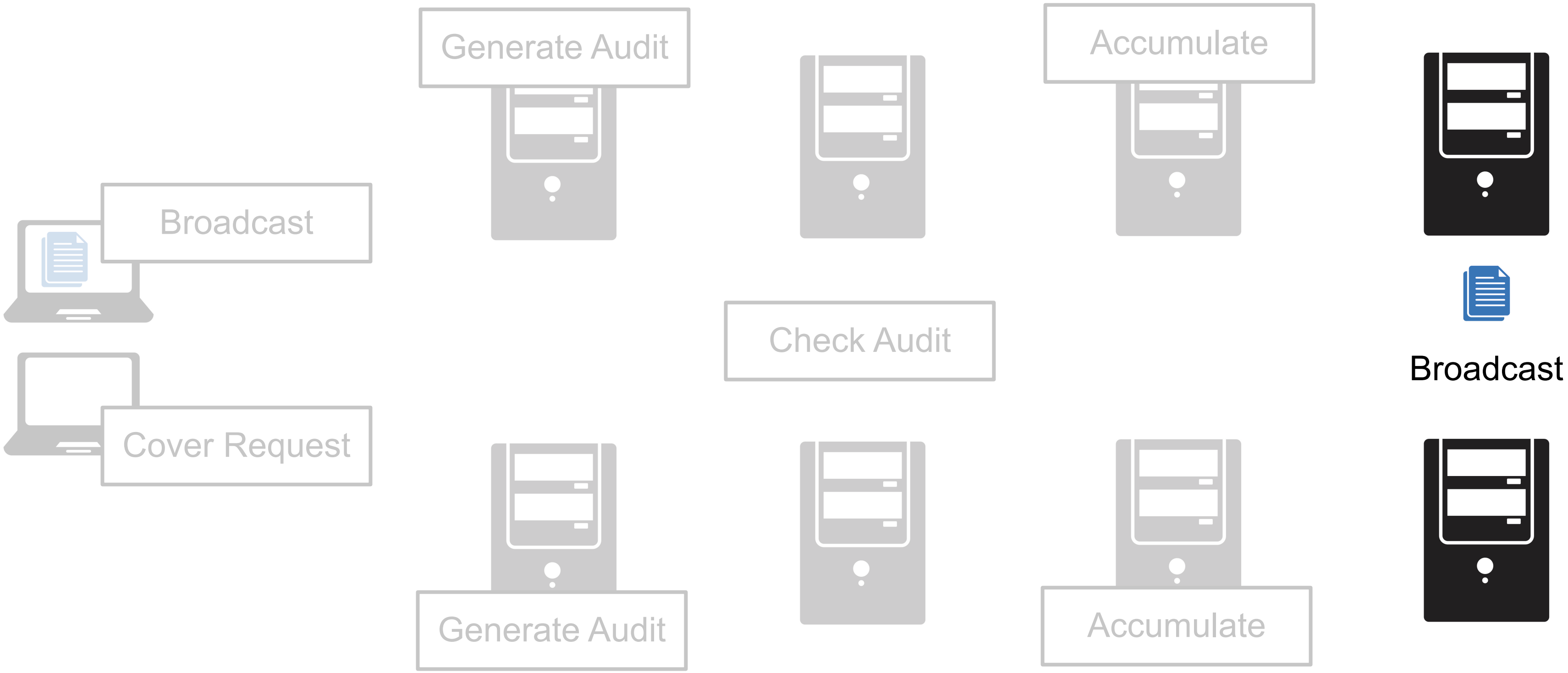
Putting it all together



Putting it all together



Putting it all together



Evaluation

Implementation

- GitHub: znewman01/spectrum-impl (written in Rust)
  - Compare with prior work: Riposte, Express, Blinder
- Terraform templates and scripts for reproducing experiments
  - Including prior work

README.md

spectrum-impl

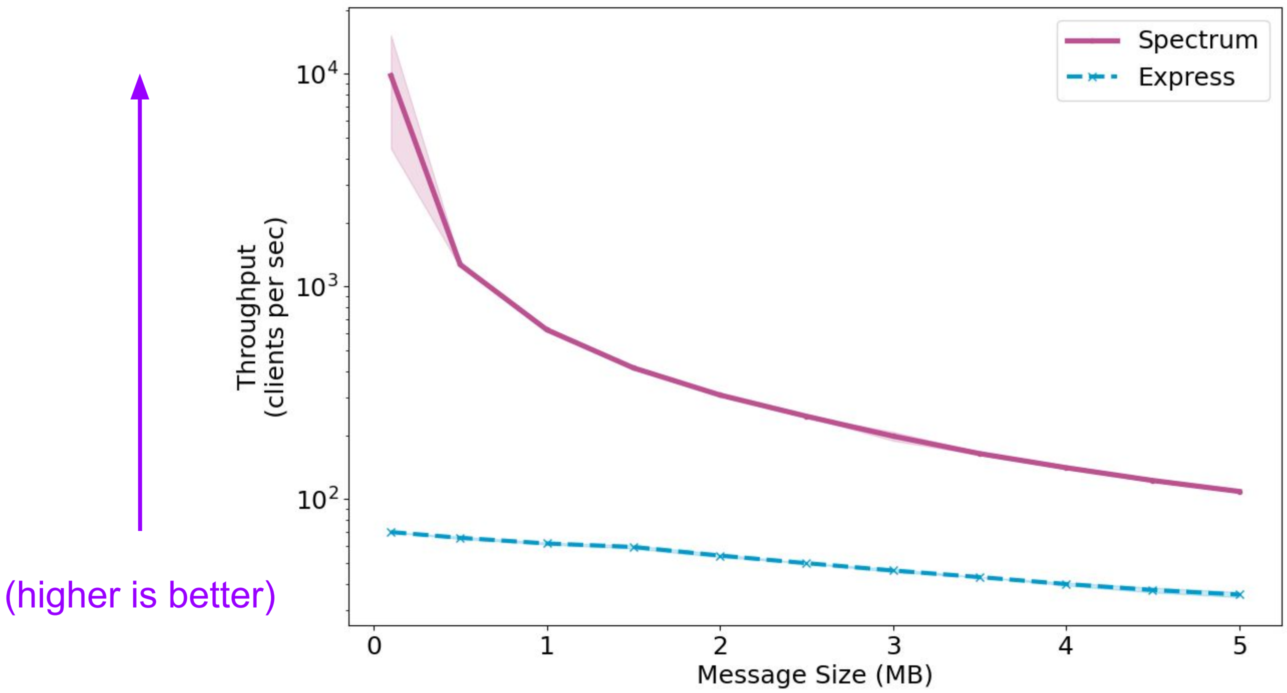
build passing

Implementation and experiments for the Spectrum paper.

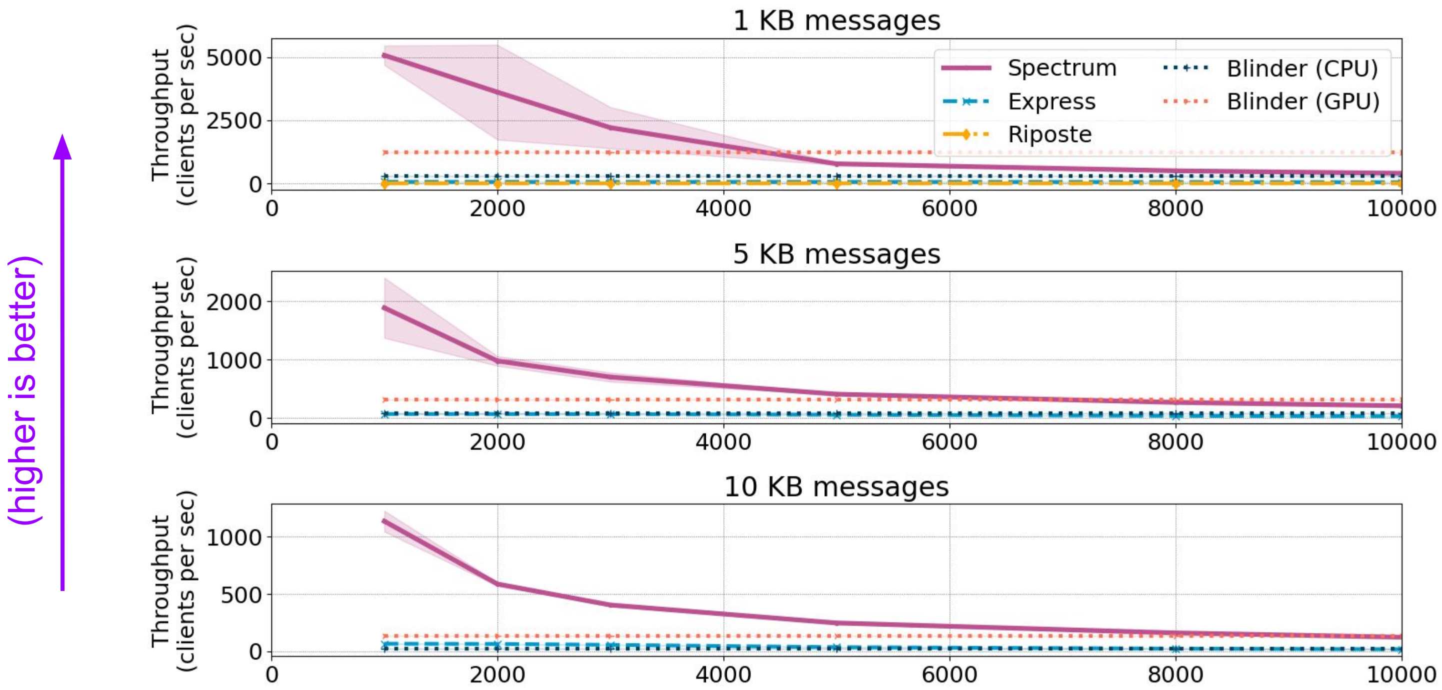
Disclaimer: research code, not for production use.

```
$ python -m experiments spectrum experiments.json
• [infrastructure] Environment(instance_type='c5.4xlarge', client_machines=1, worker_machines=2)
i [infrastructure] no changes to apply
✔ [infrastructure] connected (SSH)
✔ [infrastructure] etcd healthy

• Experiment(clients=50, channels=3, message_size=10240, instance_type='c5.4xlarge', clients_per_machi
ne=50, workers_per_machine=1, worker_machines_per_group=1, protocol=Symmetric(security=16))
✔ [experiment] 5375 queries in 1232ms => 4362 qps

• [infrastructure] Environment(instance_type='c5.4xlarge', client_machines=2, worker_machines=2)
✔ [infrastructure] found changes to apply:
  • aws_instance.client[1] will be created
✔ [infrastructure] created
✔ [infrastructure] connected (SSH)
✔ [infrastructure] etcd healthy

• Experiment(clients=100, channels=3, message_size=10240, instance_type='c5.4xlarge', clients_per_mach
ine=50, workers_per_machine=1, worker_machines_per_group=1, protocol=Symmetric(security=16))
⠿ [experiment] running
```

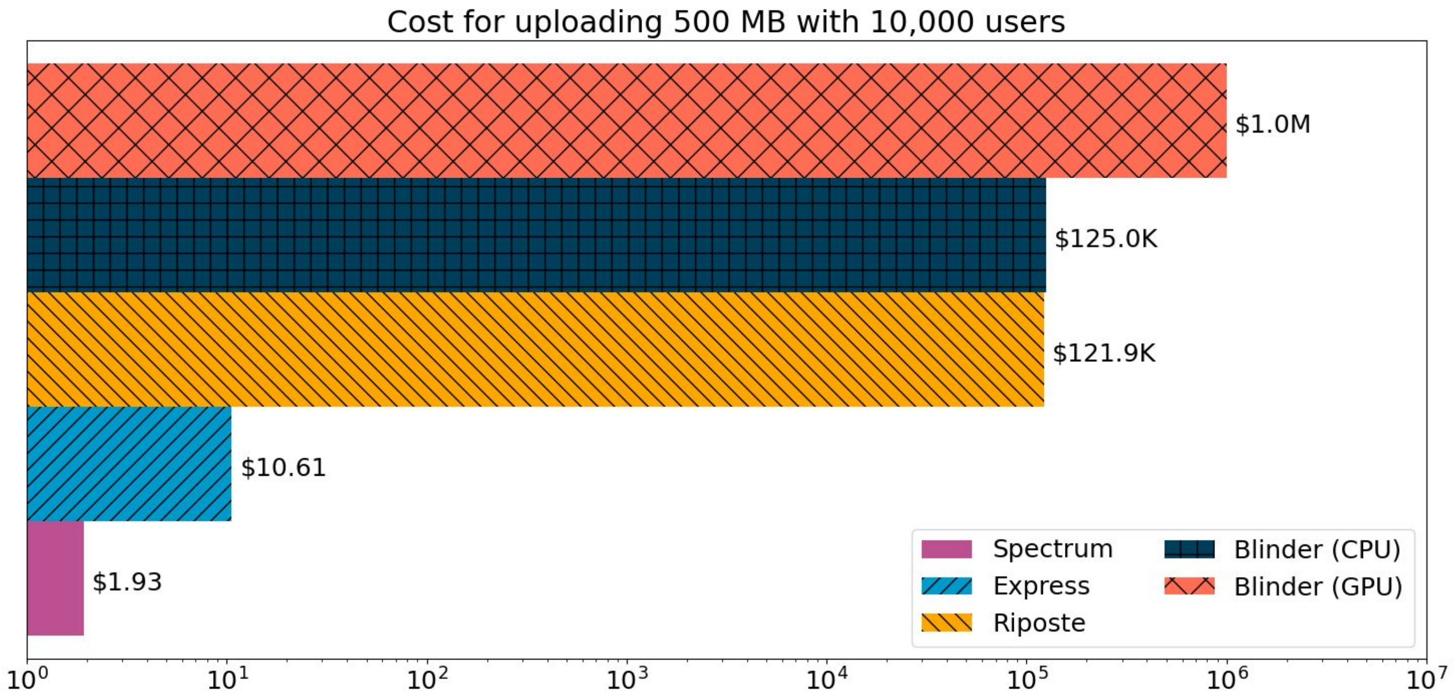
Spectrum: really fast with one channel & BIG messages



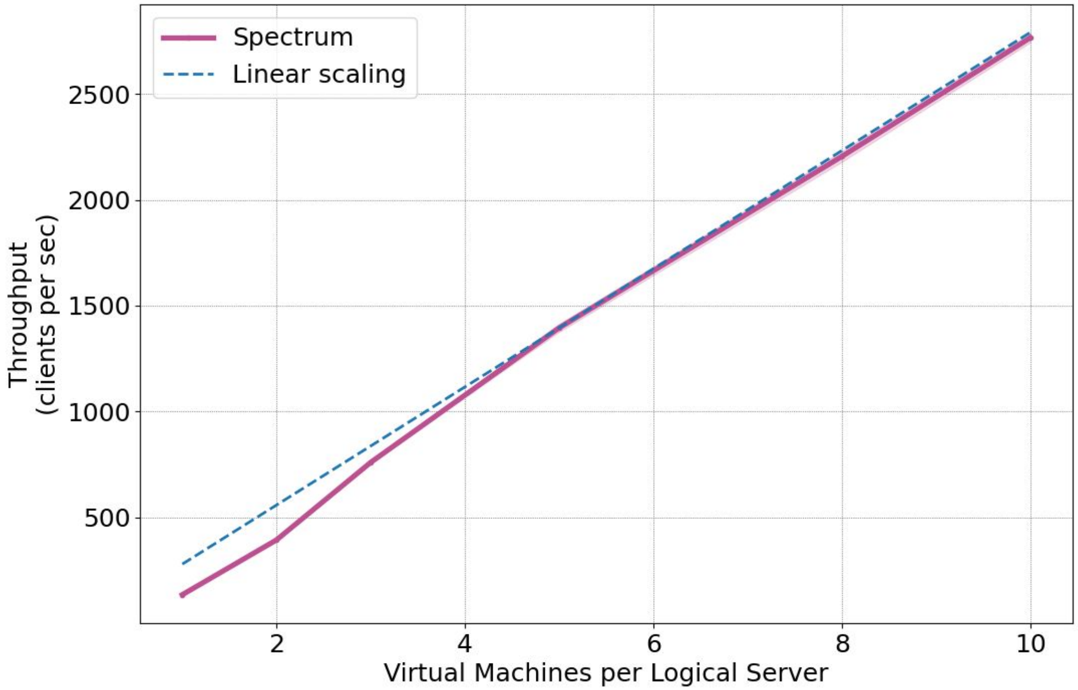
Spectrum: still fast with many channels (10K users)



Cost to upload a full-length documentary



Can parallelize each server



To share (anonymity set 10K)

Document	**Size** (MB)	**Time**	**Cost** ($ USD)
PDF (e.g., ePrint for Spectrum)	1	10 s	$ 0.01
Podcast (1 hr.)	50	8m 30s	$ 0.19
Documentary (2 hr. @ 720p)	500	1h 24m	$ 1.93

Thank You

NSDI'22 (to appear)
ePrint: ia.cr/2021/325

Zack Newman: zjn@mit.edu
Sacha Servan-Schreiber: 3s@mit.edu