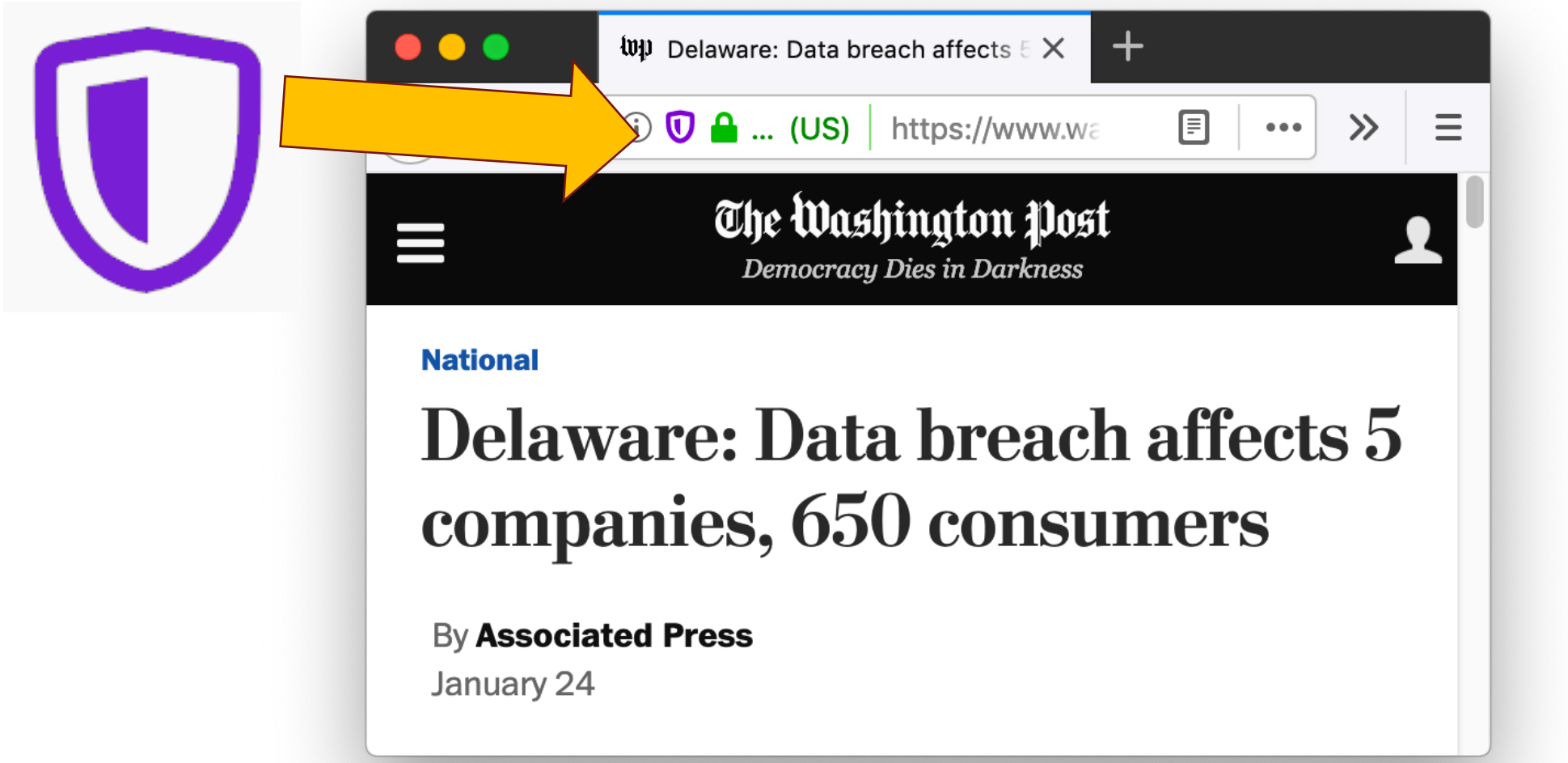
Privacy-preserving Firefox telemetry with Prio

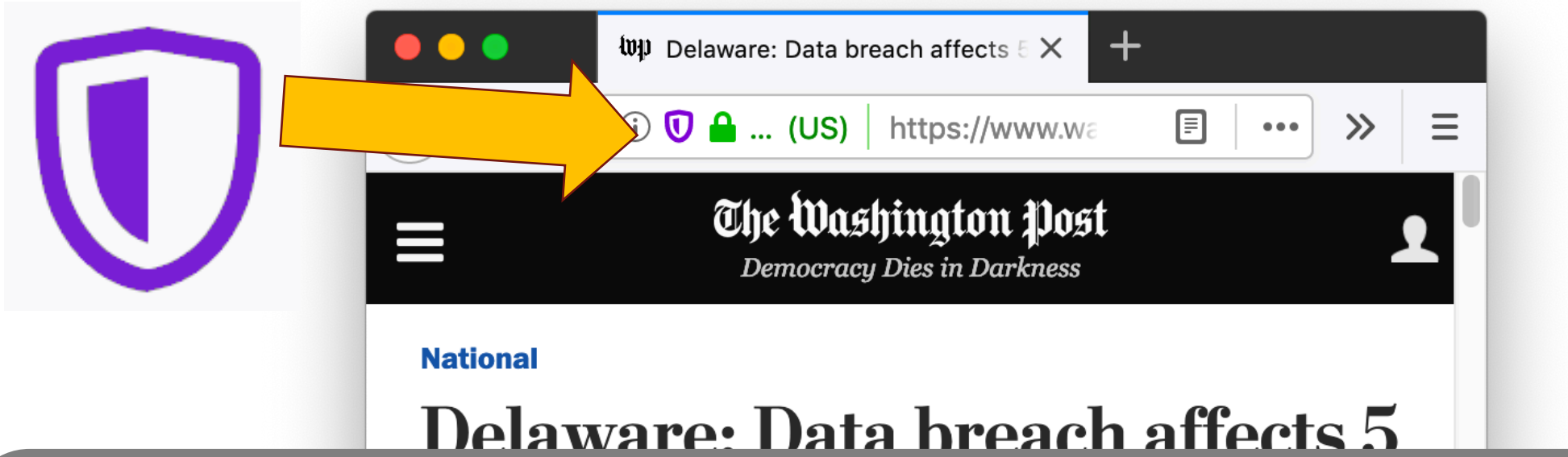
Henry Corrigan-Gibbs
(EPFL → MIT CSAIL)

In collaboration with: Dan Boneh (Stanford),
Gary Chen, Steven Englehardt, Robert Helmer, Chris Hutten-Czapski,
Anthony Miyaguchi, Eric Rescorla, and Peter Saint-Andre (Mozilla)

Running example:
Measuring effectiveness of tracking protection



Running example:
Measuring effectiveness of tracking protection

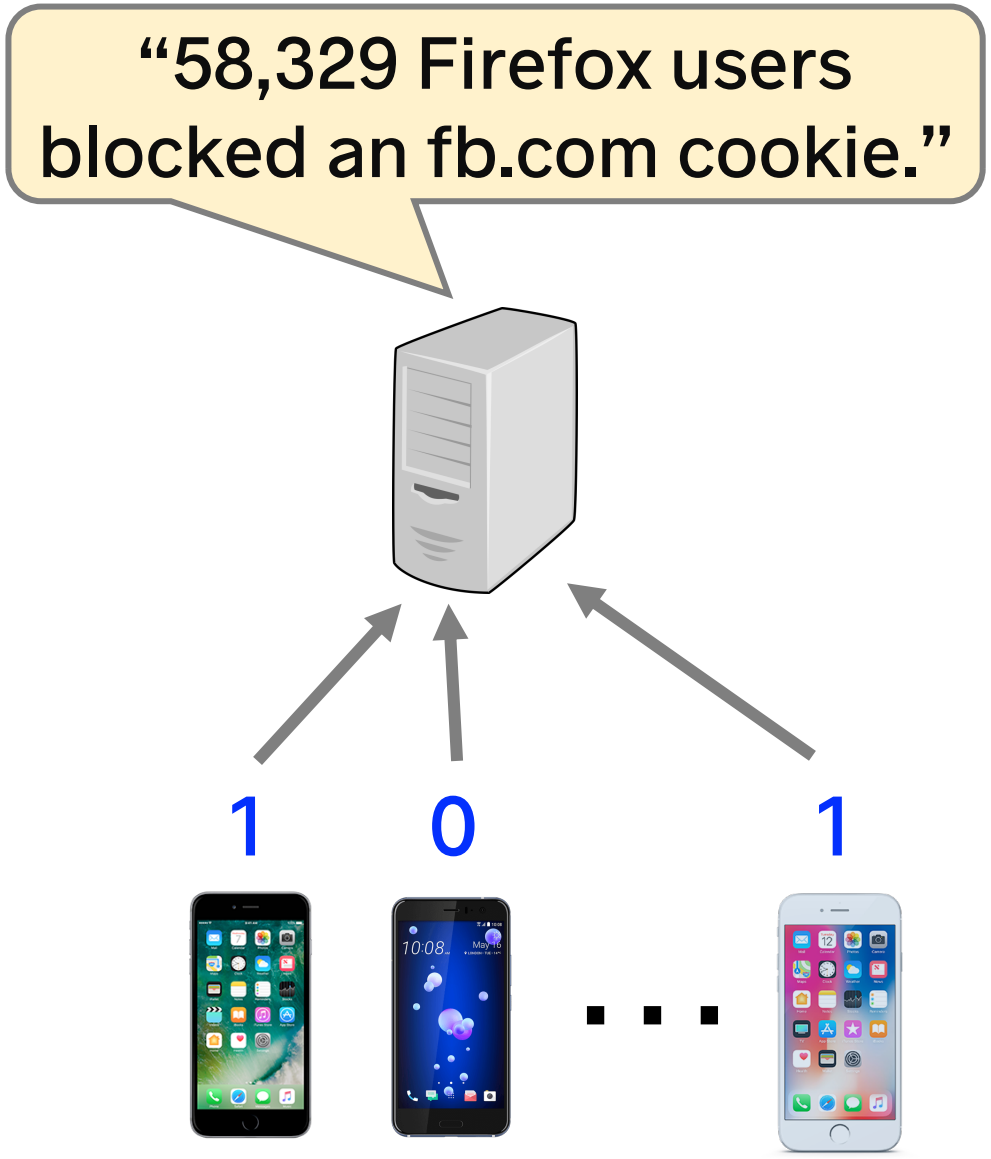


Mozilla wants to know:

“How many Firefox users blocked a tracking cookie from fb.com?”

Software vendors often answer these questions by collecting **sensitive** usage data directly.

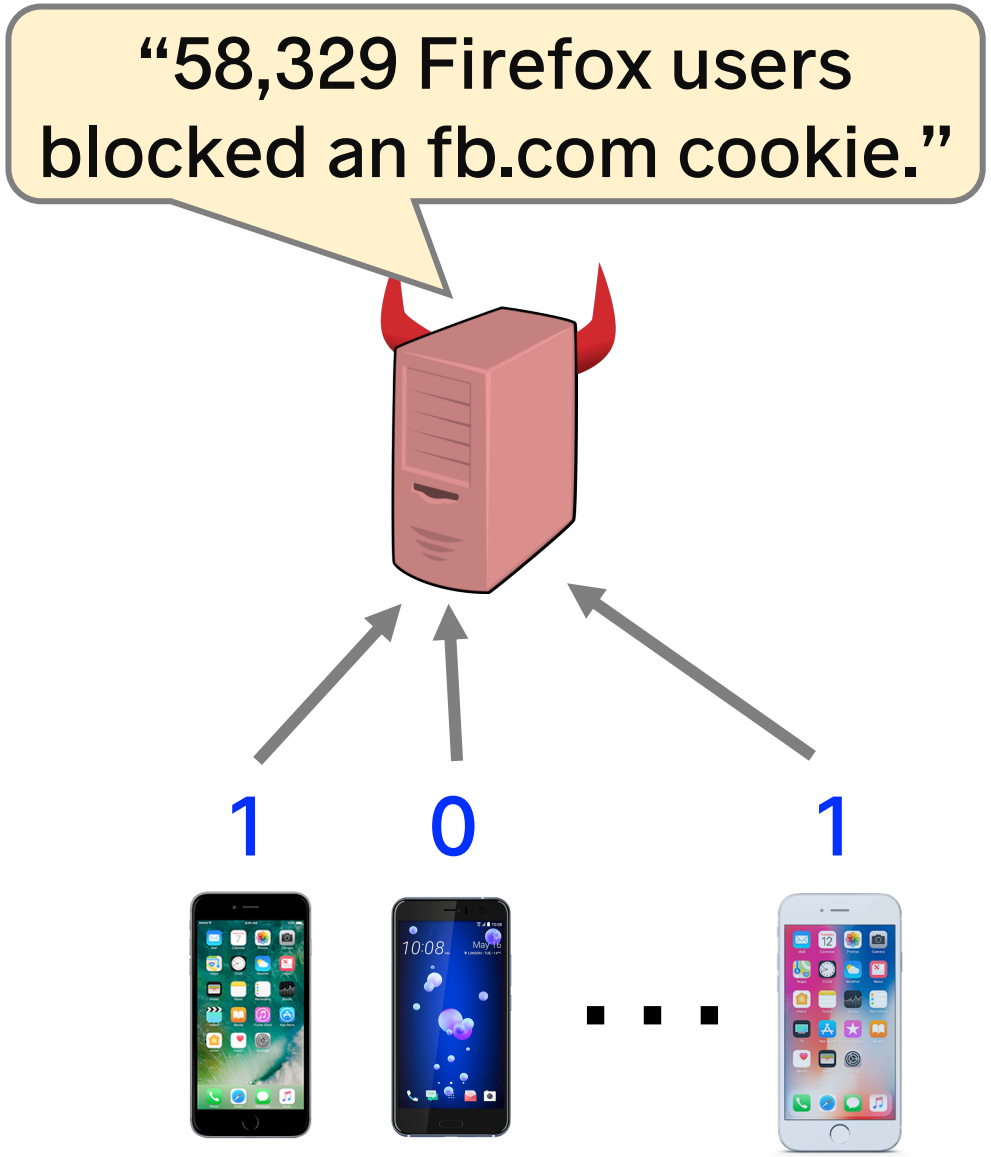
→ Single point of failure.



Software vendors often answer these questions by collecting **sensitive** usage data directly.

→ Single point of failure.

- Theft by attackers
- Abuse by malicious insiders
- Snooping by governments



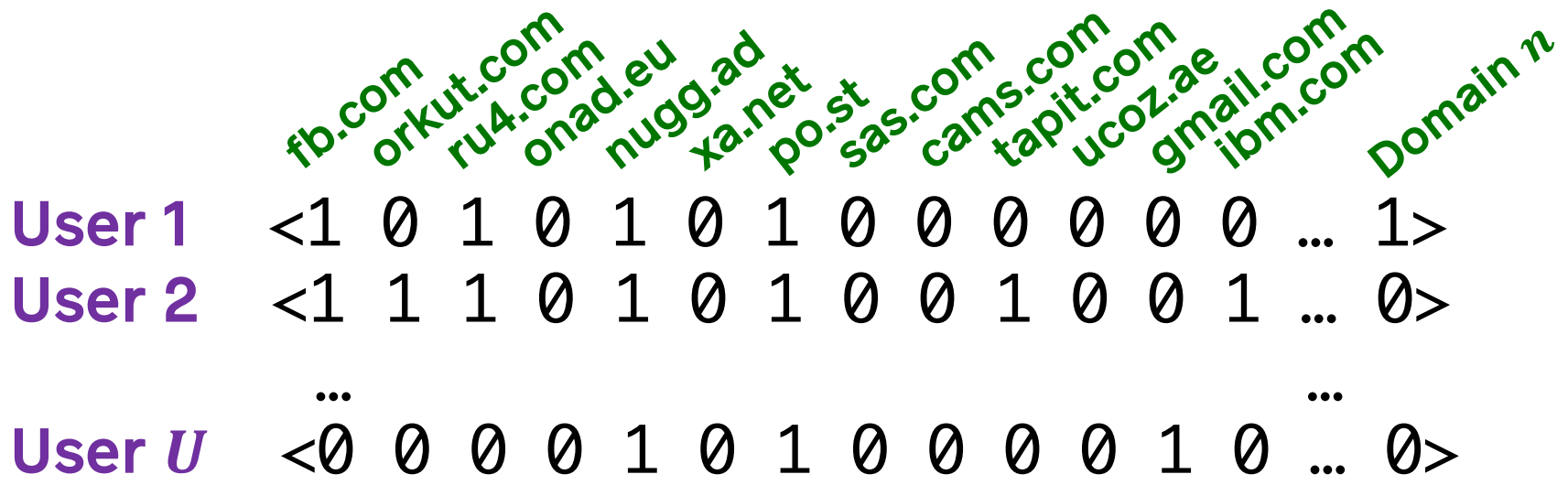
Prio: Aggregate data without the privacy risks

C-G and Boneh (NSDI 2017)

- Collect aggregate usage data
  without seeing any single user's data.
- New cryptography makes this system practical
  - Proofs on secret-shared data
- Basis for Mozilla's new privacy-preserving telemetry system
  - In pilot phase: Enabled by default in Firefox's "Nightly" build
  - Largest deployment of technology based on PCPs
    (probabilistically checkable proofs)

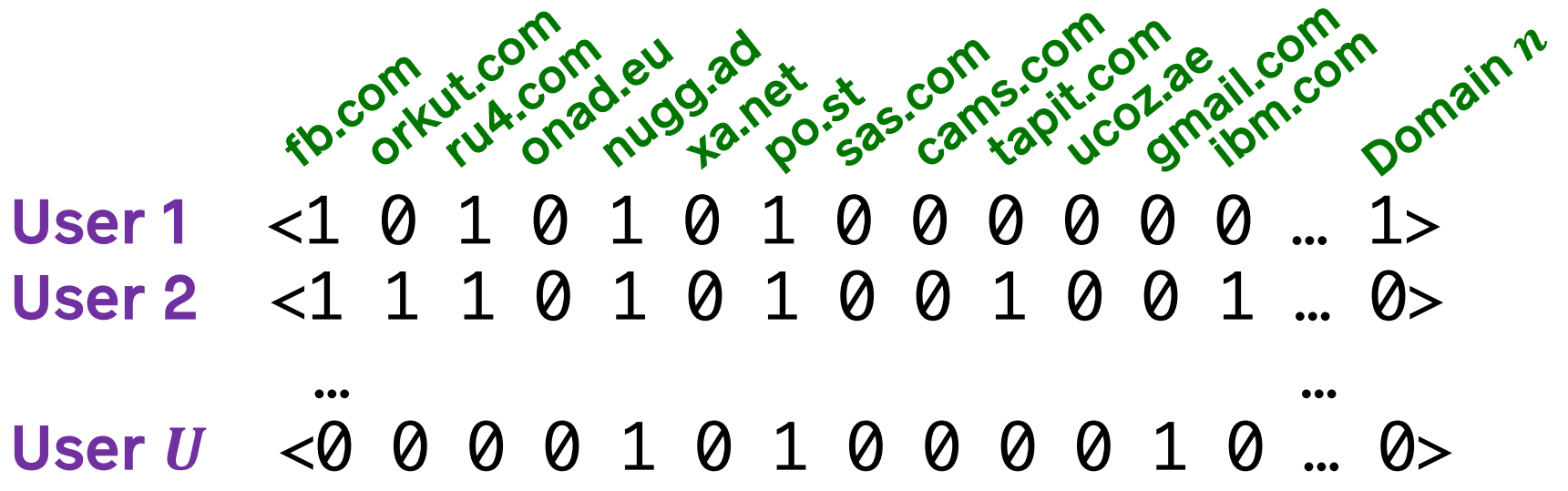
Running example:
Measuring effectiveness of tracking protection

- There are $n \approx 2{,}500$ domains on the tracking-protection blocklist
- For each blocked domain, each user i has a bit
  - Bit is "1" iff user i's browser ever blocked cookies from domain.com
  - These bits are **sensitive** – reveal user's browsing history



Running example:
Measuring effectiveness of tracking protection

- Mozilla wants the sum of these vectors over all users i



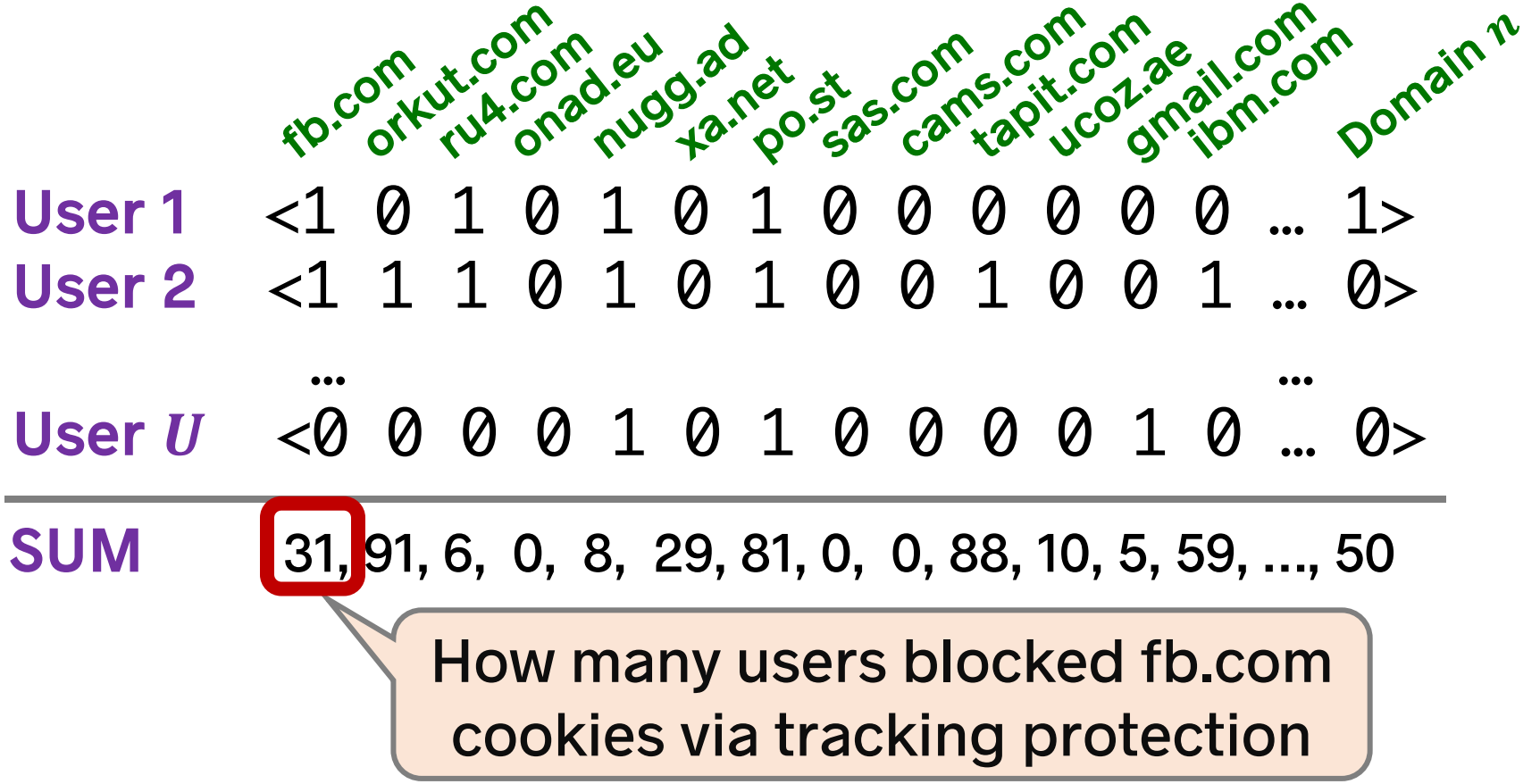
Running example:
Measuring effectiveness of tracking protection

- Mozilla wants the sum of these vectors over all users i

	fb.com	orkut.com	ru4.com	onad.eu	nugg.ad	xa.net	po.st	sas.com	cams.com	tapit.com	ucoz.ae	gmail.com	ibm.com	…	Domain n
User 1	<1	0	1	0	1	0	1	0	0	0	0	0	0	…	1>
User 2	<1	1	1	0	1	0	1	0	0	1	0	0	1	…	0>
	…													…	
User U	<0	0	0	0	1	0	1	0	0	0	0	1	0	…	0>
SUM	31,	91,	6,	0,	8,	29,	81,	0,	0,	88,	10,	5,	59,	…,	50

Running example:
Measuring effectiveness of tracking protection

- Mozilla wants the sum of these vectors over all users i



Running example:
Measuring effectiveness of tracking protection

- Mozilla wants the sum of these vectors over all users i

	fb.com	orkut.com	ru4.com	onad.eu	nugg.ad	xa.net	po.st	sas.com	cams.com	tapit.com	ucoz.ae	gmail.com	ibm.com	…	Domain n
User 1	<1	0	1	0	1	0	1	0	0	0	0	0	0	…	1>
User 2	<1	1	1	0	1	0	1	0	0	1	0	0	1	…	0>
	…														…
User U	<0	0	0	0	1	0	1	0	0	0	0	1	0	…	0>
SUM	31	91	6	0	8	29	81	0	0	88	10	5	59	…,	50

Running example:
Measuring effectiveness of tracking protection

- Mozilla wants the sum of these vectors over all users i

		fb.com	orkut.com	ru4.com	onad.eu	nugg.ad	xa.net	po.st	sas.com	cams.com	tapit.com	ucoz.ae	gmail.com	ibm.com	Domain n
User 1	x_1	0	1	0	1	0	1	0	0	0	0	0	0	…	1>
User 2	x_2	1	1	0	1	0	1	0	0	1	0	0	1	…	0>
	…														
User U	x_U	0	0	0	1	0	1	0	0	0	0	1	0	…	0>
SUM	$\Sigma_{i=1}^{U} x_i$	, 6,	0,	8,	29,	81,	0,	0,	88,	10,	5,	59,	…,	50	

Running example:
Measuring effectiveness of tracking protection

- Mozilla wants the sum of these vectors over all users i

Prio: System goals

1. Correctness. If clients and servers are honest, servers learn $\Sigma_i x_i$

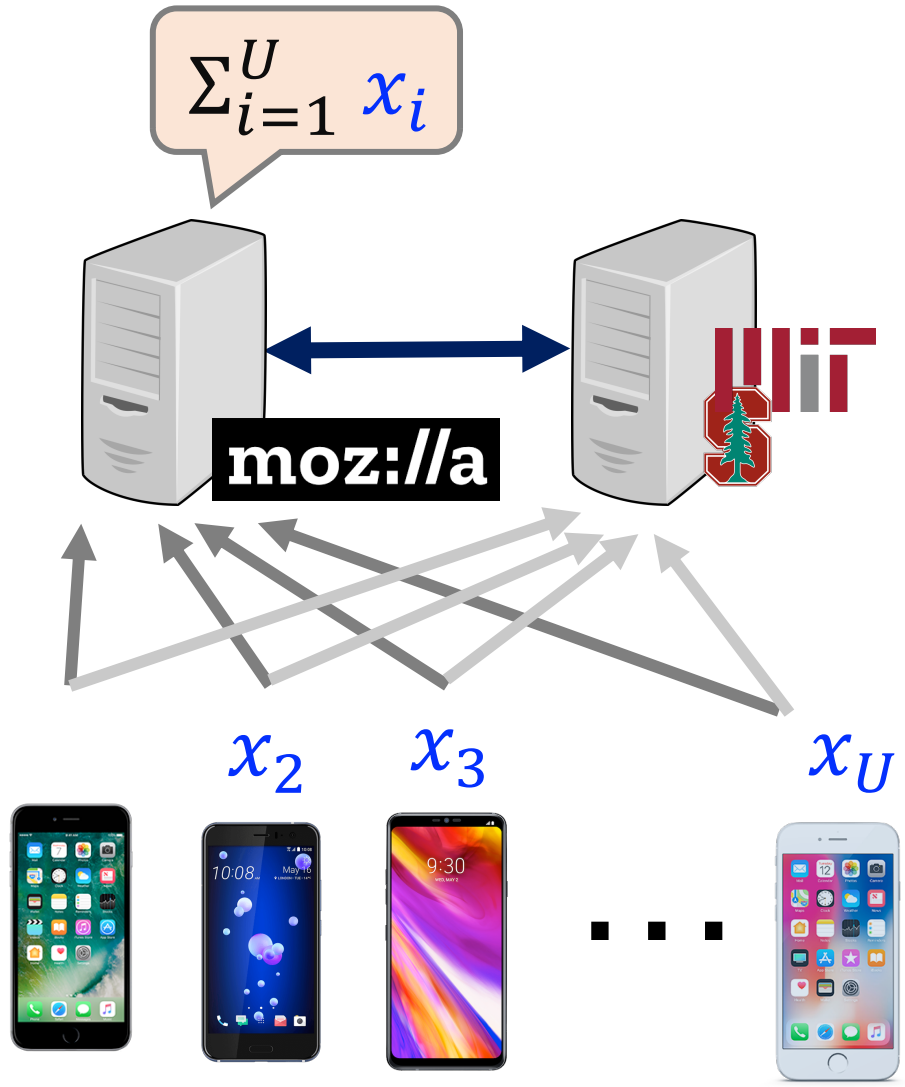
Extension: Maintain correctness in spite of server faults

2. f-Privacy. Attacker must compromise <u>all</u> servers to learn more than $\Sigma_i x_i$

Extension: Differential privacy [DMNS06]

3. Disruption resistance. The worst that a malicious client can do is lie about her input.

4. Efficiency. Handle millions of submissions per server per hour



Prio: System goals

1. Correctness. If clients and servers are honest, servers learn $\Sigma_i x_i$

Extension: Maintain correctness in spit

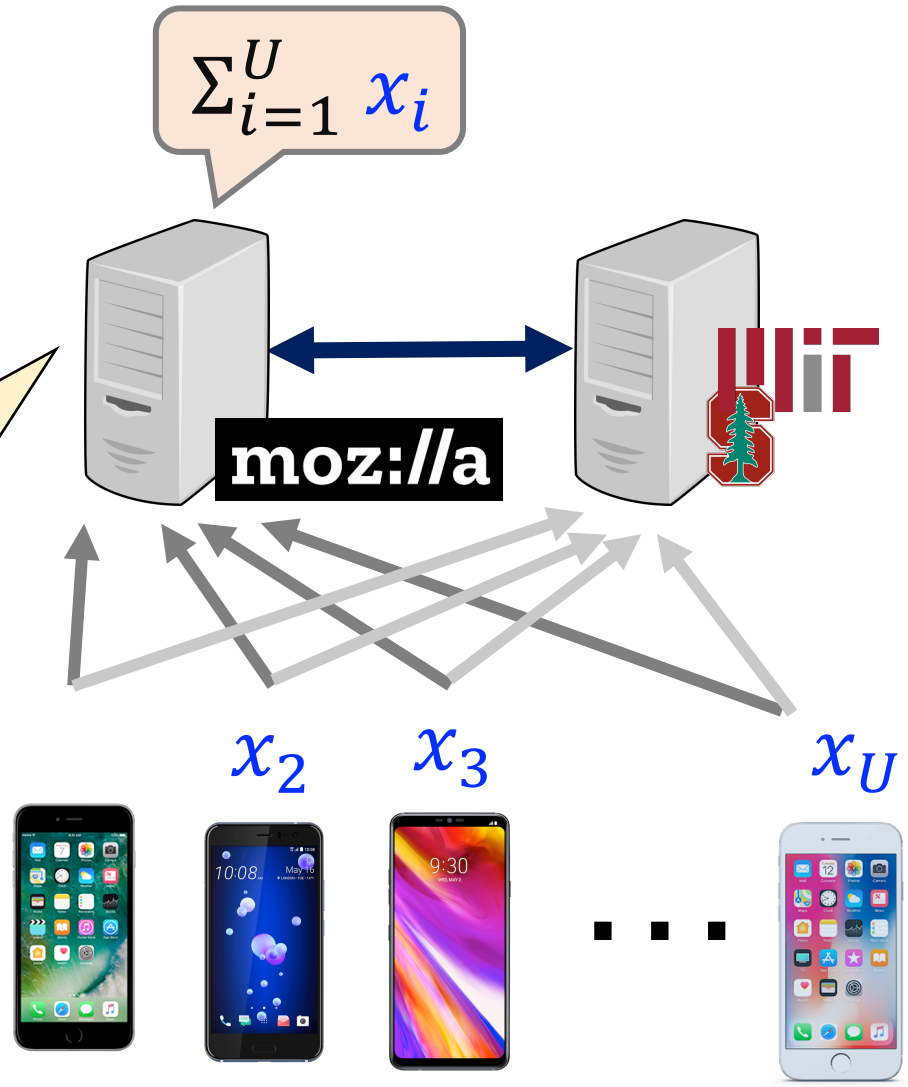
2. f-Privacy. A compromise a more than $\Sigma_i x_i$

Extension: Diffe

3. Disruption The worst that a malicious client can do is lie about her input.

4. Efficiency. Handle millions of submissions per server per hour

Attacker must compromise <u>all</u> servers to learn private data.



Prio: System goals

1. Correctness. If clients and servers are honest, servers learn $\Sigma_i x_i$

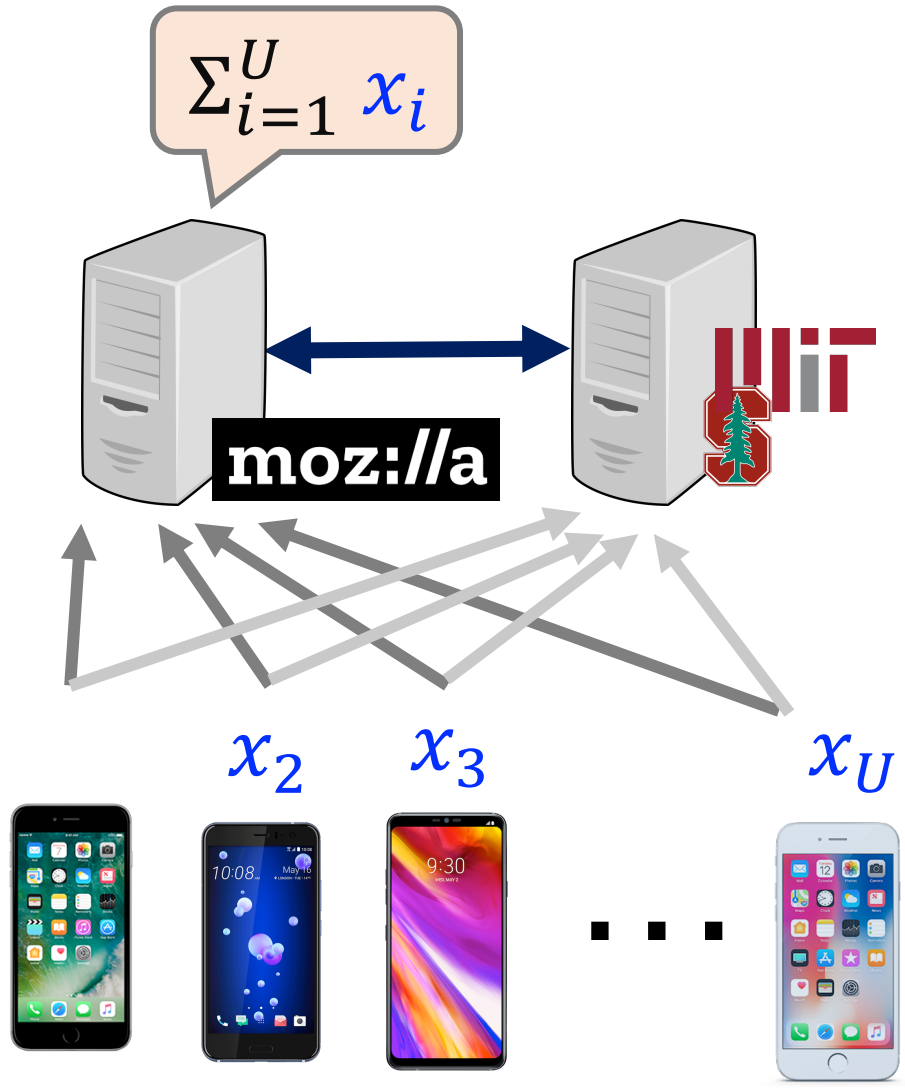
Extension: Maintain correctness in spite of server faults

2. f-Privacy. Attacker must compromise <u>all</u> servers to learn more than $\Sigma_i x_i$

Extension: Differential privacy [DMNS06]

3. Disruption resistance. The worst that a malicious client can do is lie about her input.

4. Efficiency. Handle millions of submissions per server per hour



Prio: System goals

1. Correctness. If clients and servers are honest, servers learn $\Sigma_i x_i$

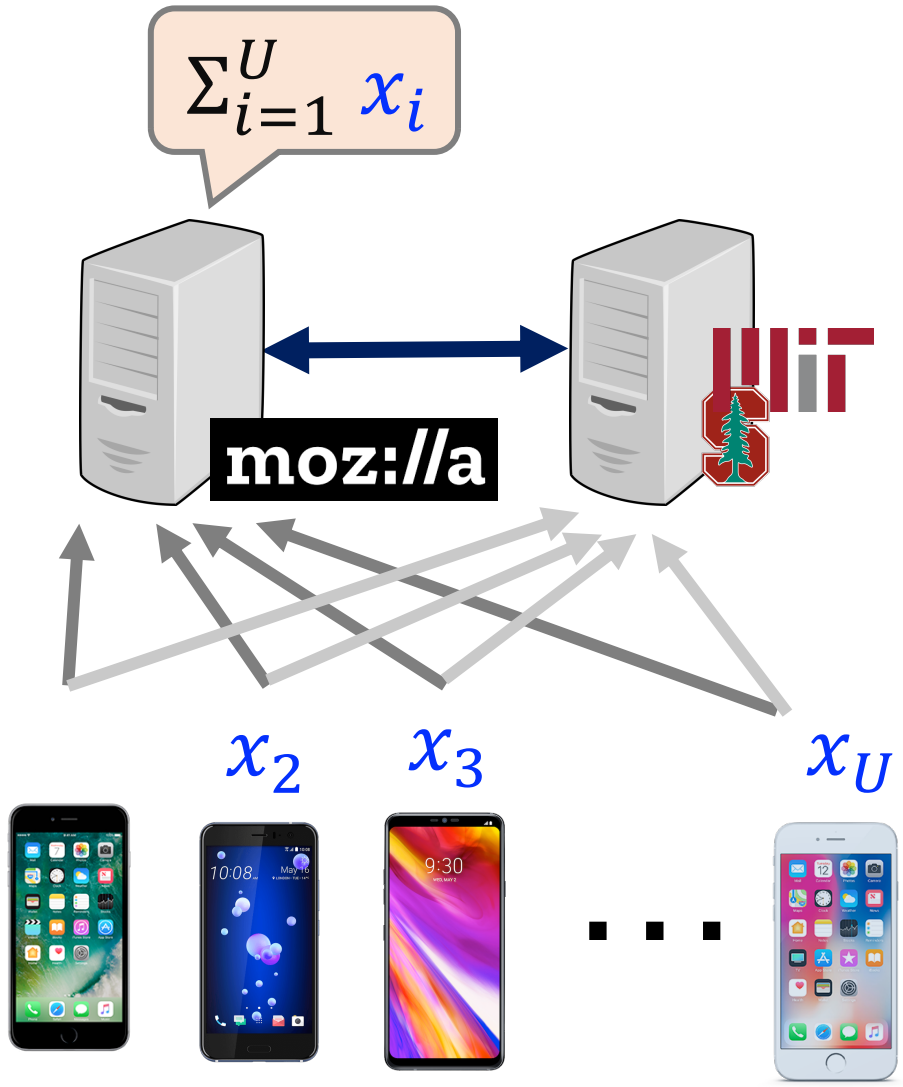
Extension: Maintain correctness in spite of server faults

2. f-Privacy. Attacker must compromise <u>all</u> servers to learn more than $\Sigma_i x_i$

Extension: Differential privacy [DMNS06]

3. Disruption resistance. The worst that a malicious client can do is lie about her input.

4. Efficiency. Handle millions of submissions per server per hour



Prio: System goals

1. Correctness. If clients and servers are honest, servers learn $\Sigma_i x_i$

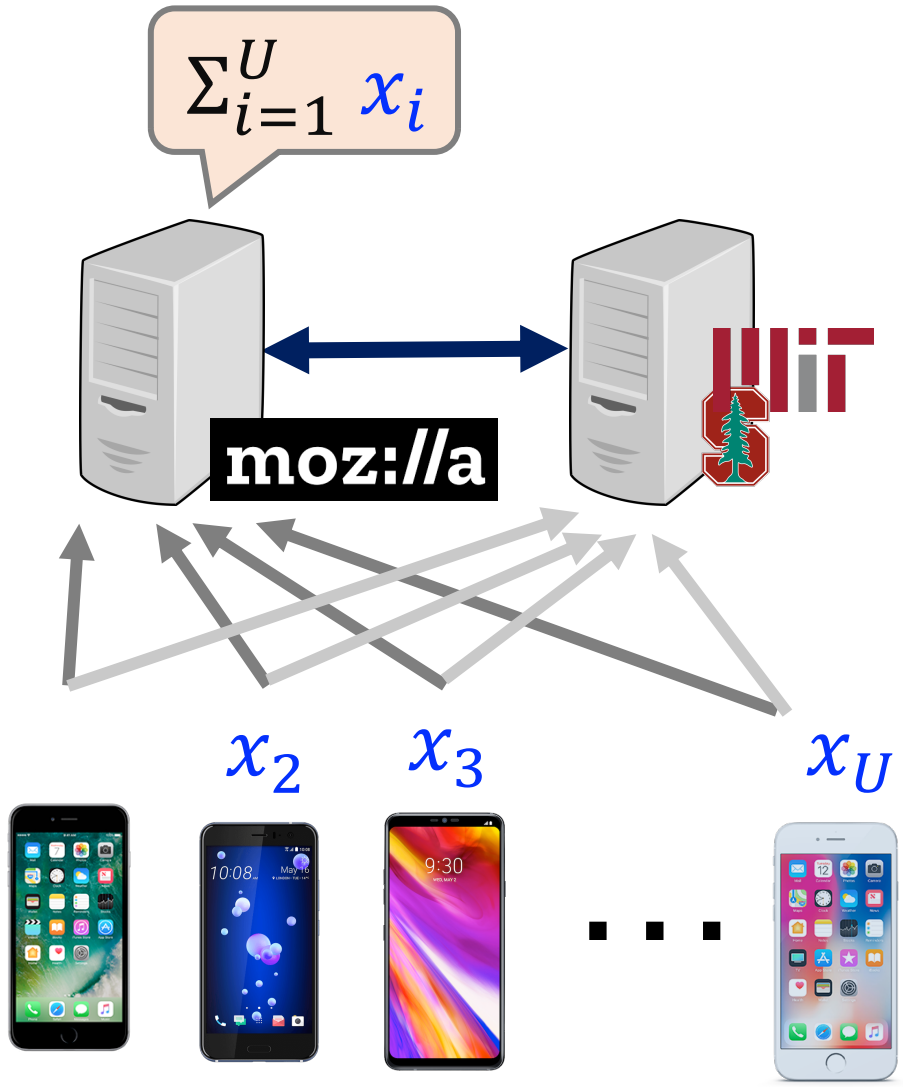
Extension: Maintain correctness in spite of server faults

2. f-Privacy. Attacker must compromise <u>all</u> servers to learn more than $\Sigma_i x_i$

Extension: Differential privacy [DMNS06]

3. Disruption resistance.
The worst that a malicious client can do is lie about her input.

4. Efficiency. Handle millions of submissions per server per hour



Prio: System goals

1. Correctness. If clients and servers are honest, servers learn $\Sigma_i x_i$

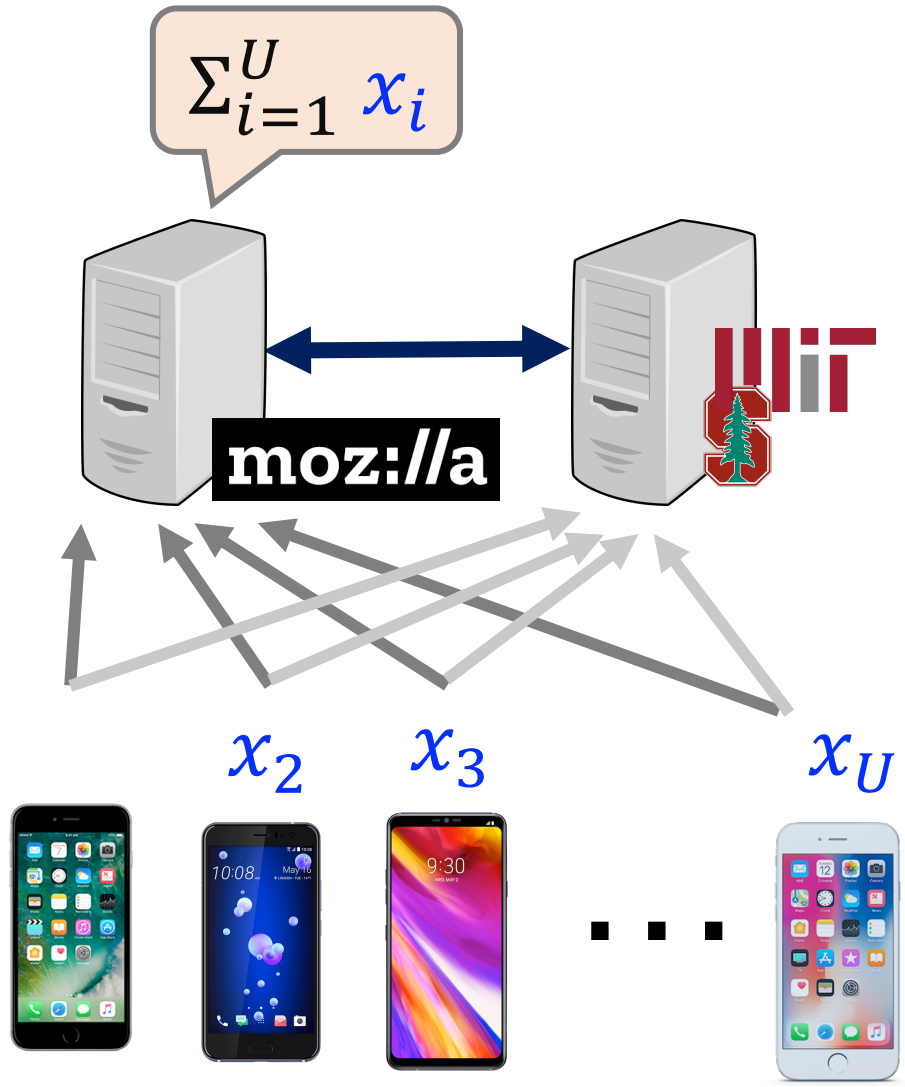
Extension: Maintain correctness in spite of server faults

2. f-Privacy. Attacker must compromise <u>all</u> servers to learn more than $\Sigma_i x_i$

Extension: Differential privacy [DMNS06]

3. Disruption resistance. The worst that a malicious client can do is lie about her input.

4. Efficiency. Handle millions of submissions per server per hour



Prio: System goals

1. Correctness. If clients and servers are honest, servers learn $\Sigma_i x_i$

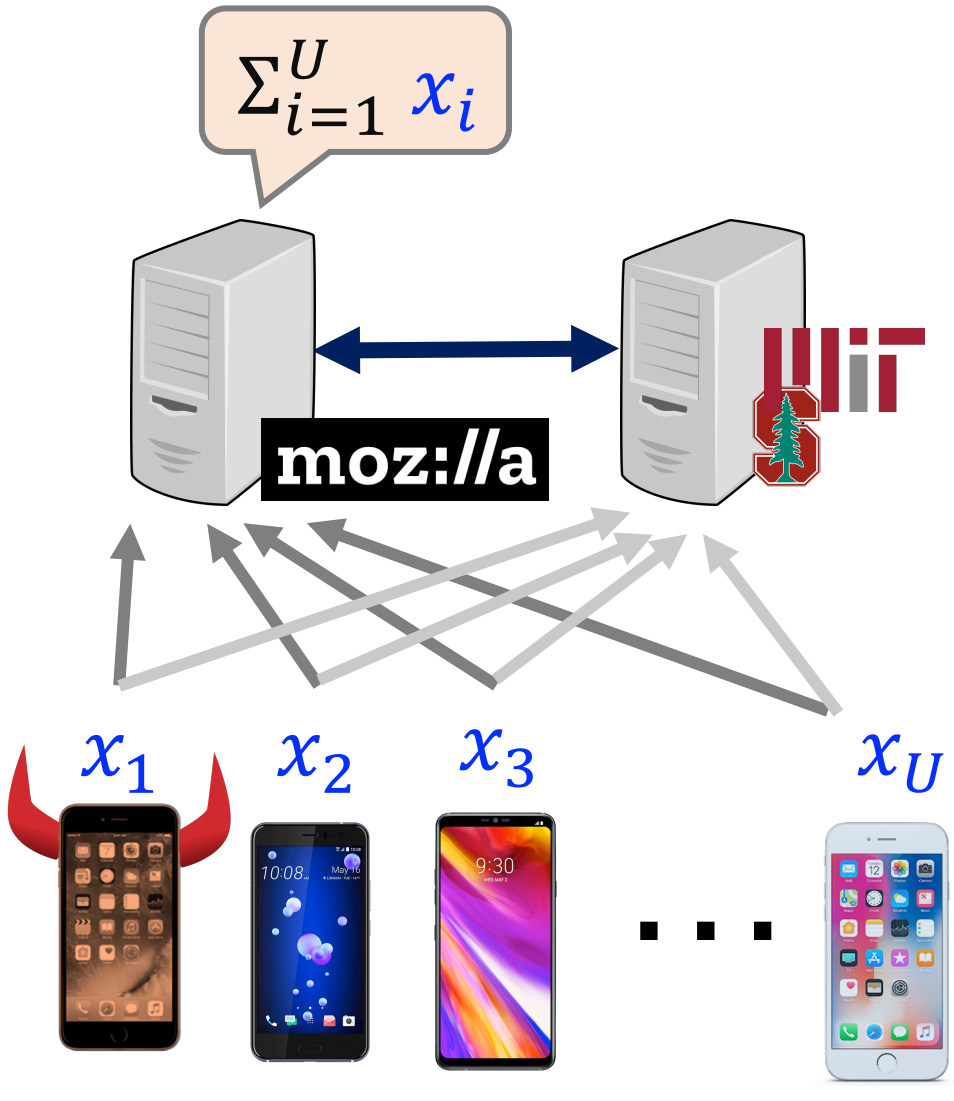
Extension: Maintain correctness in spite of server faults

2. f-Privacy. Attacker must compromise <u>all</u> servers to learn more than $\Sigma_i x_i$

Extension: Differential privacy [DMNS06]

3. Disruption resistance. The worst that a malicious client can do is lie about her input.

4. Efficiency. Handle millions of submissions per server per hour



Prio: System goals

1. Correctness. If clients and servers are honest, servers learn $\Sigma_i x_i$

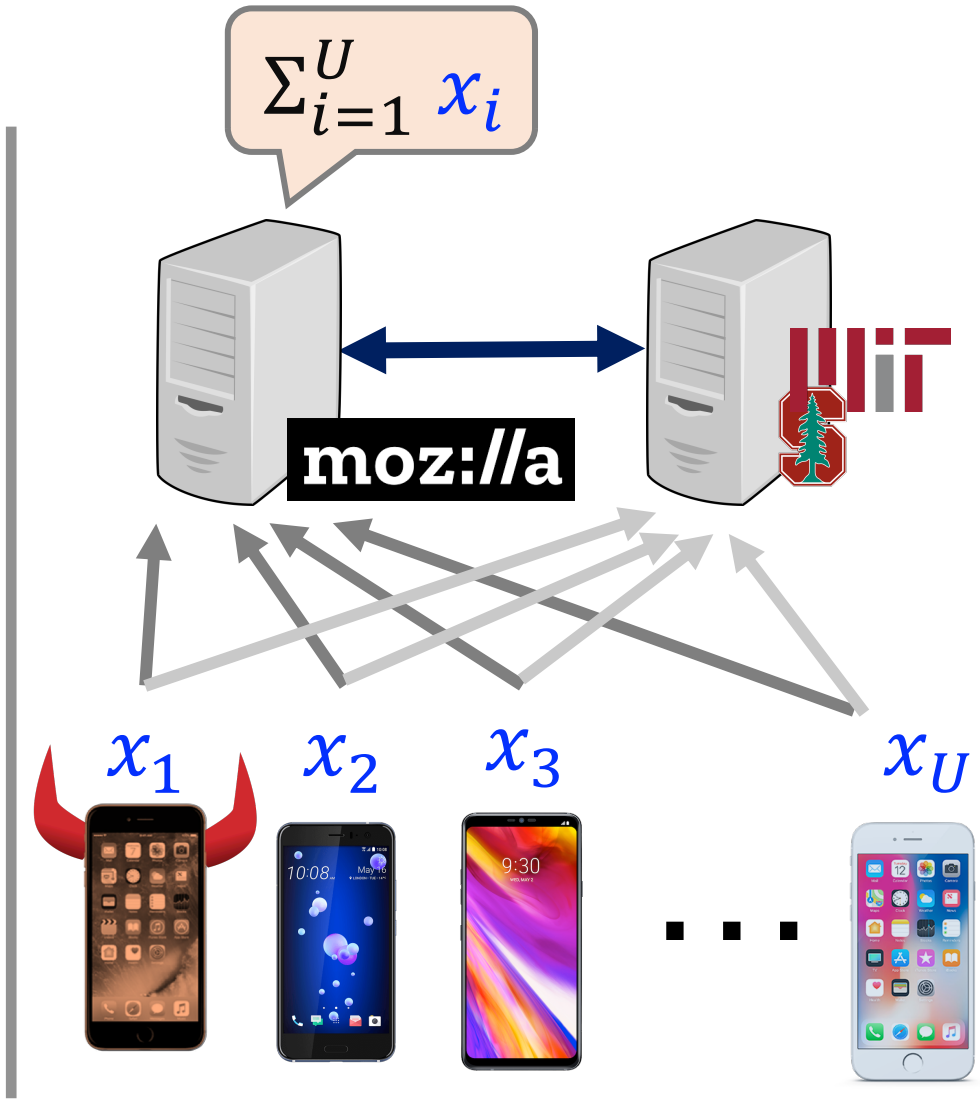
Extension: Maintain correctness in spite of server faults

2. f-Privacy. Attacker must compromise <u>all</u> servers to learn more than $\Sigma_i x_i$

Extension: Differential privacy [DMNS06]

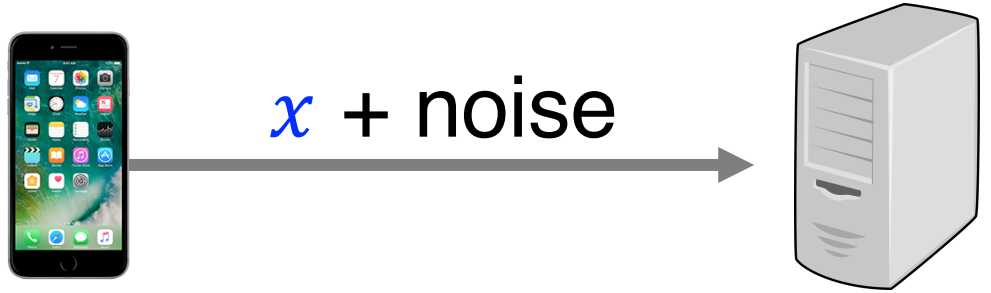
3. Disruption resistance. The worst that a malicious client can do is lie about her input.

4. Efficiency. Handle millions of submissions per server per hour



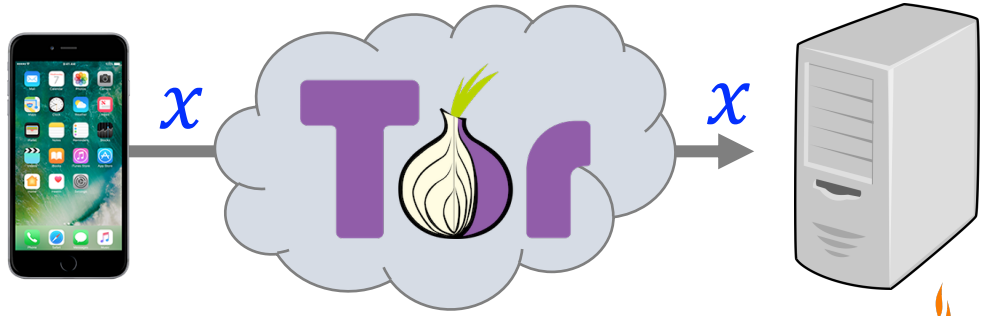
Relax correctness

Randomized response: [W65], [DMNS06], [DJW13], [BS15]
RAPPOR (2014, 2016), Wang et al. (2017),
Ding et al. (2017)...



Relax privacy model

Tor: PrivStats (2011), ANONIZE (2014), ...
SGX: Prochlo (2017), SGX-BigMatrix (2017), ...
Honest but curious: PDDP (2012), SplitX (2013), ...

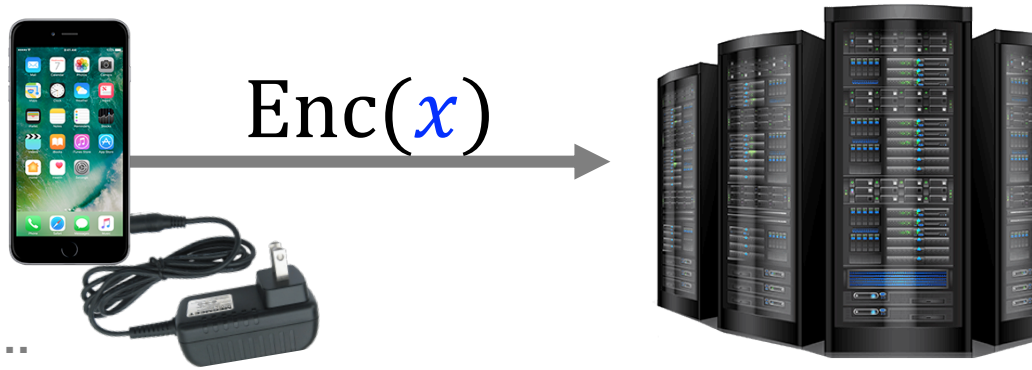


Relax disruption resistance

Private metering (2011), PrivEx-S2 (2014),
PrivCount (2016), Federated ML (2016, 2017), ...

Relax efficiency

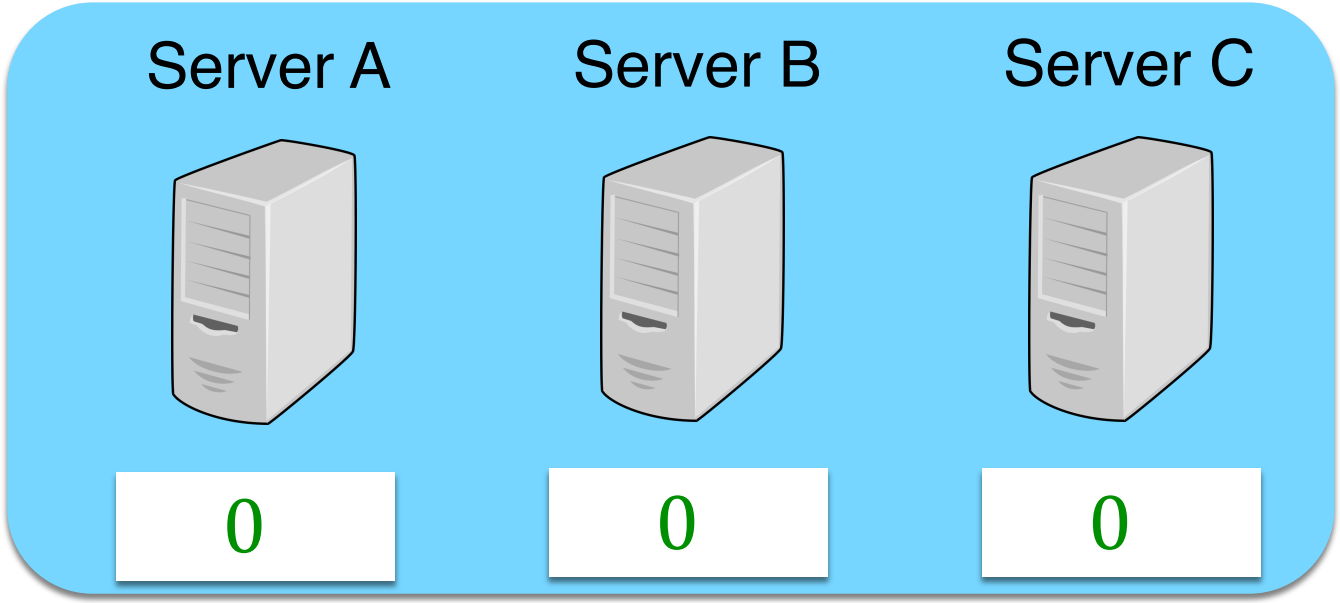
P4P (2010), Grid aggregation (2011), Haze (2013),
PrivEx-D2 (2014), Succinct sketches (2016), HisTorϵ (2017), ...
General MPC [GMW87], [BGW88]: FairPlay (2004), FairplayMP (2008), SEPIA (2010), Private matrix factorization (2013), UnLynx (2017), Private ridge regression (2018), Shuffle model (2017, 2019), ...



Straw-man scheme

Private sums without disruption resistance

[C88], [BGW88], ...
[KDK11] [DFKZ13] [PrivEx14] ...

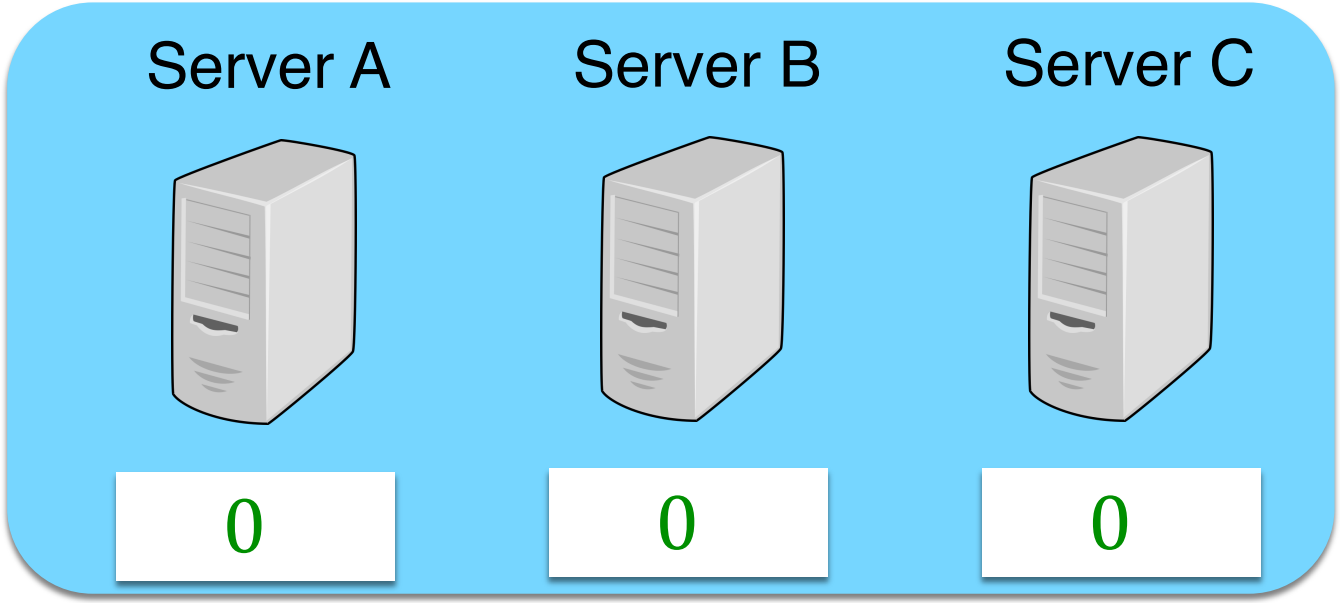


$x_1 = 1$

Straw-man scheme

Private sums without disruption resistance

[C88], [BGW88], ...
[KDK11] [DFKZ13] [PrivEx14] ...



$x_1 = \mathbf{1}$

Pick three random "shares" that sum to $x_1 = \mathbf{1}$.

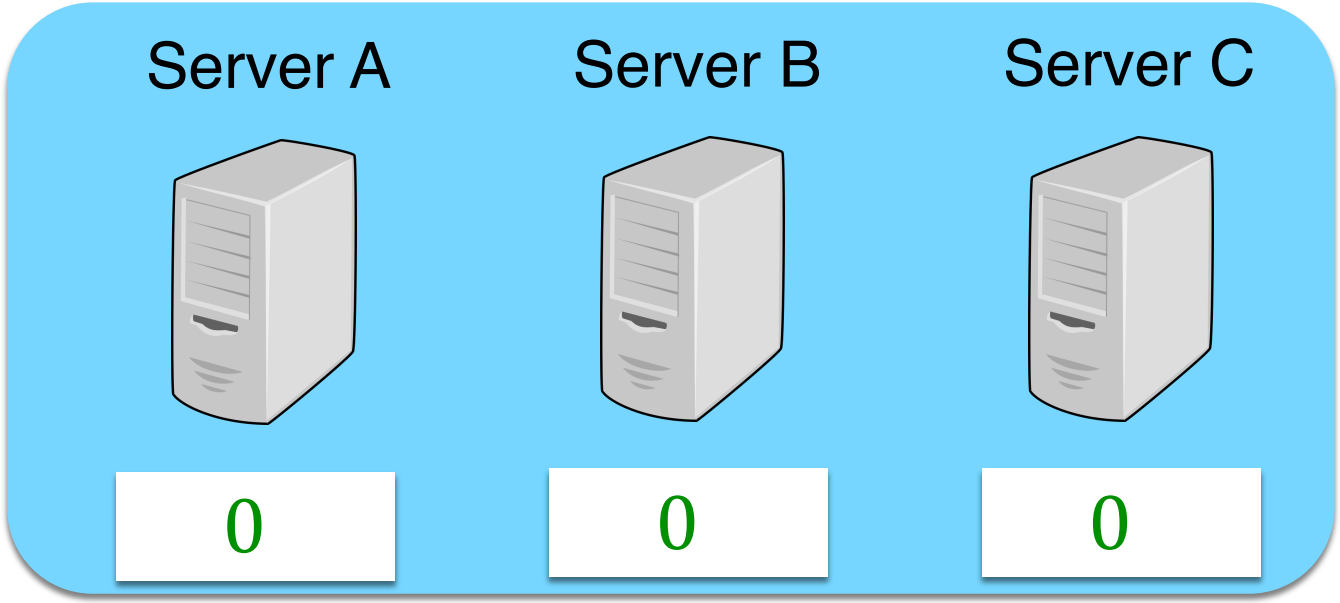
$$\mathbf{1} = 15 + (-12) + (-2) \pmod{p}$$

Send one share to each server.

Straw-man scheme

Private sums without disruption resistance

[C88], [BGW88], ...
[KDK11] [DFKZ13] [PrivEx14] ...



$x_1 = \mathbf{1}$

Pick three random "shares" that sum to $x_1 = \mathbf{1}$.

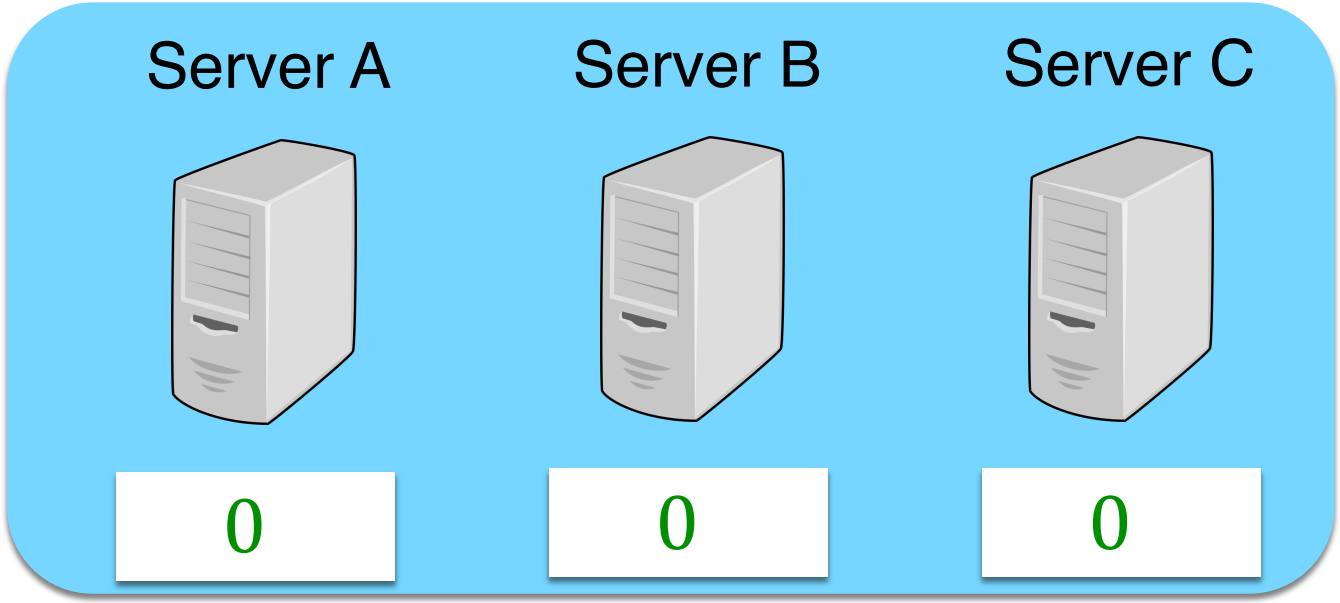
$$\mathbf{1} = 15 + (-12) + (-2) \pmod{p}$$

Send one share to each server.

Straw-man scheme

Private sums without disruption resistance

[C88], [BGW88], ...
[KDK11] [DFKZ13] [PrivEx14] ...



$15 \quad -12 \quad -2$

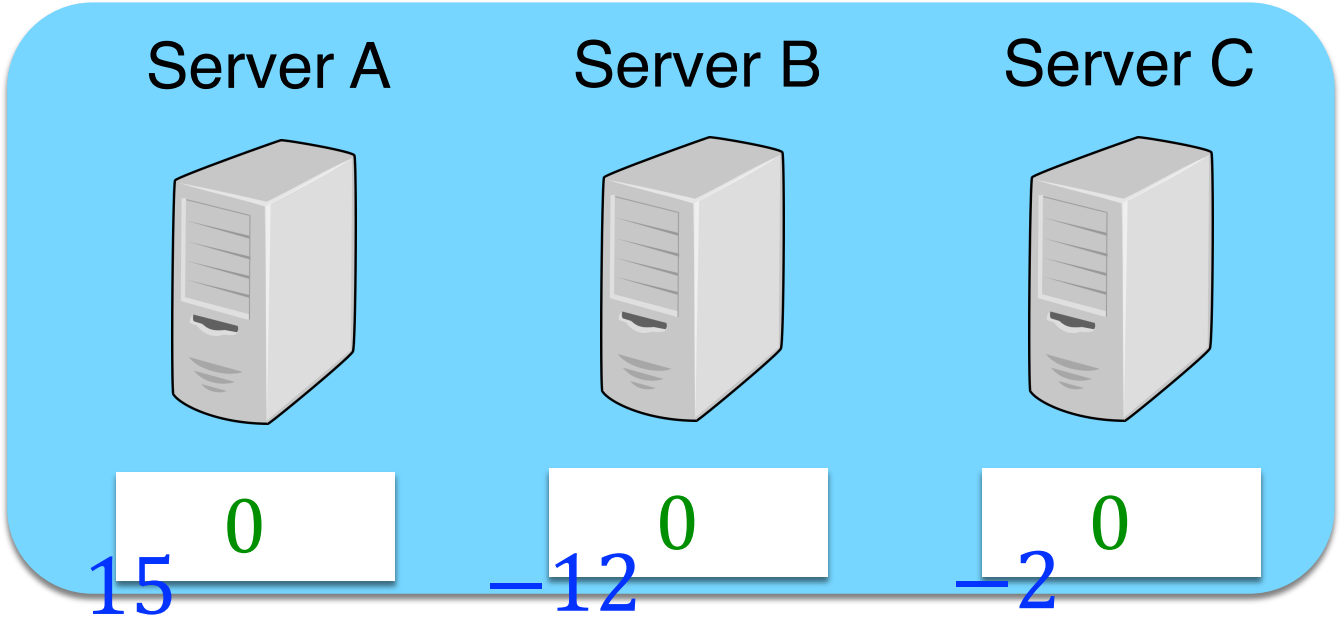
$x_1 = \mathbf{1}$

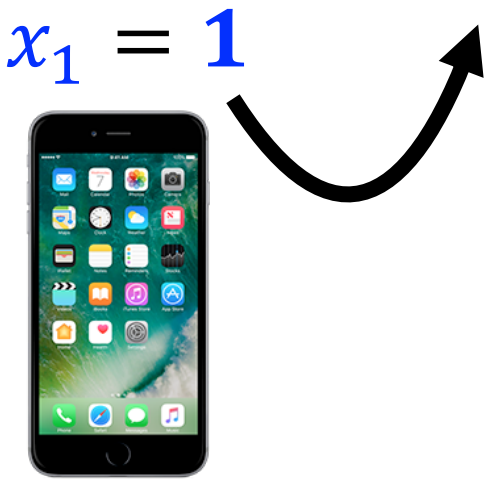
Pick three random "shares" that sum to $x_1 = \mathbf{1}$.

$$\mathbf{1} = 15 + (-12) + (-2) \pmod{p}$$

Send one share to each server.

Straw-man scheme
Private sums without disruption resistance



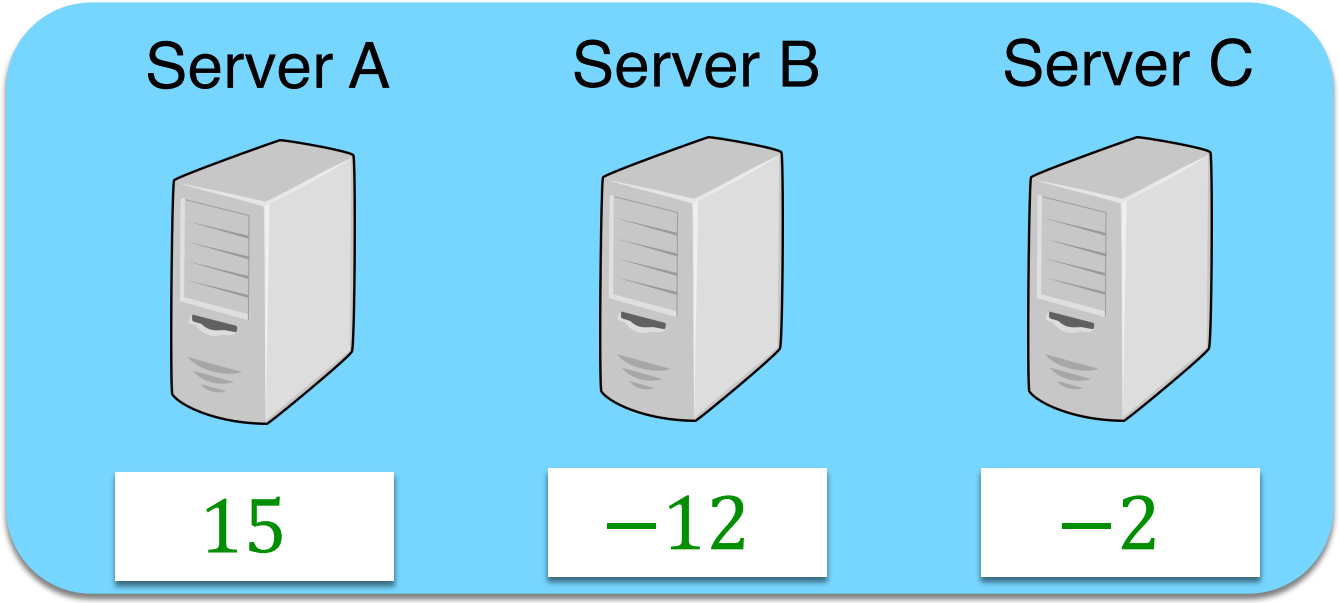


Pick three random "shares" that sum to $x_1 = 1$.

$$1 = 15 + (-12) + (-2) \pmod{p}$$

Send one share to each server.

Straw-man scheme
Private sums without disruption resistance



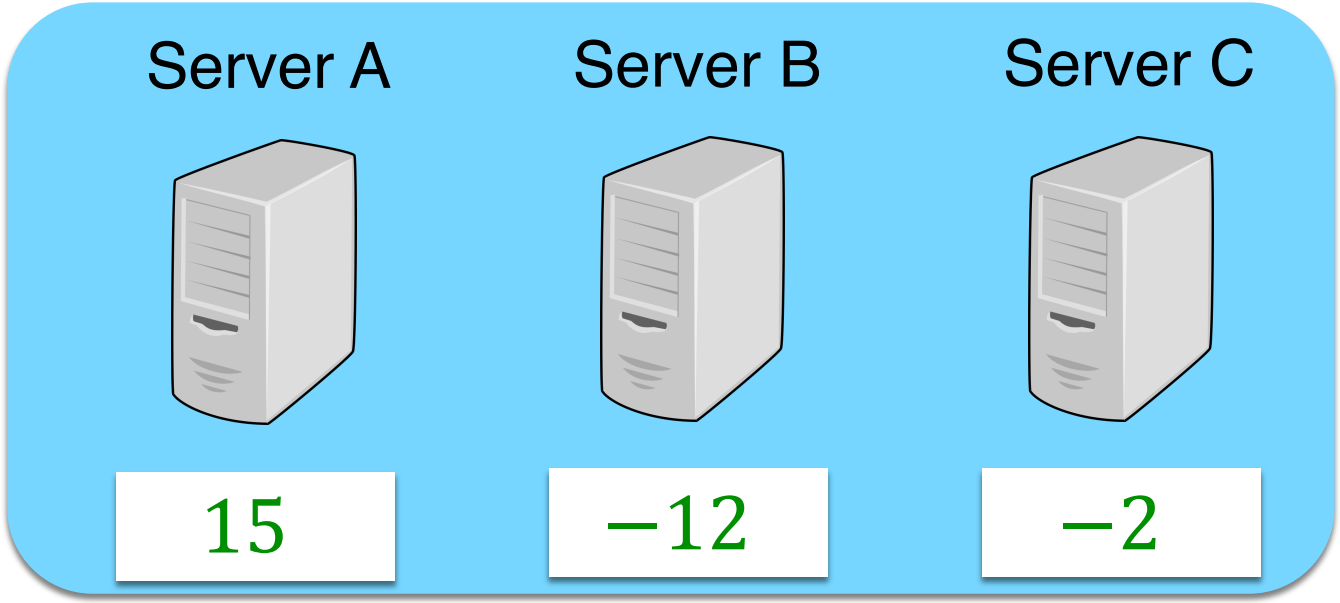
$x_1 = \mathbf{1}$

Pick three random "shares" that sum to $x_1 = \mathbf{1}$.

$$\mathbf{1} = 15 + (-12) + (-2) \pmod{p}$$

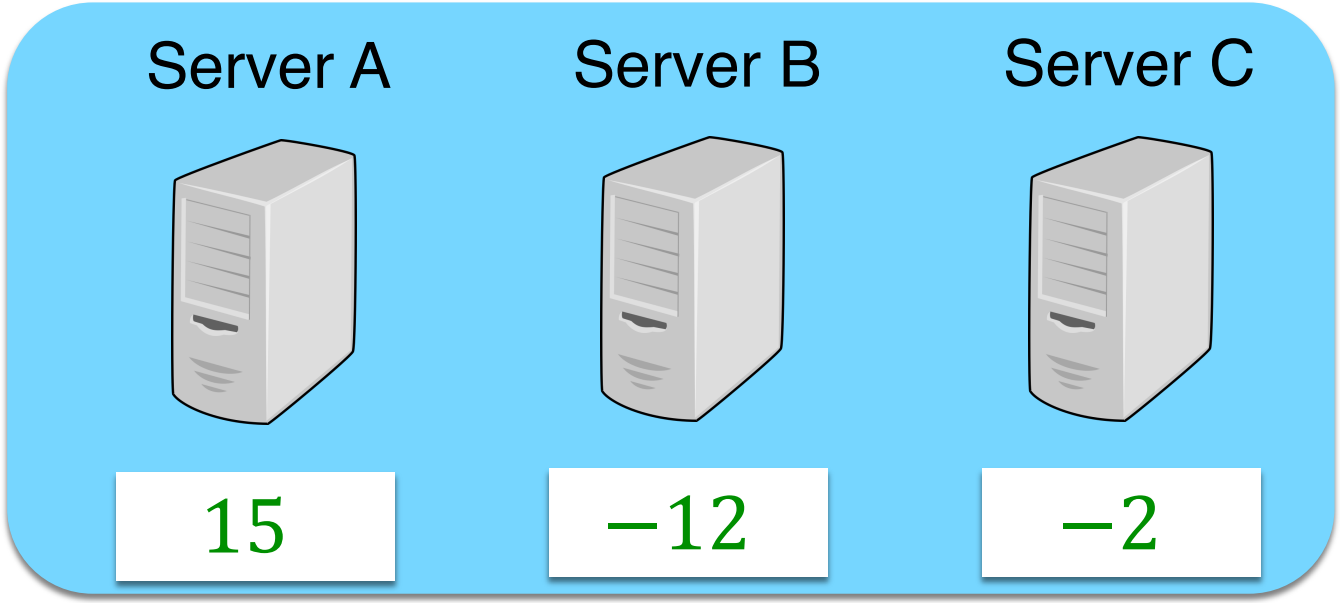
Send one share to each server.

Straw-man scheme
Private sums without disruption resistance



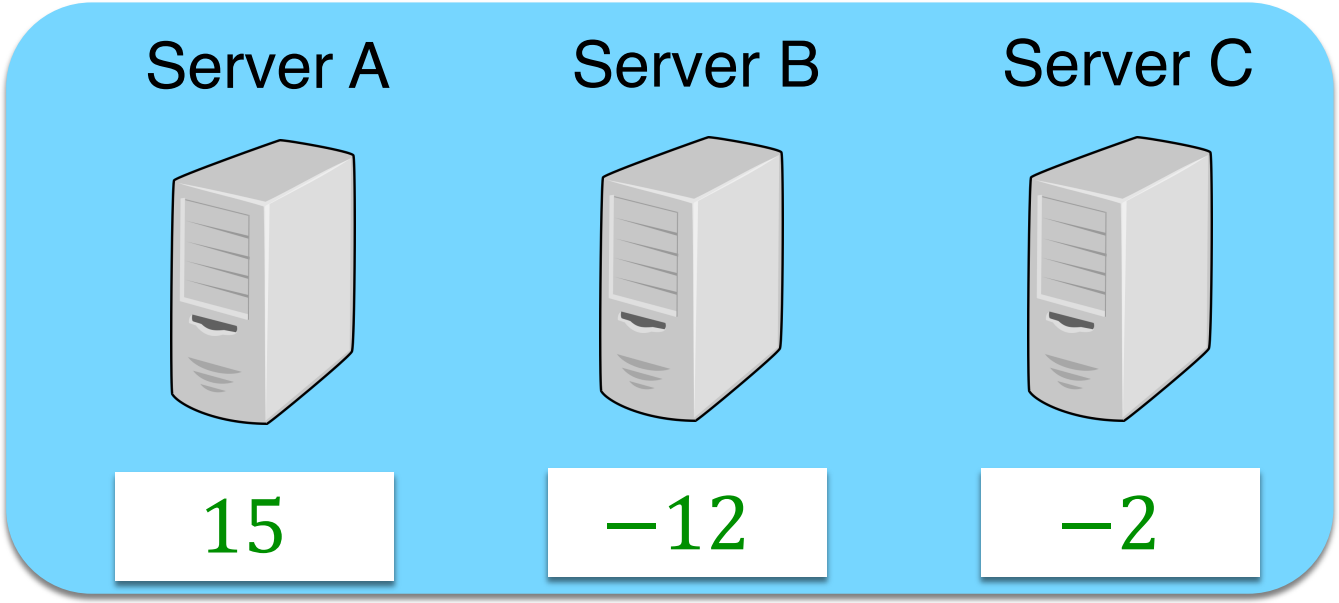
$x_2 = \mathbf{0}$

Straw-man scheme
Private sums without disruption resistance



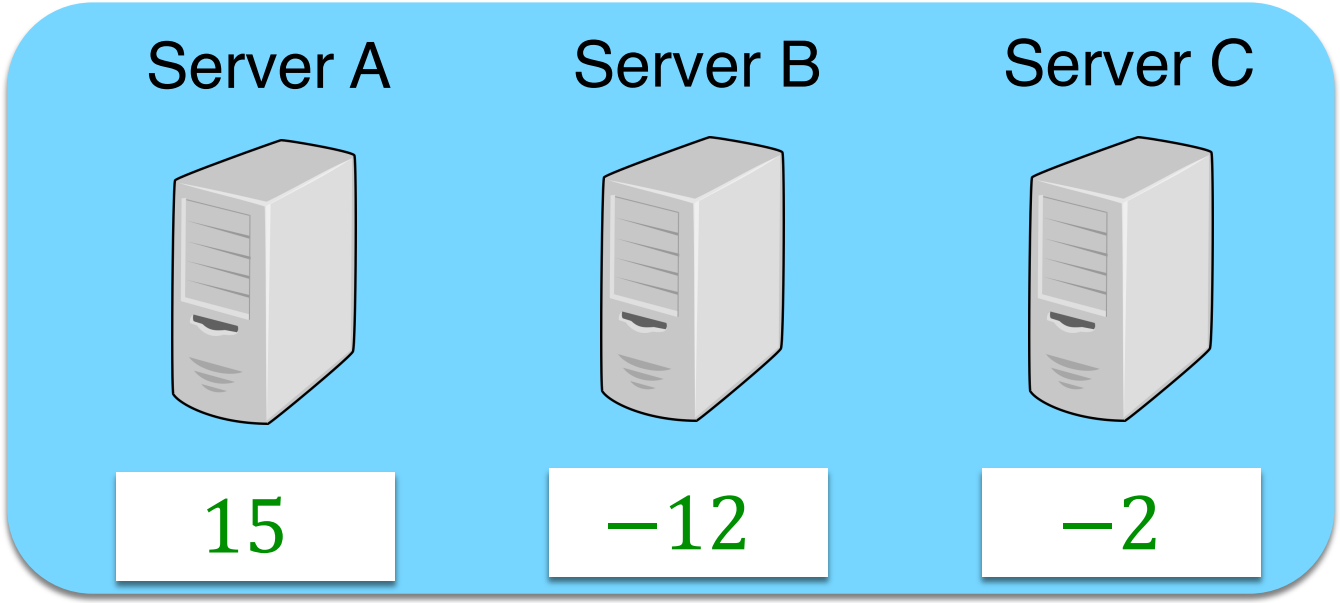
$x_2 = \mathbf{0}$

Straw-man scheme
Private sums without disruption resistance



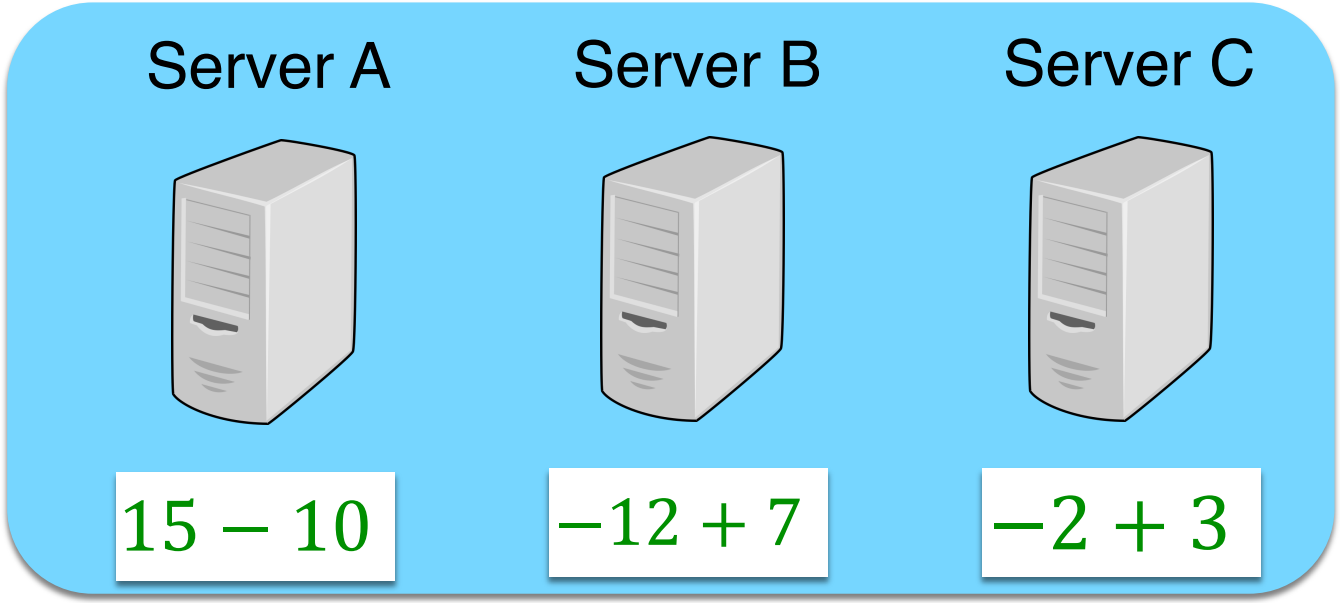
$$x_2 = \mathbf{0} = (-10) + 7 + 3$$

Straw-man scheme
Private sums without disruption resistance



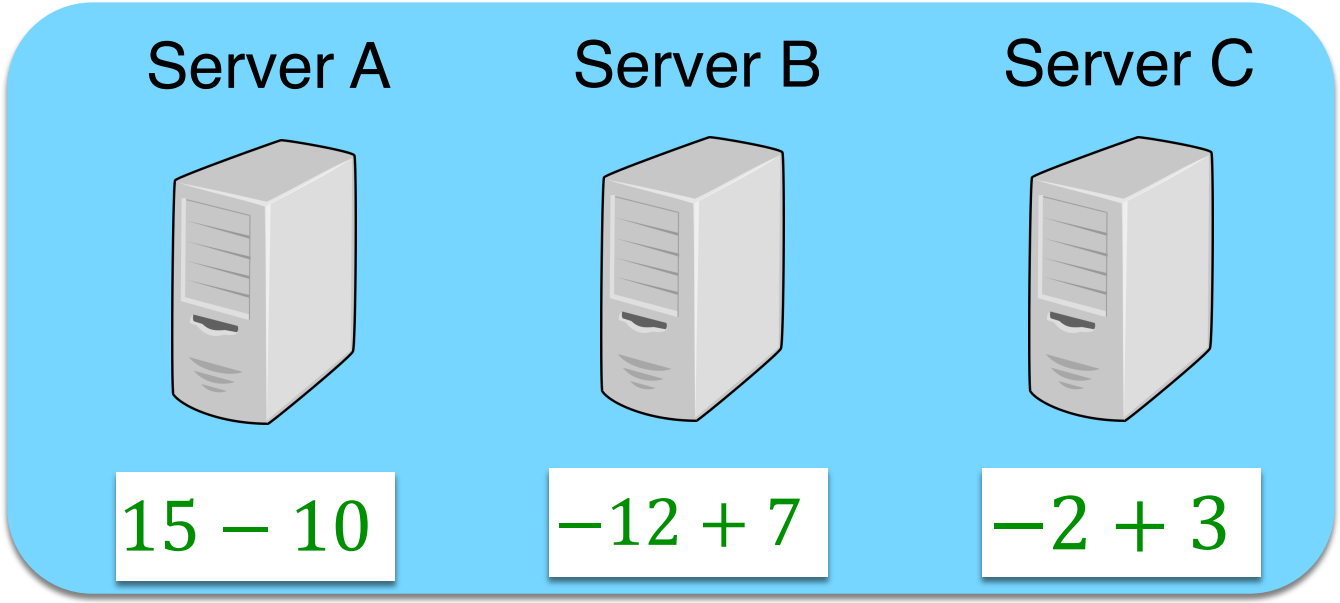
$x_2 = \mathbf{0}$ $\quad -10 \quad 7 \quad 3$

Straw-man scheme
Private sums without disruption resistance

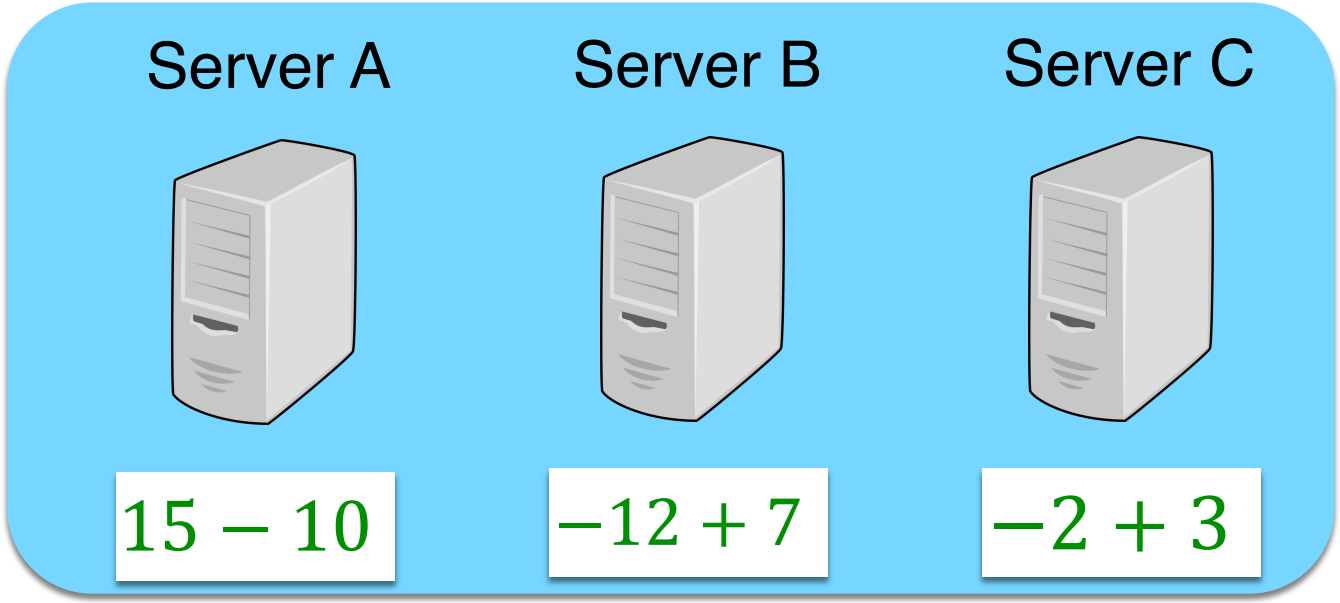


$x_2 = \mathbf{0}$

Straw-man scheme
Private sums without disruption resistance

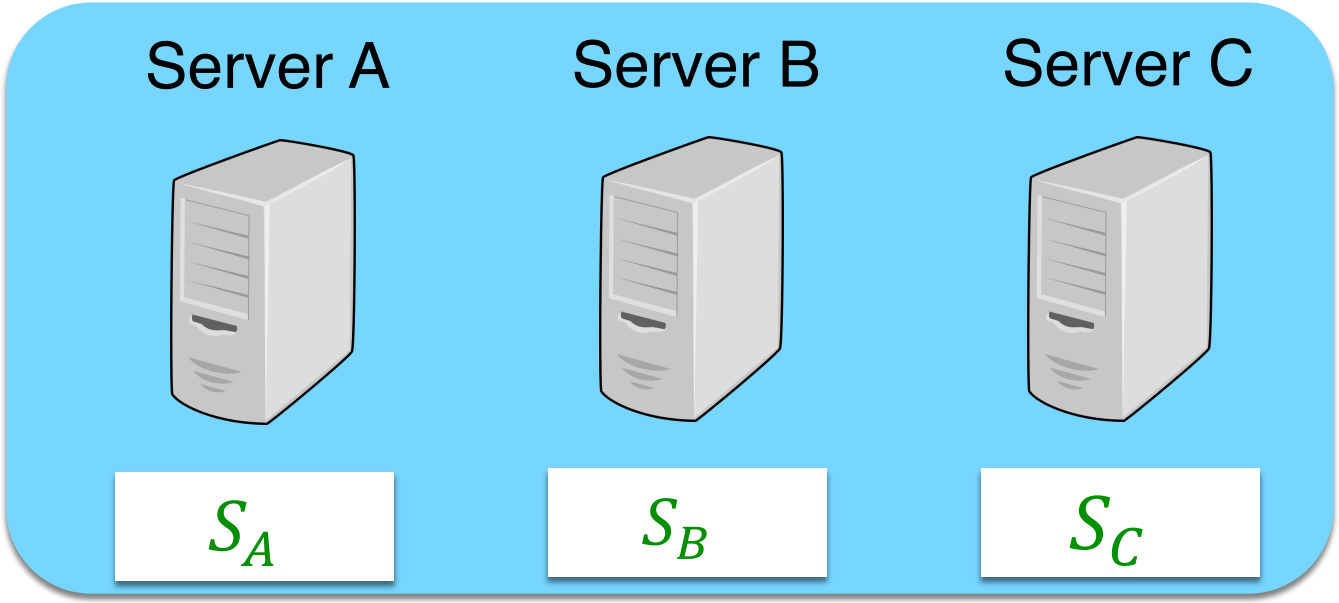


Straw-man scheme
Private sums without disruption resistance

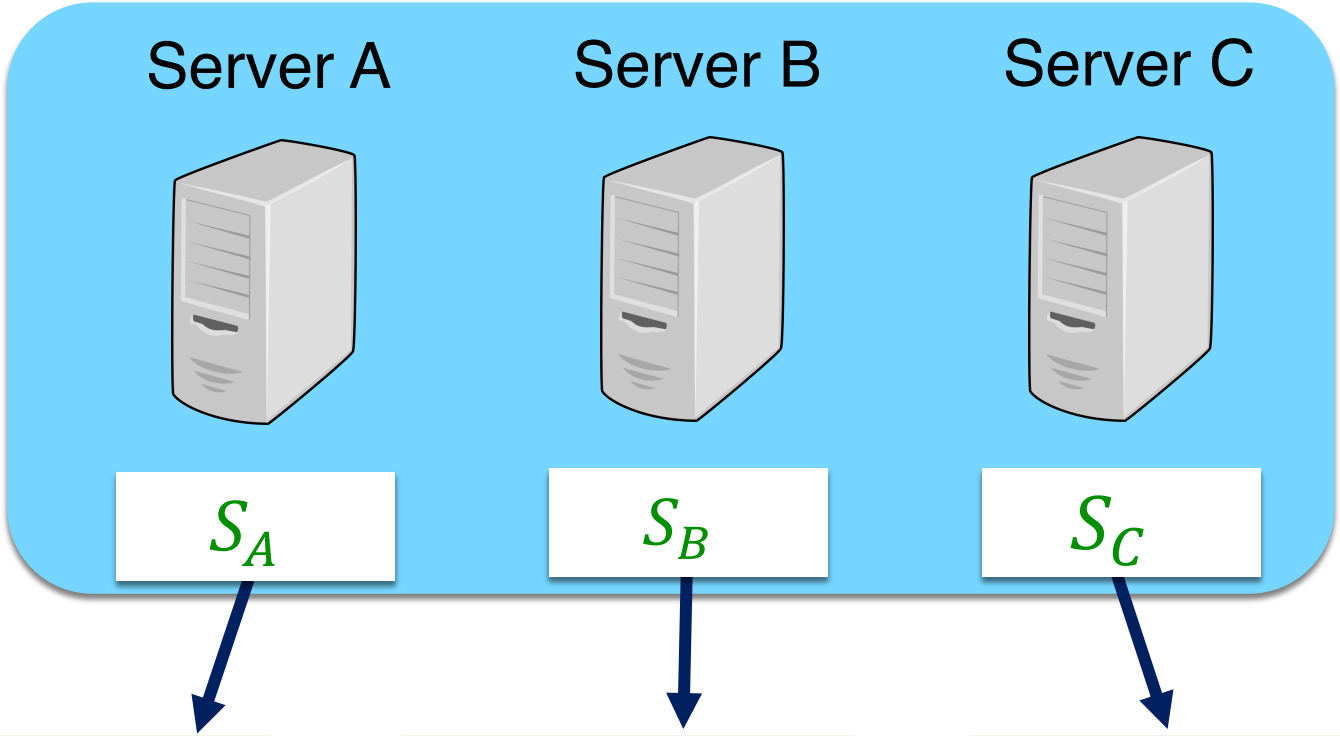


■ ■ ■

Straw-man scheme
Private sums without disruption resistance



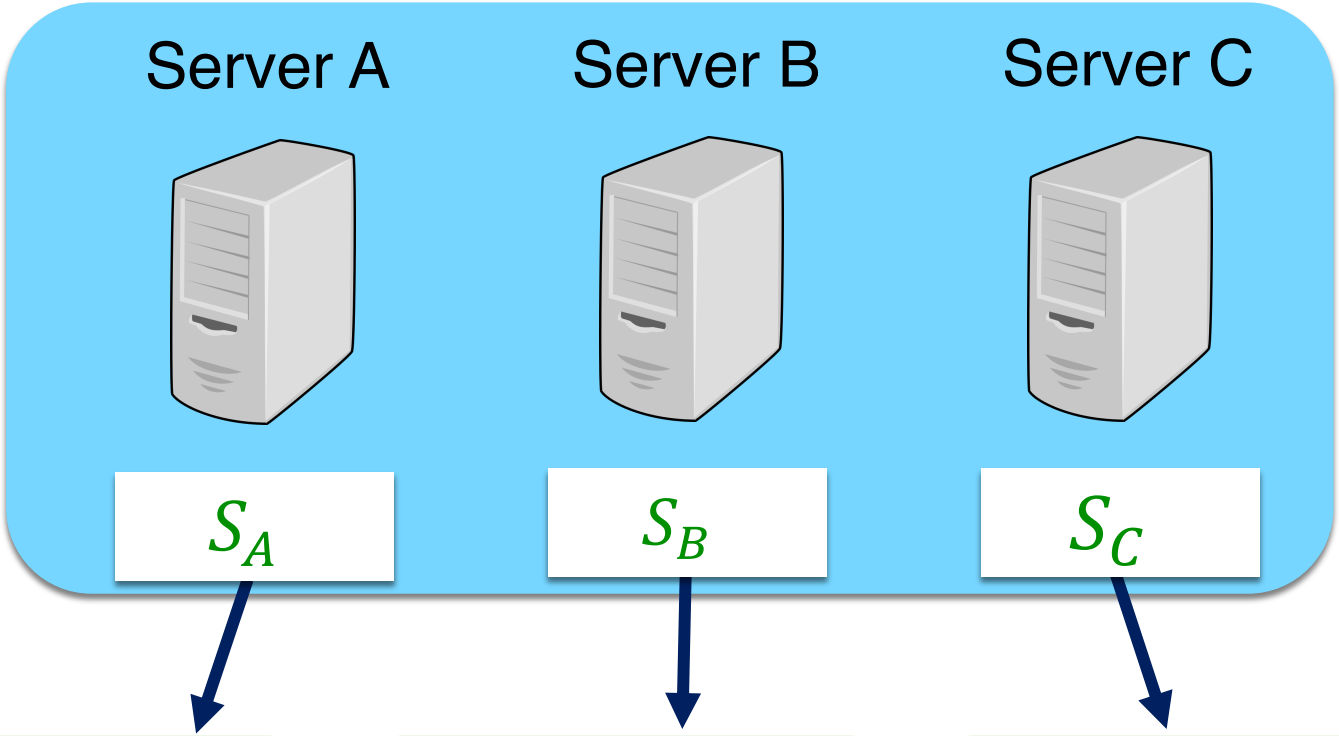
Straw-man scheme
Private sums without disruption resistance



$$(15 - 10 + \cdots) + (-12 + 7 + \cdots) + (-2 + 3 + \cdots)$$
$$= \quad x_1 + x_2 + x_3 + \cdots$$

Servers learn the sum of the clients' values and nothing else.

Straw-man scheme
Private sums without disruption resistance



$$(15 - 10 + \cdots) + (-12 + 7 + \cdots) + (-2 + 3 + \cdots)$$
$$= \quad x_1 + x_2 + x_3 + \cdots$$

e.g., learn that 58,329 users blocked trackers from fb.com… **don't** learn which users did

Servers learn the sum of the clients' values and nothing else.

Private sums: Straw-man scheme

✓ **Correctness.** Servers learn the sum of the x_is

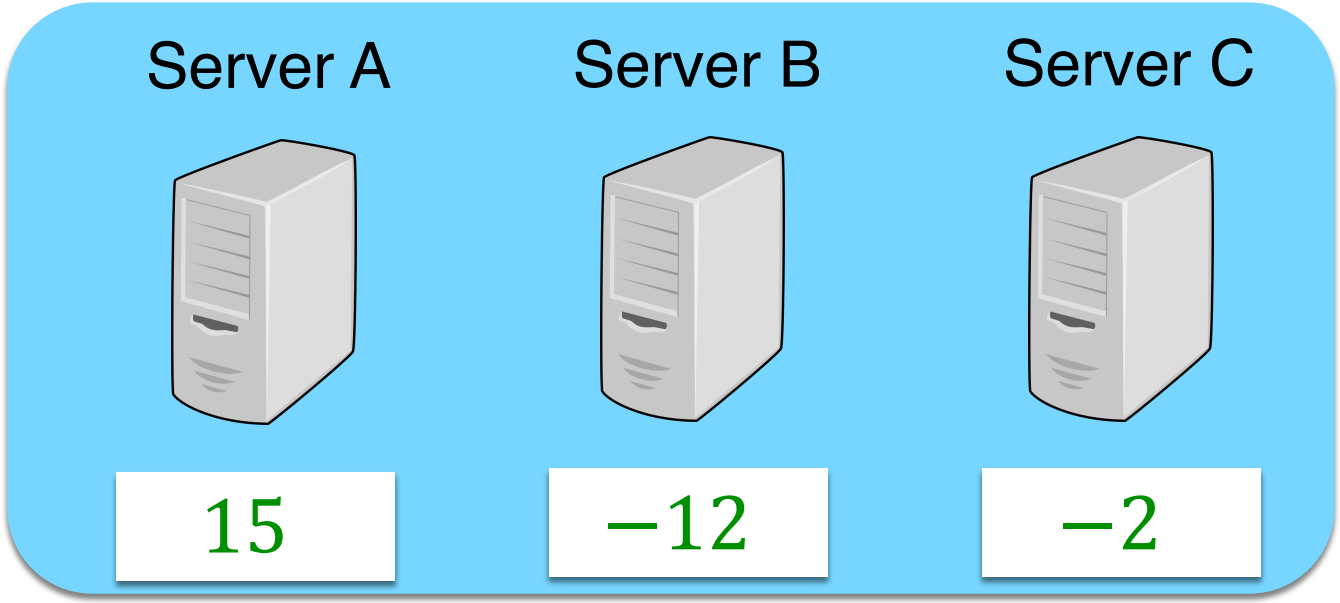
✓ **f-Privacy.** Attacker must compromise <u>all</u> servers to learn more than sum of x_is

✓ **Efficiency.** No heavy cryptographic operations

✗ **Disruption resistance.** **One malicious client can corrupt the output.**

Straw-man scheme

One malicious client can corrupt output

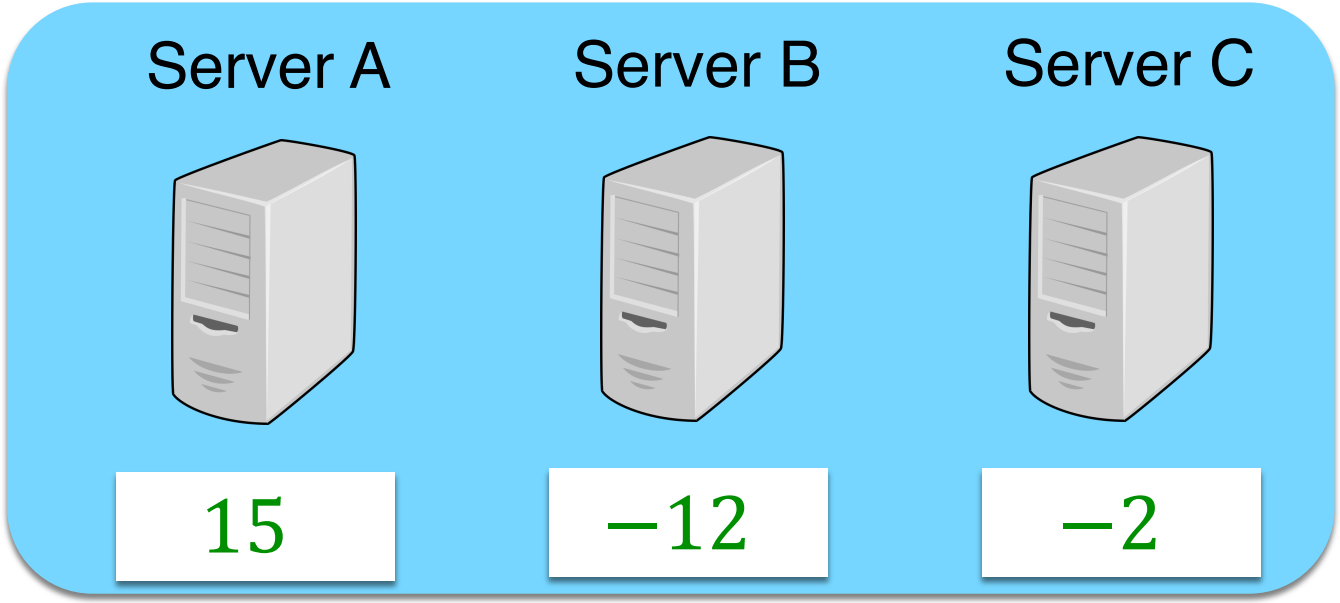


Evil ad network

Should be a value in the set $\{0,1\}$

Straw-man scheme

One malicious client can corrupt output



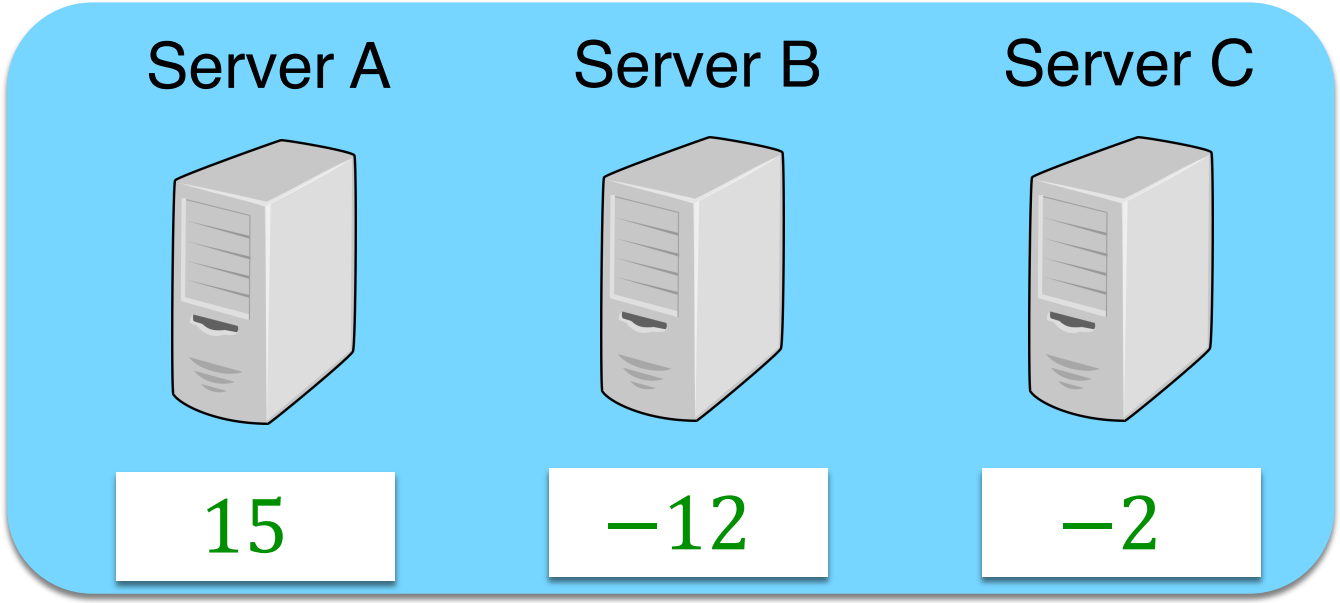
$$x_2 = -53 = -19 + -16 + -18$$

Should be a value in the set $\{0,1\}$

Evil ad network

Straw-man scheme

One malicious client can corrupt output



$x_2 = -53 \quad -19 \quad -16 \quad -18$

Evil ad network

Should be a value in the set $\{0,1\}$

Straw-man scheme

One malicious client can corrupt output

Evil ad network

Powerful but costly tools...

Multiparty computation

[GMW87], [BGW88]

Powerful but costly tools...

Multiparty computation

[GMW87], [BGW88]

Traditional zero-knowledge proofs

[GMR89]

Powerful but costly tools...

Multiparty computation

[GMW87], [BGW88]

Traditional zero-knowledge proofs

[GMR89]

New tool: Proof on secret-shared data

Techniques for providing disruption resistance

Testing that a length-n vector (e.g., data for n trackers) consists of secret-shared 0/1 integers.

	Public-key ops.		Communication		Slow-down
	Client	Server	C-to-S	S-to-S	
Dishonest-maj. MPC	0	$\widetilde{\Theta}(n)$	0	$\widetilde{\Theta}(n)$	5,000× at server
GGPR-style zkSNARK	$\widetilde{\Theta}(n)$	$\tilde{O}(1)$	$\tilde{O}(1)$	$\tilde{O}(1)$	500× at client
Discrete-log-based NIZK	$\widetilde{\Theta}(n)$	$\widetilde{\Theta}(n)$	$\widetilde{\Theta}(n)$	$\widetilde{\Theta}(n)$	50× at server

(Table hides log factors.)

Techniques for providing disruption resistance

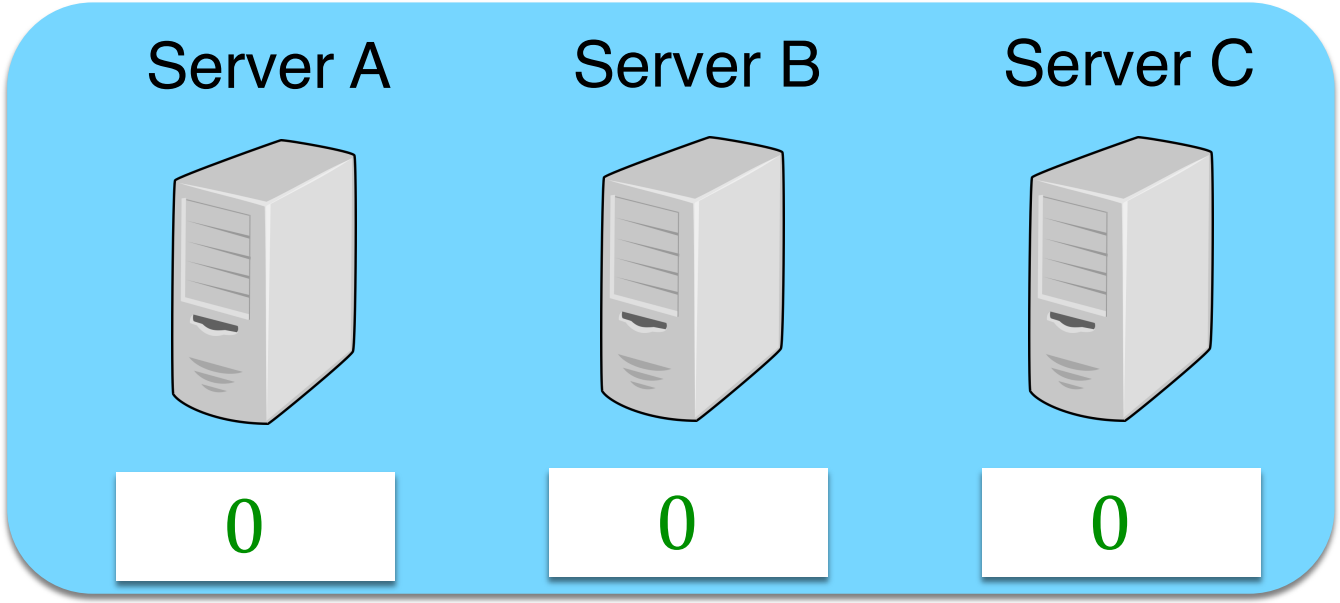
Testing that a length-n vector (e.g., data for n trackers) consists of secret-shared 0/1 integers.

	Public-key ops.		Communication		Slow-down
	Client	Server	C-to-S	S-to-S	
Dishonest-maj. MPC	0	$\widetilde{\Theta}(n)$	0	$\widetilde{\Theta}(n)$	5,000× at server
GGPR-style zkSNARK	$\widetilde{\Theta}(n)$	$\tilde{O}(1)$	$\tilde{O}(1)$	$\tilde{O}(1)$	500× at client
Discrete-log-based NIZK	$\widetilde{\Theta}(n)$	$\widetilde{\Theta}(n)$	$\widetilde{\Theta}(n)$	$\widetilde{\Theta}(n)$	50× at server
Prio (latest version)	0	0	$\tilde{O}(1)$	$\tilde{O}(1)$	1×

(Table hides log factors.)

Contribution:

Prevent disruption using proofs on secret-shared data

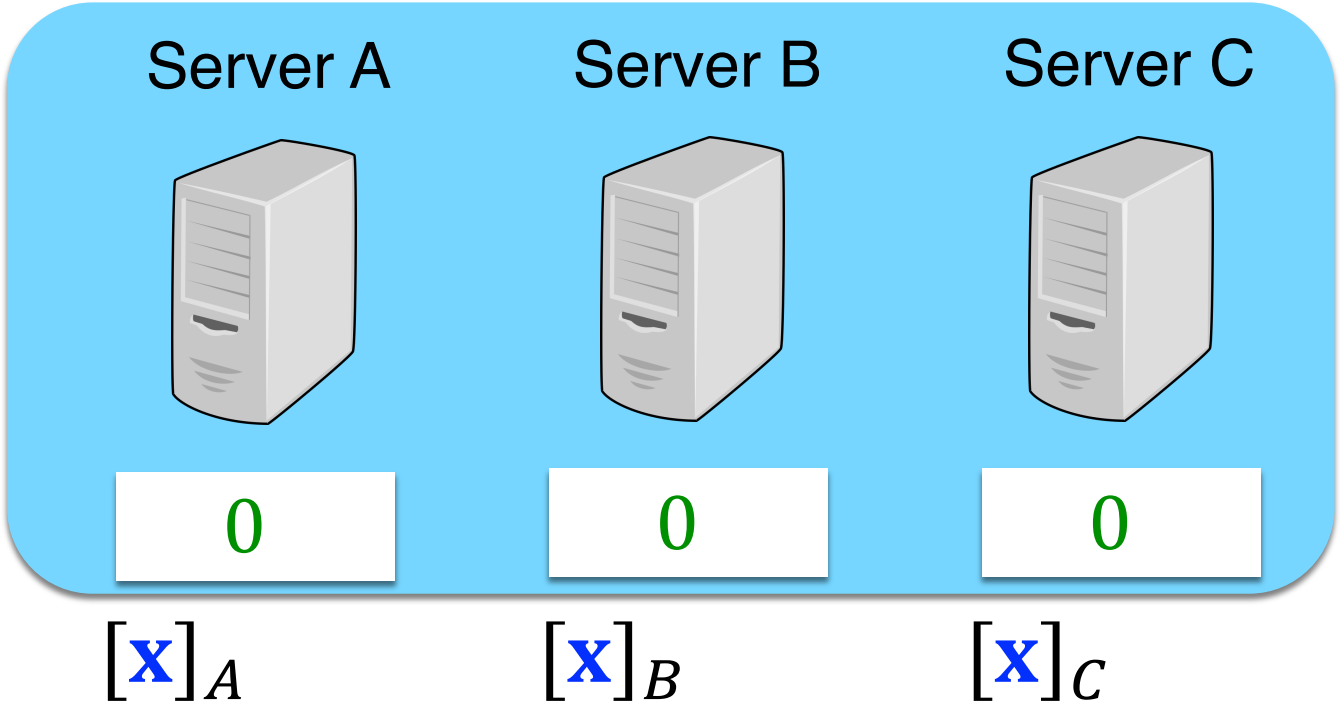


$\mathbf{x} \in \{0,1\}^n$

Data for n domains

Contribution:

Prevent disruption using proofs on secret-shared data

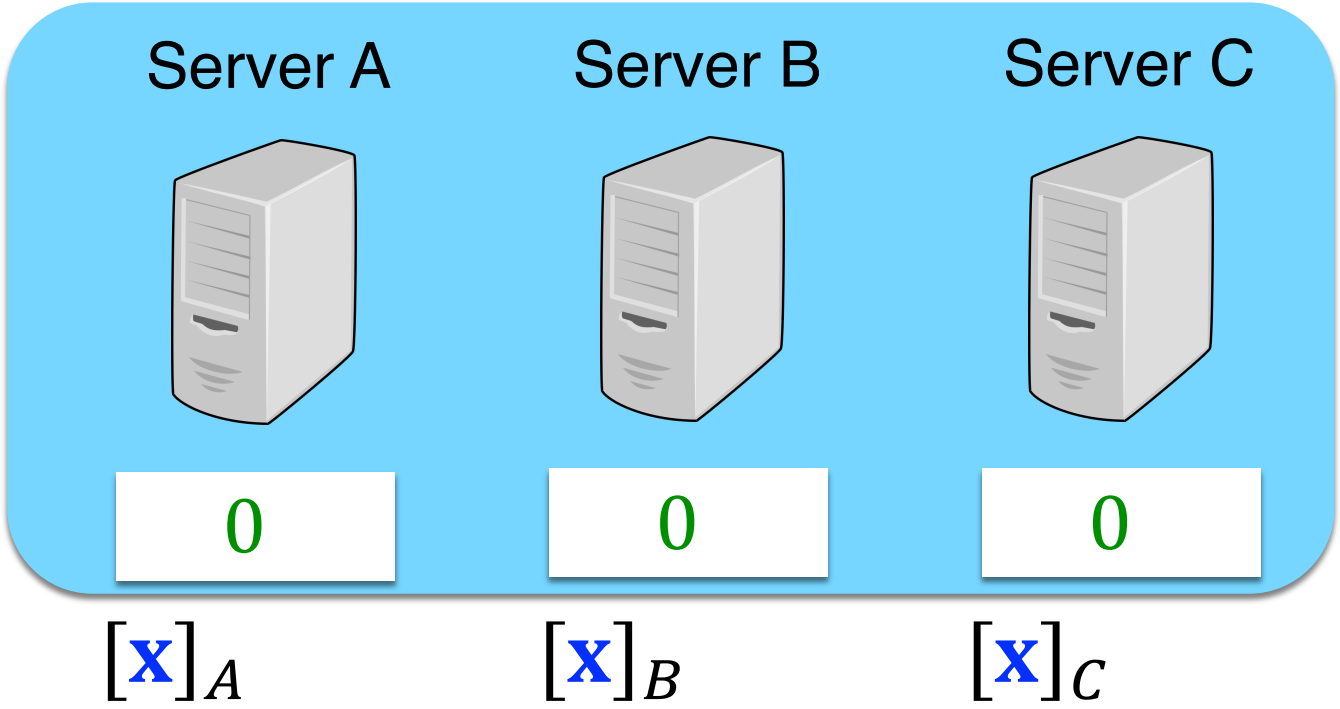


$\mathbf{x} \in \{0,1\}^n$

Data for n domains

Contribution:

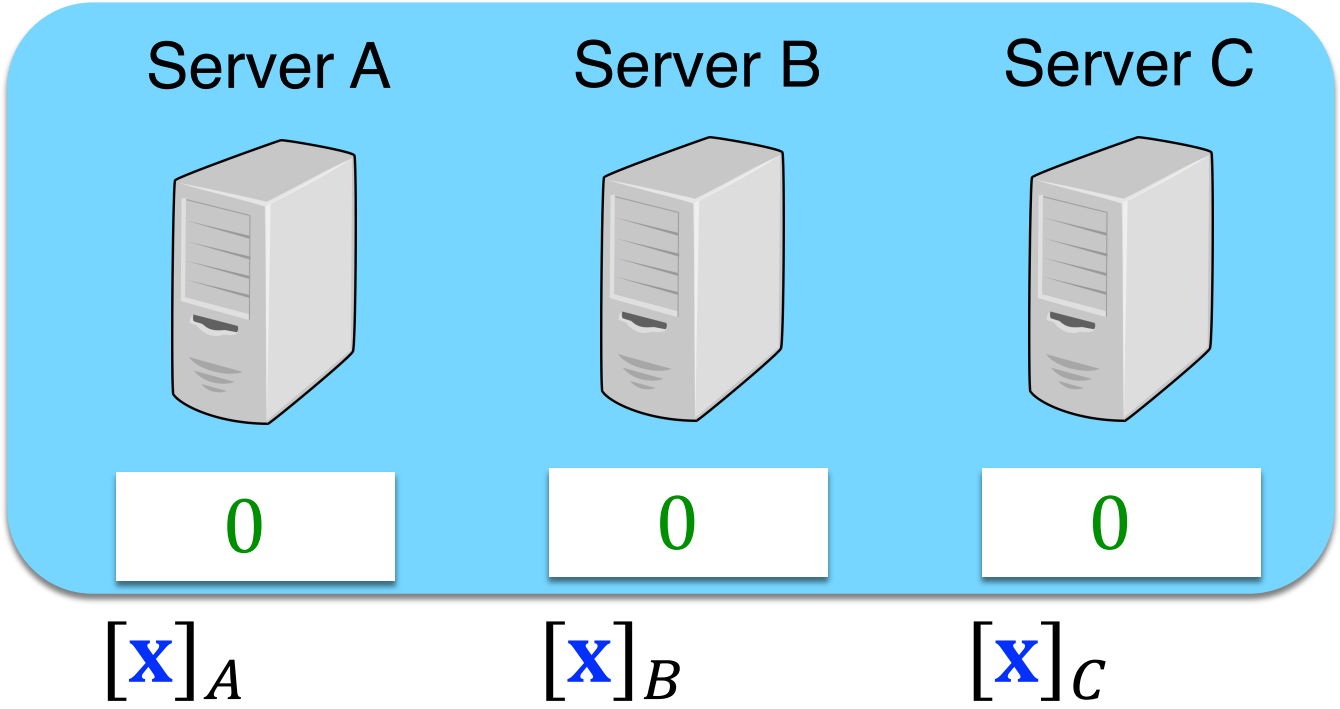
Prevent disruption using proofs on secret-shared data



$\mathbf{x} \in \{0,1\}^n$

Data for n domains

Contribution:
Prevent disruption using proofs on secret-shared data



$\mathbf{x} \in \{0,1\}^n$

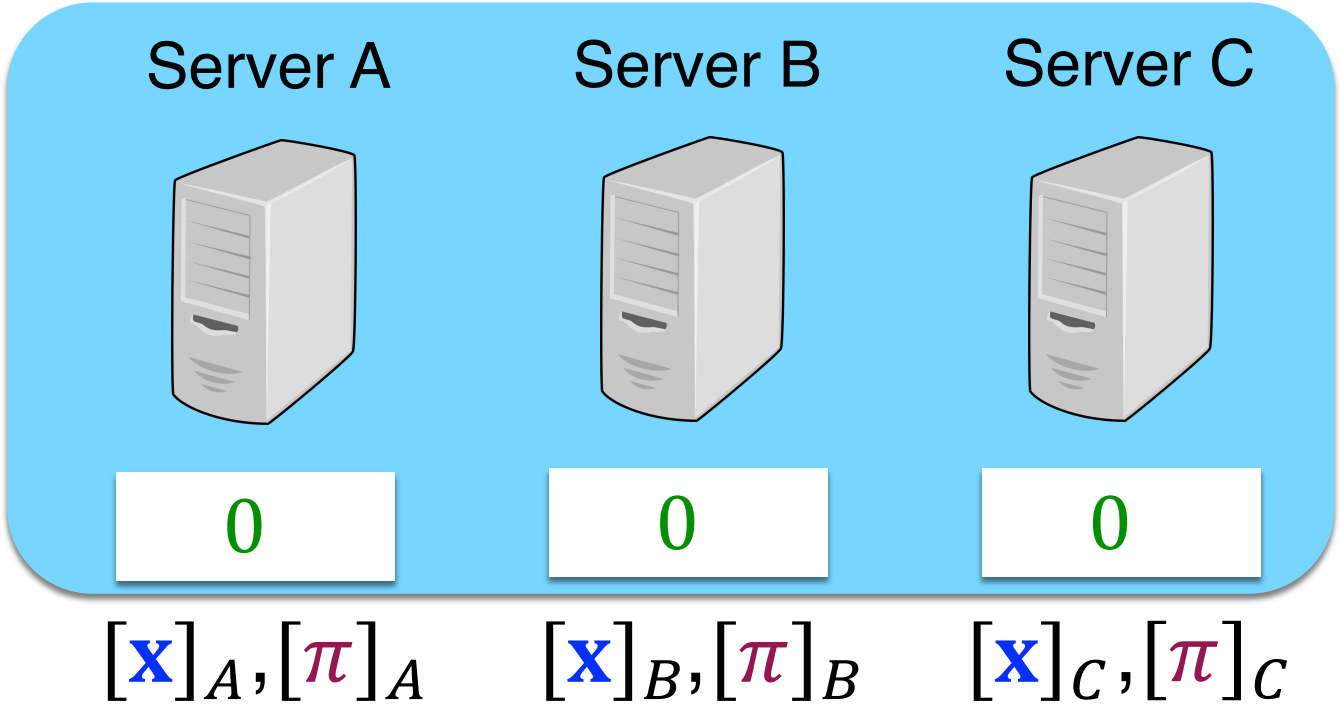
Data for n domains

Want to be convinced that

$\mathbf{x} = [\mathbf{x}]_A + [\mathbf{x}]_B + [\mathbf{x}]_C \in \{0,1\}^n \quad \in \mathbb{Z}_p^n$

More generally, that $\mathrm{Valid}(\mathbf{x})$ holds, for some predicate Valid

Contribution:
Prevent disruption using proofs on secret-shared data

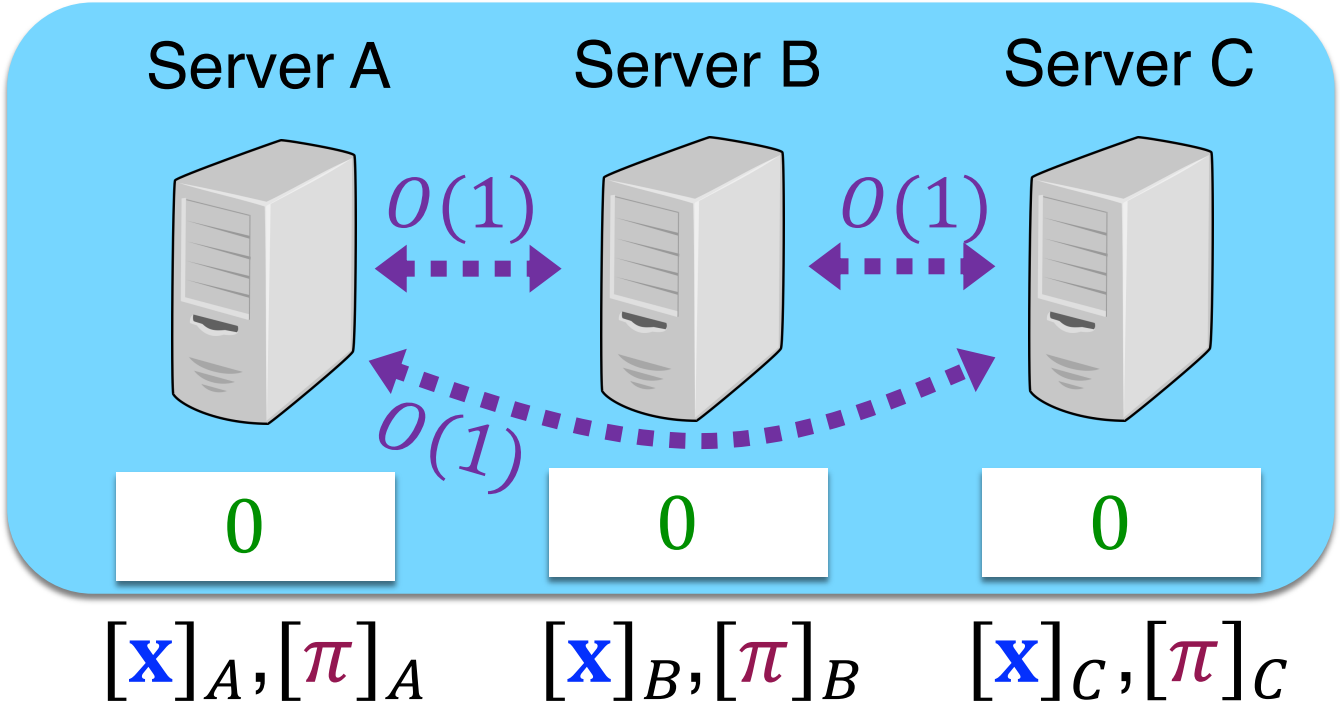


$\mathbf{x} \in \{0,1\}^n$

Data for n domains

- Client sends proof to servers that $\mathbf{Valid}(\mathbf{x})$ holds
  - For our example, $\mathbf{Valid}(\mathbf{x}) =$ "$\mathbf{x} \in \{0,1\}^n$"
  - Servers exchange $O(1)$ bytes to check proof
- Prevents disruption in Prio
  - Servers reject invalid client submissions

Contribution:
Prevent disruption using proofs on secret-shared data



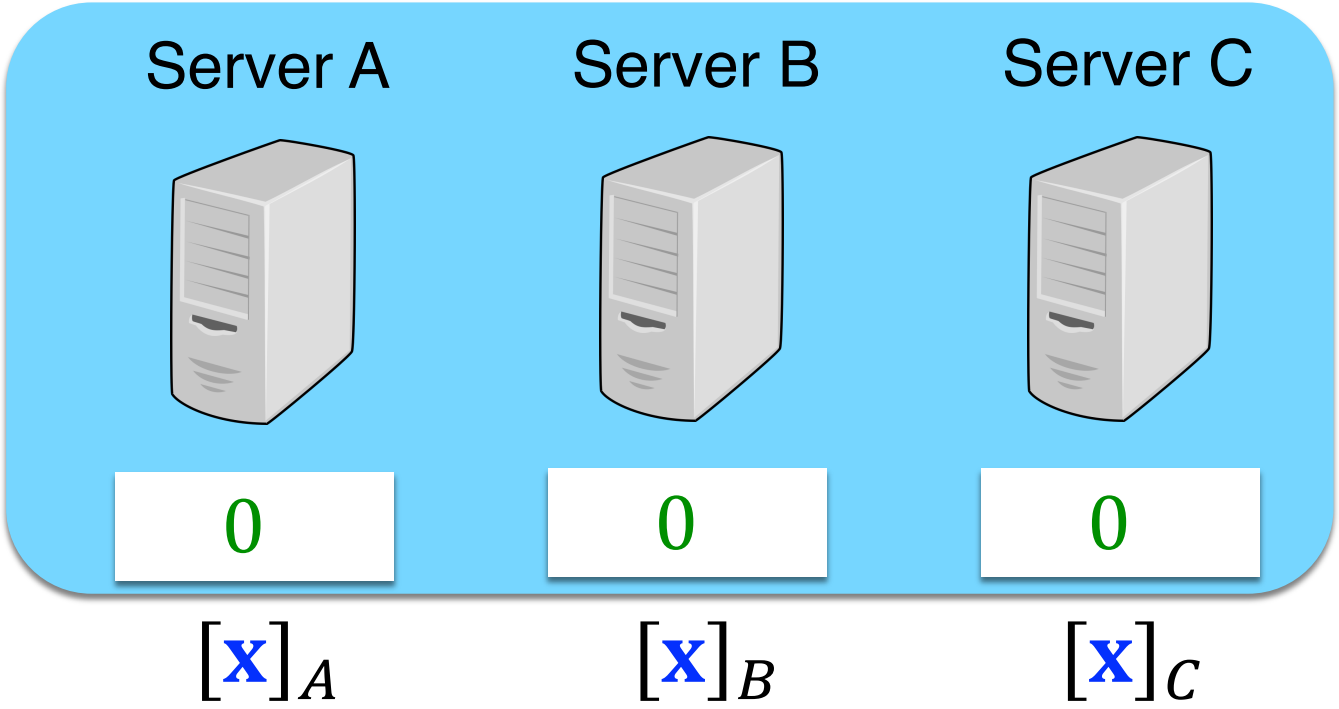
$\mathbf{x} \in \{0,1\}^n$

Data for n domains

- Client sends proof to servers that $\mathbf{Valid}(\mathbf{x})$ holds
  - For our example, $\mathbf{Valid}(\mathbf{x}) =$ "$\mathbf{x} \in \{0,1\}^n$"
  - Servers exchange $O(1)$ bytes to check proof
- Prevents disruption in Prio
  - Servers reject invalid client submissions

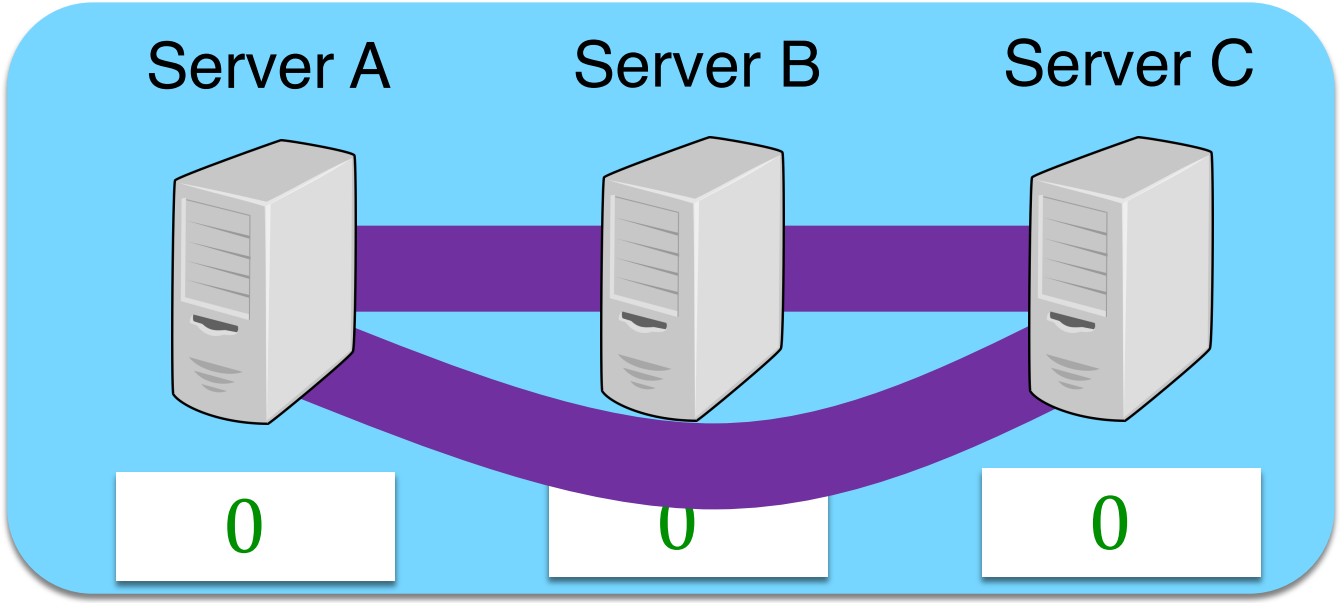
How to construct a proof on secret-shared data*

*simplified



How to construct a proof on secret-shared data*

*simplified



Could use secure multi-party computation to check that $\mathrm{Valid}(\mathbf{x})$ holds

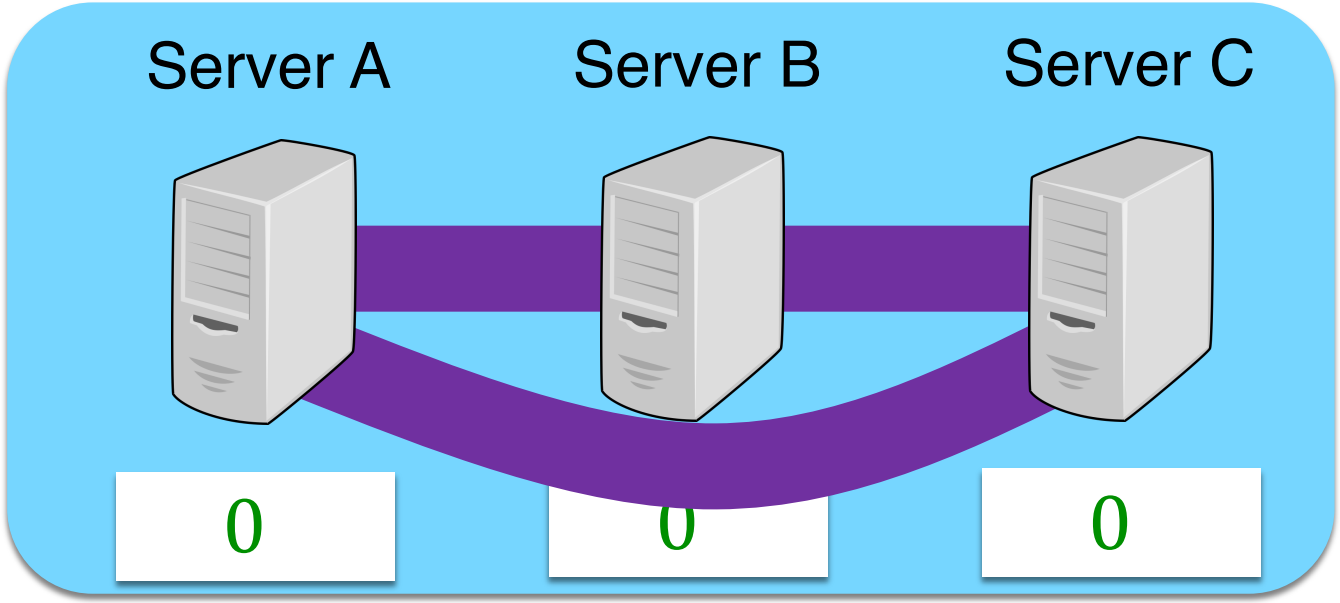
[GMW87], [BGW88], ...

How to construct a proof on secret-shared data*

*simplified

$\mathbf{x} \in \{0,1\}^n$

Data for n domains



$[\mathbf{x}]_A$ $[\mathbf{x}]_B$ $[\mathbf{x}]_C$

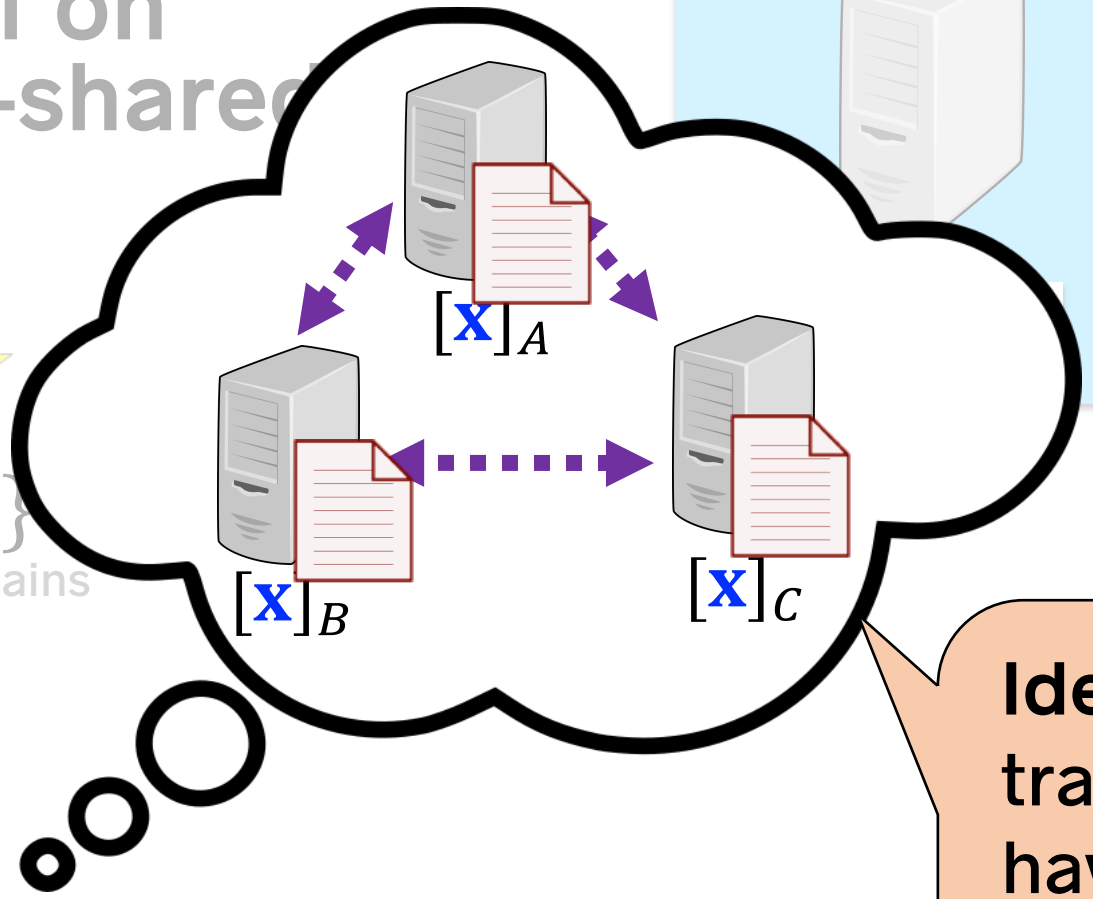
Could use secure multi-party computation to check that $\mathbf{Valid}(\mathbf{x})$ holds

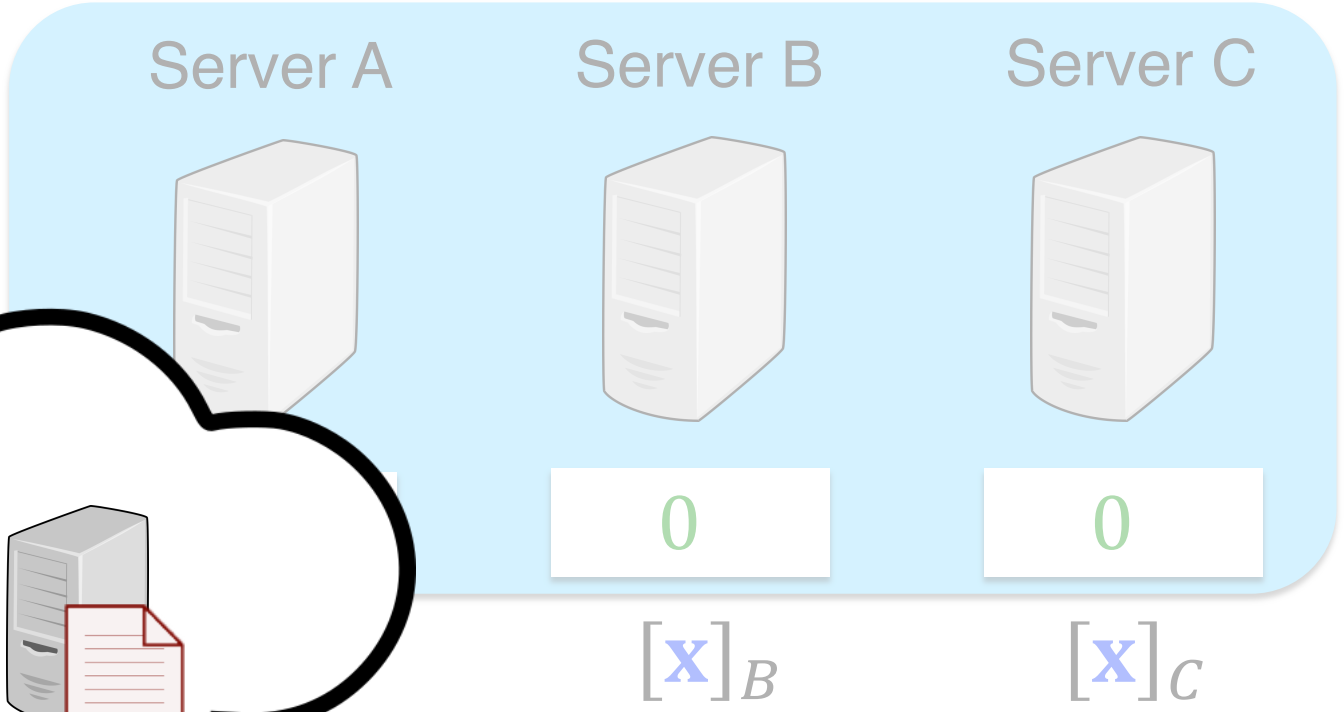
[GMW87], [BGW88], ...

How to construct a proof on secret-shared

$x \in \{0,1\}^n$

Data for n domains

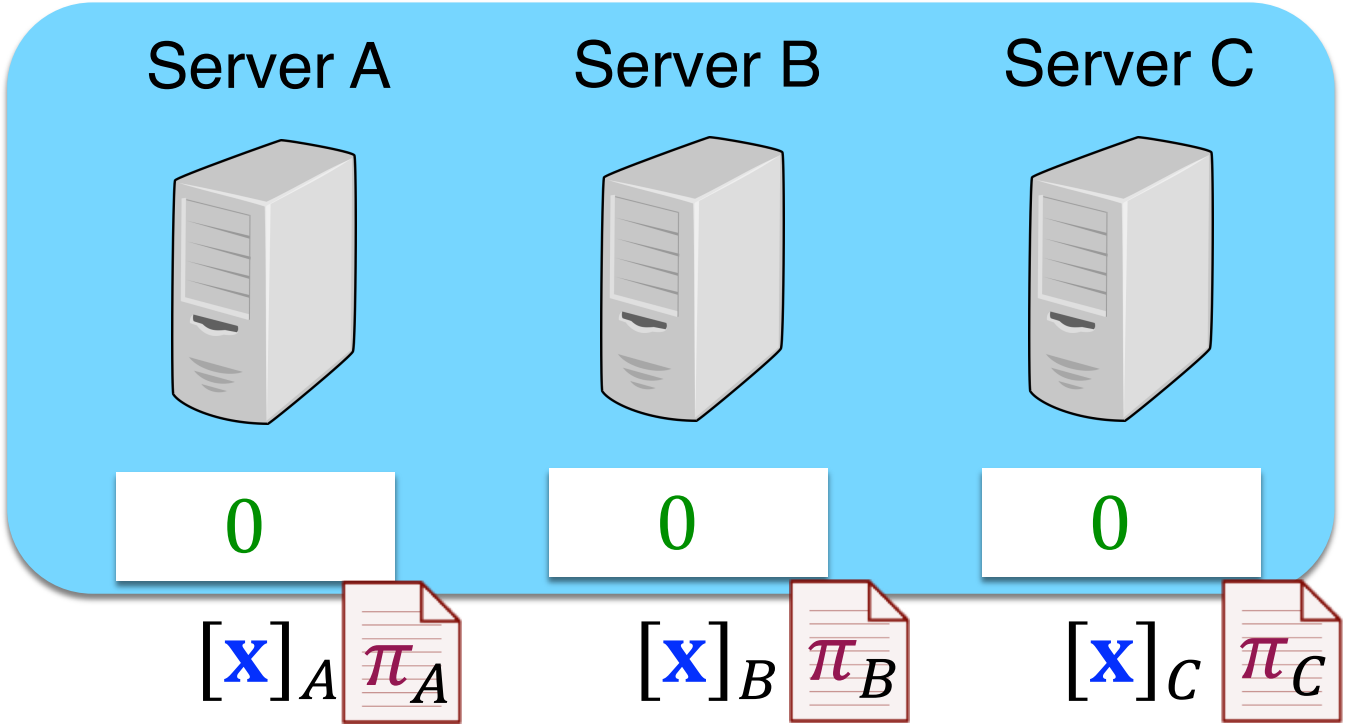




Idea: Client generates transcripts that servers would have observed in a multi-party computation of **Valid**(x).

See also [IKOS07]

How to construct a proof on secret-shared data



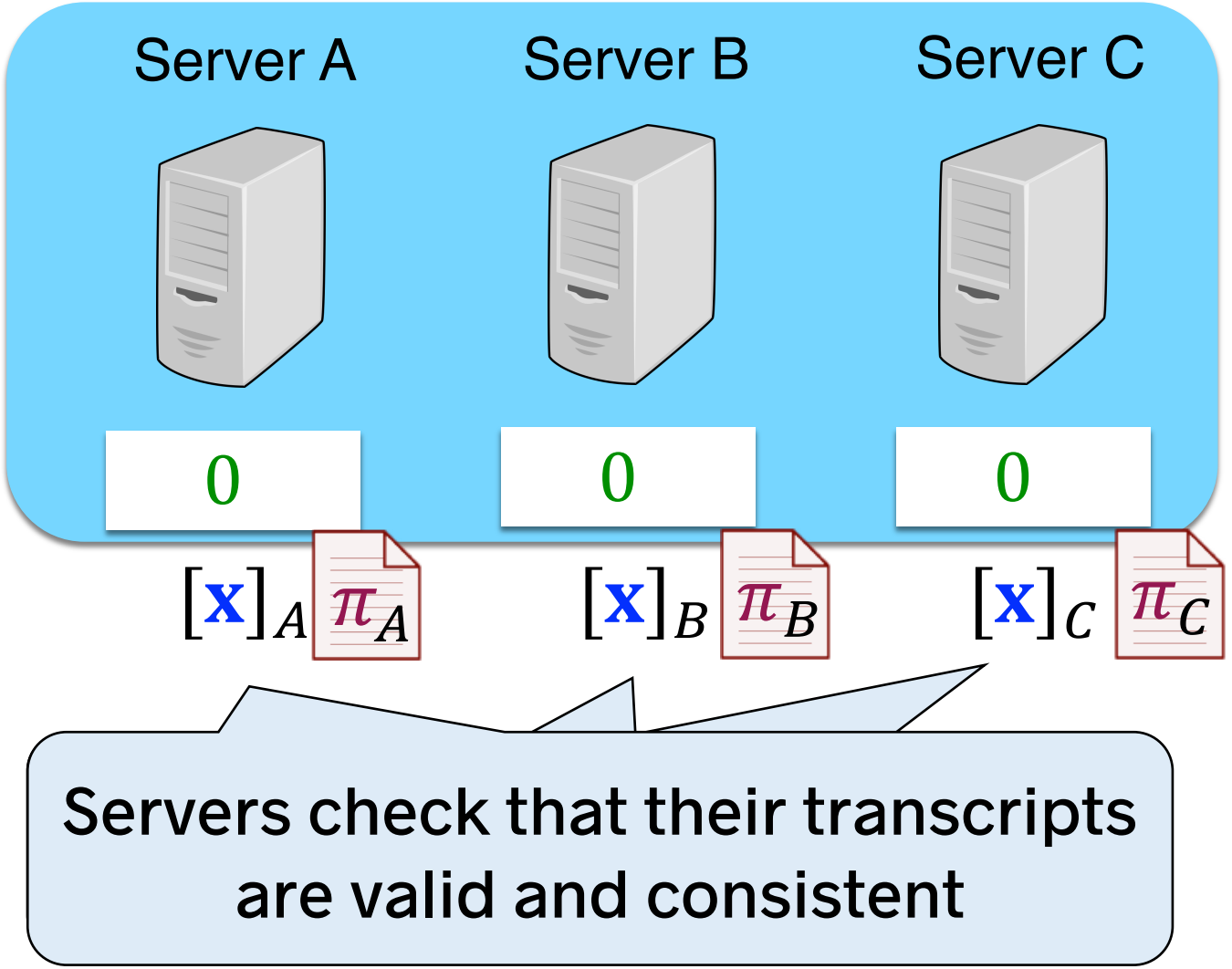
$\mathbf{x} \in \{0,1\}^n$

Data for n domains

How to construct a proof on secret-shared data

$\mathbf{x} \in \{0,1\}^n$

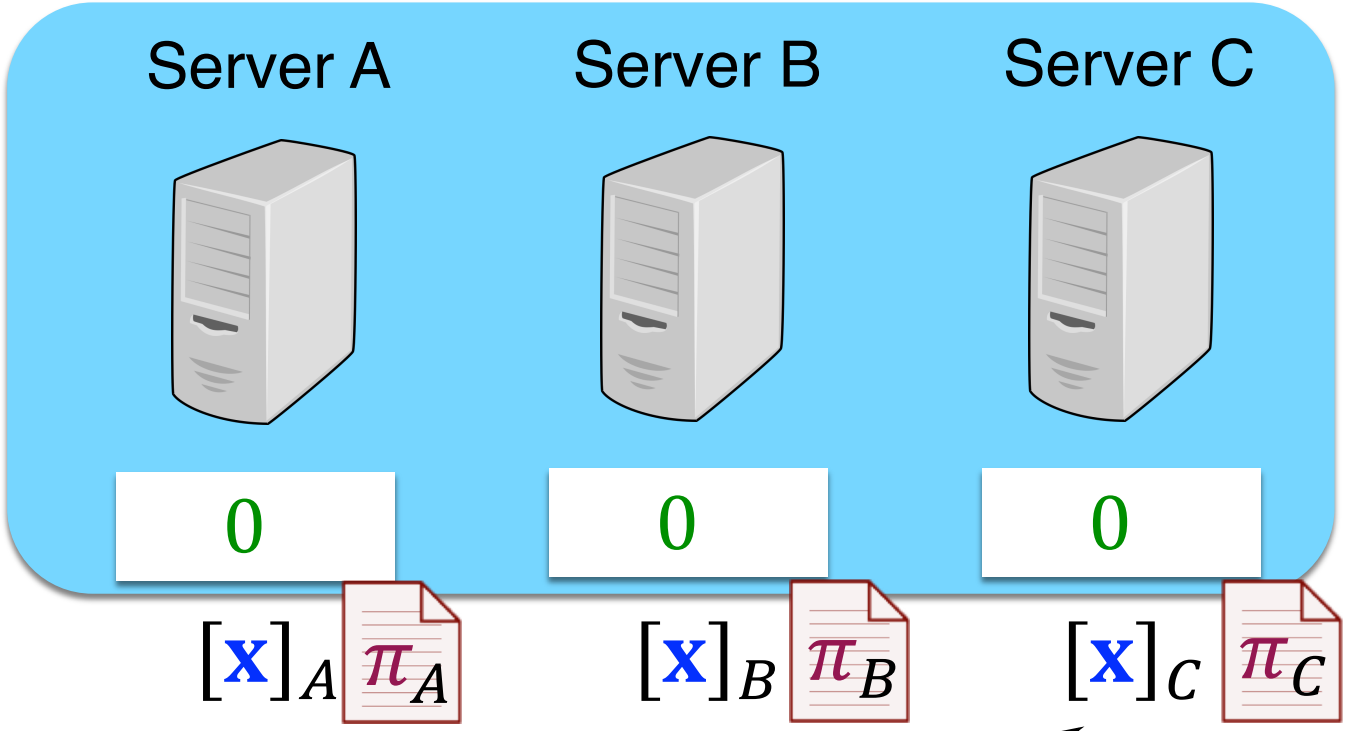
Data for n domains



How to construct a proof on secret-shared data

$\mathbf{x} \in \{0,1\}^n$

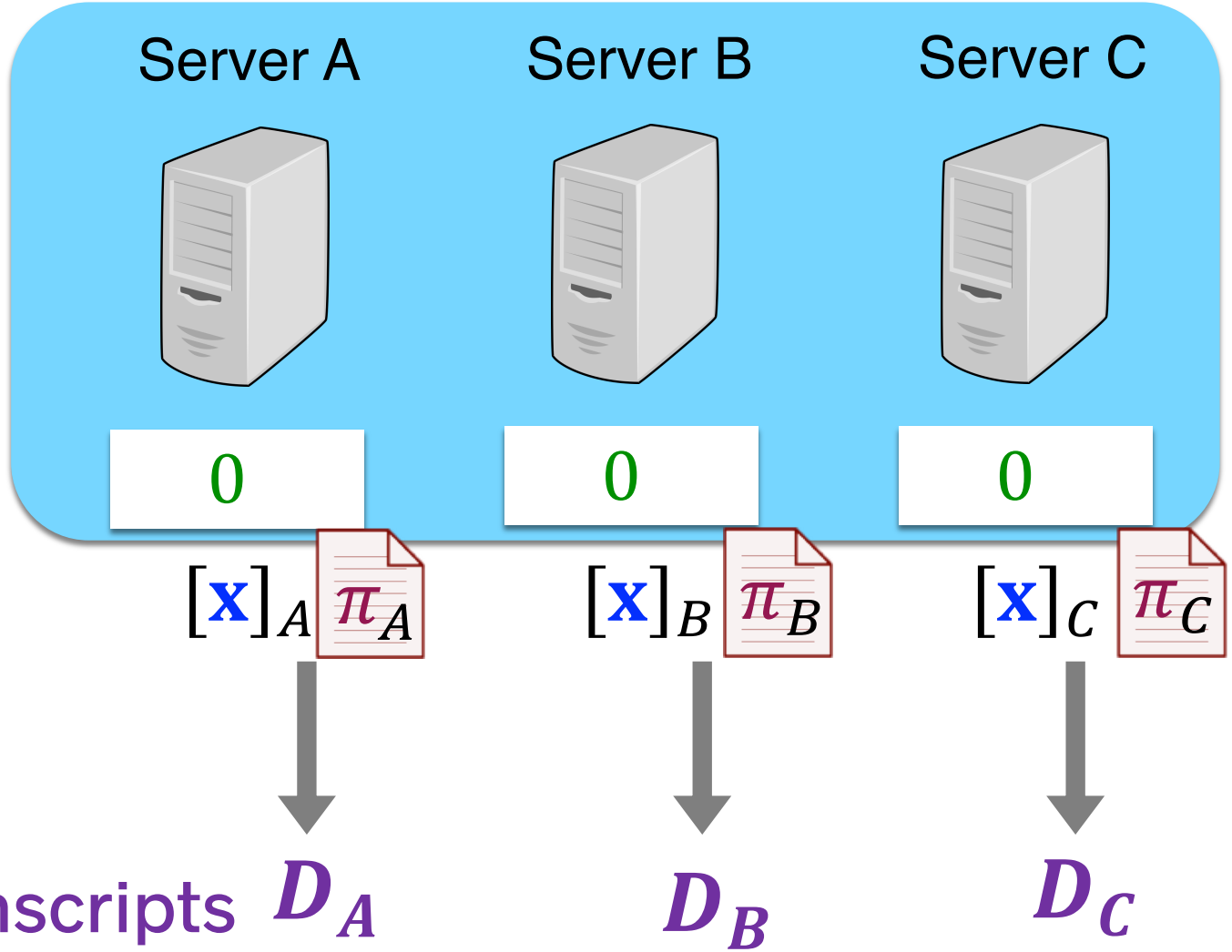
Data for n domains



Servers check that their transcripts are valid and consistent

Checking a transcript is **much easier** than generating one.

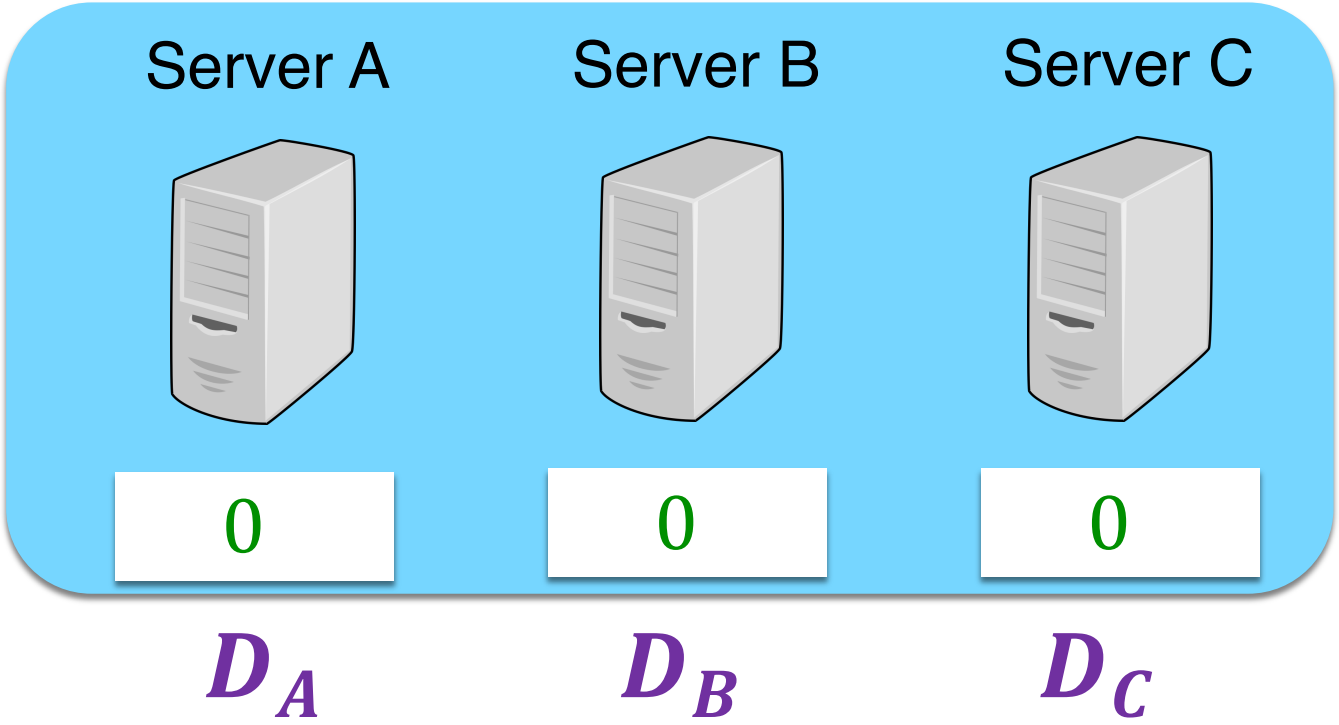
How to construct a proof on secret-shared data



"Randomized digest" of transcripts

(Leak nothing about client's value x)

How to construct a proof on secret-shared data

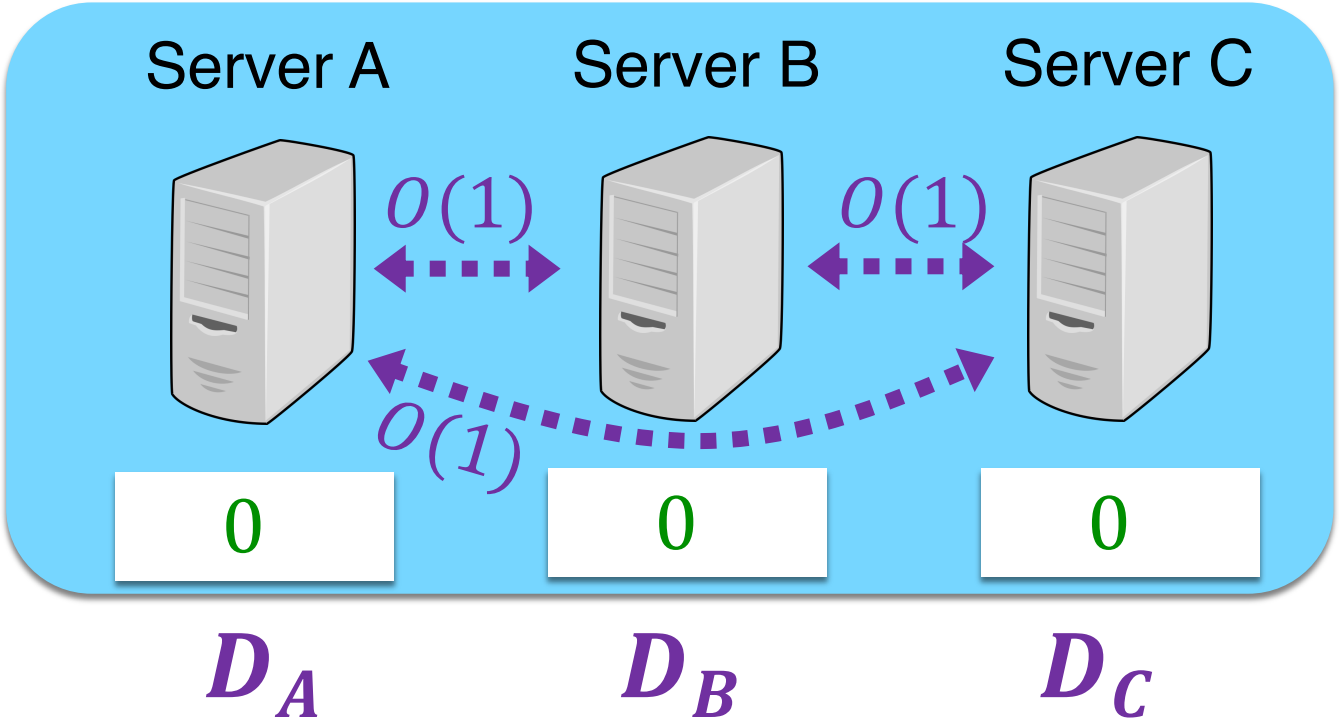


If x is well formed: $D_A + D_B + D_C = 0$

If x is malformed: $D_A + D_B + D_C \neq 0$ with high probability

Servers publish D_A, D_B, D_C and check that they sum to 0.
→ Servers accept x if so.

How to construct a proof on secret-shared data

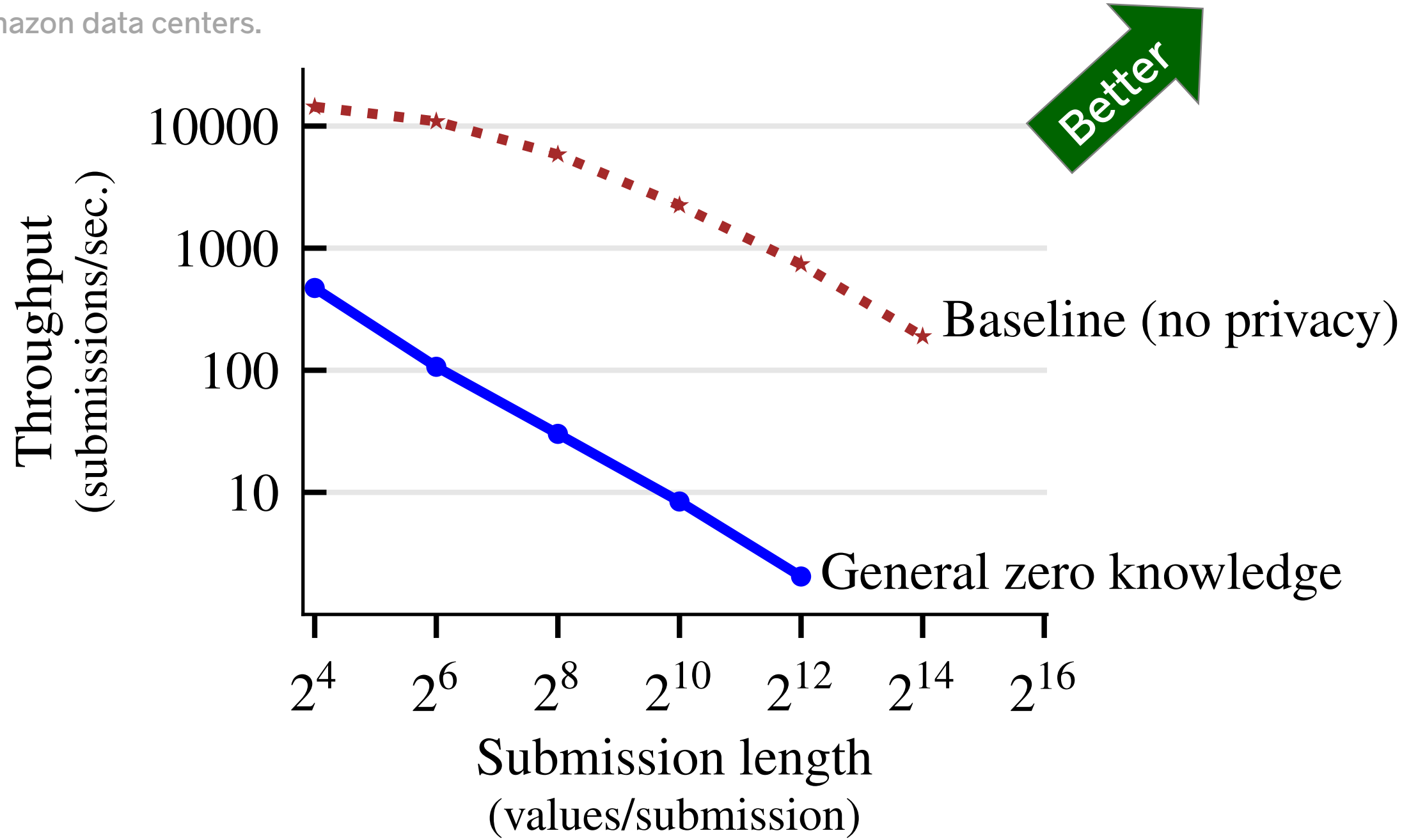


If x is well formed: $D_A + D_B + D_C = 0$

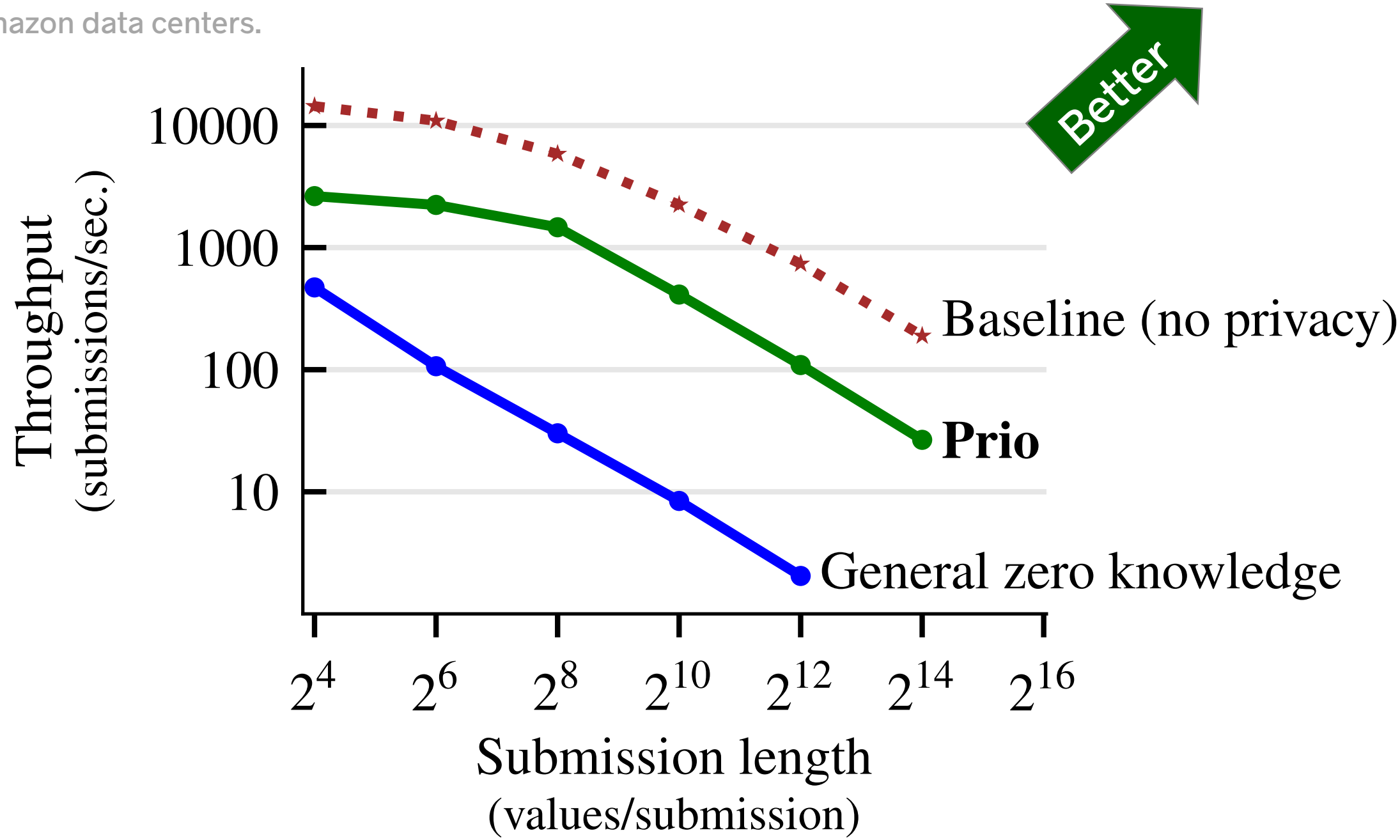
If x is malformed: $D_A + D_B + D_C \neq 0$ with high probability

Servers publish D_A, D_B, D_C and check that they sum to 0.
→ Servers accept x if so.

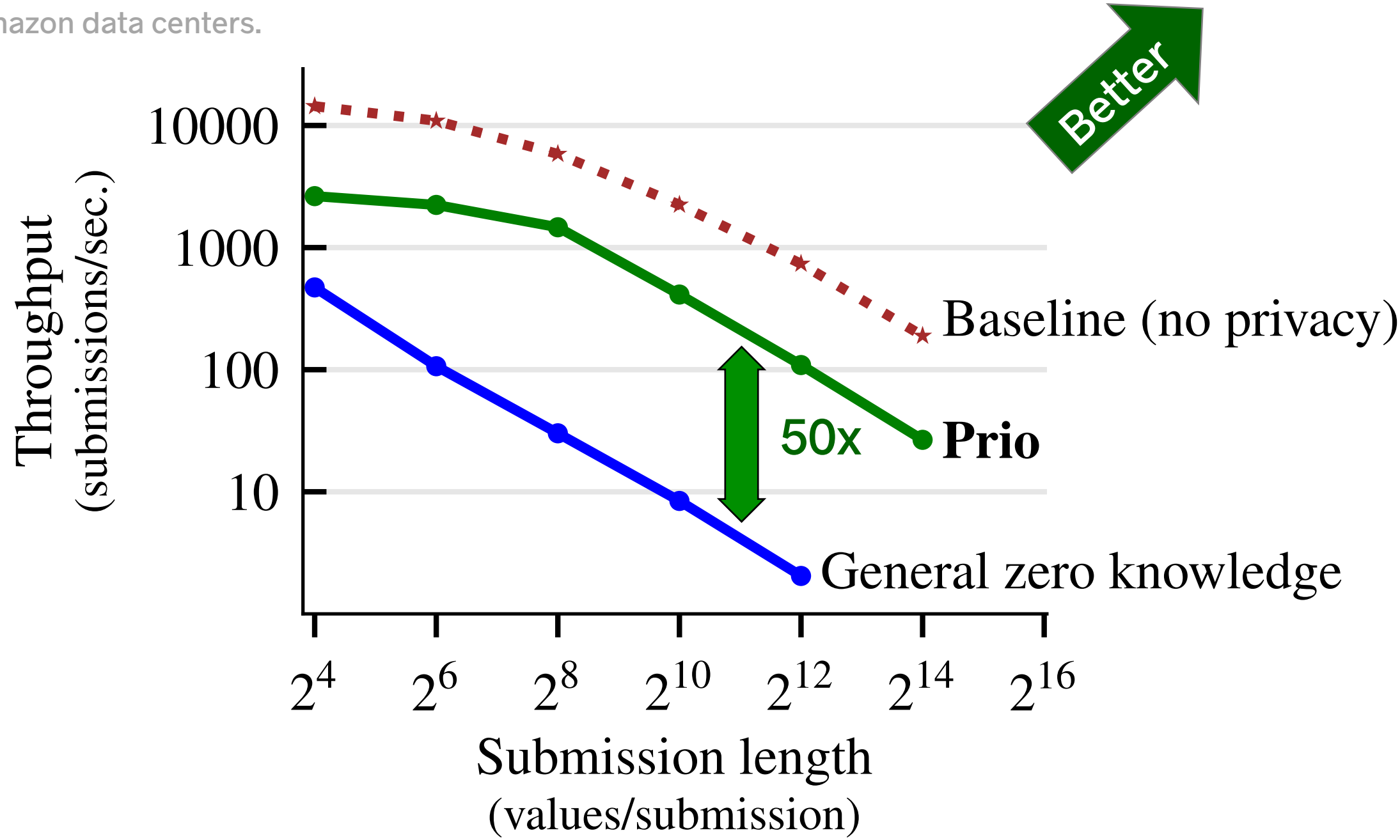
Five-server cluster in five Amazon data centers.



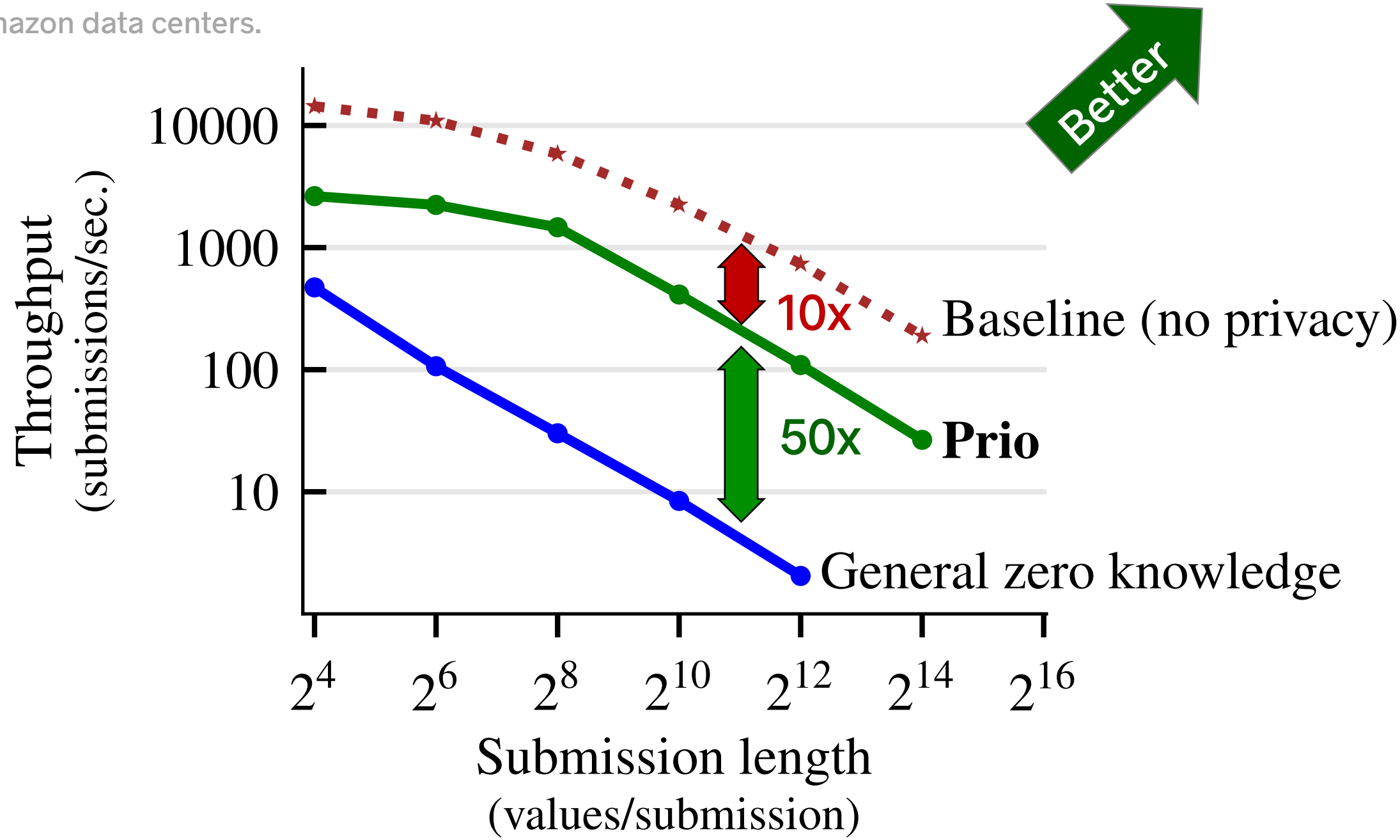
Five-server cluster in five Amazon data centers.



Five-server cluster in five Amazon data centers.



Five-server cluster in five Amazon data centers.



Prio supports a range of aggregation functions

- **Average**
- **Variance** [PBBL11]
- **Most popular** (approx.) [MDD16]
- **Min** and **max** (approx.)
- **Quality of arbitrary regression model** (R^2)
- **Least-squares regression**
- **Gradient descent step** [BIKMMPRSS17]

Firefox Deployment

Firefox deployment

Uses libprio, a C library we wrote that implements Prio

- `github.com/mozilla/libprio` - 3.5k LoC
- Encoding a length-1024 data packet: 35ms in Firefox browser (more optimizations possible)
- Python bindings to simplify server-side data analysis

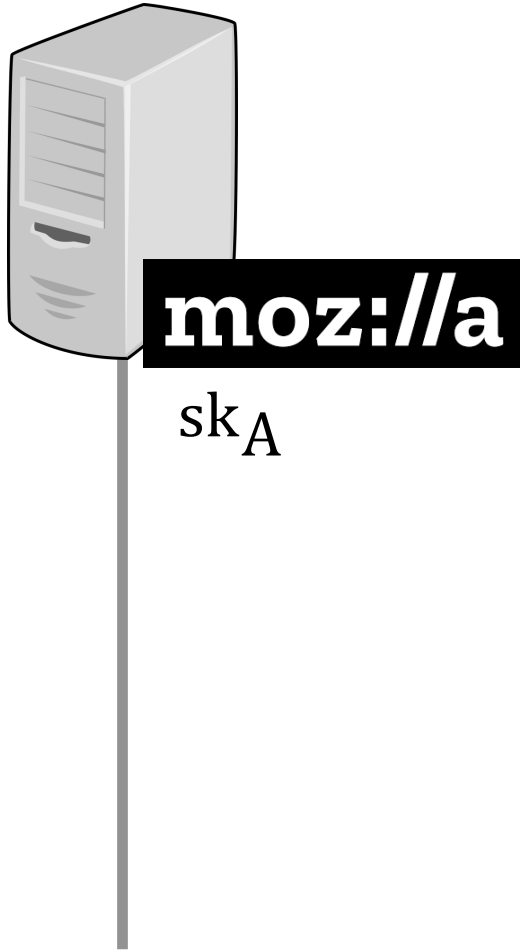
Pilot phase, 11/2018-now

- Implemented in Firefox, but Mozilla currently runs all servers
- Enabled by default only in the “Nightly” build

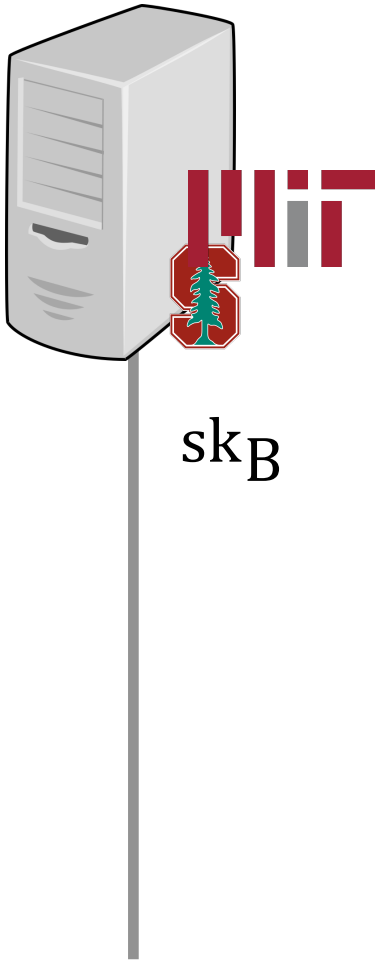
Next step: **Move second server to external org.** (In progress)

Firefox deployment

pk_A, pk_B

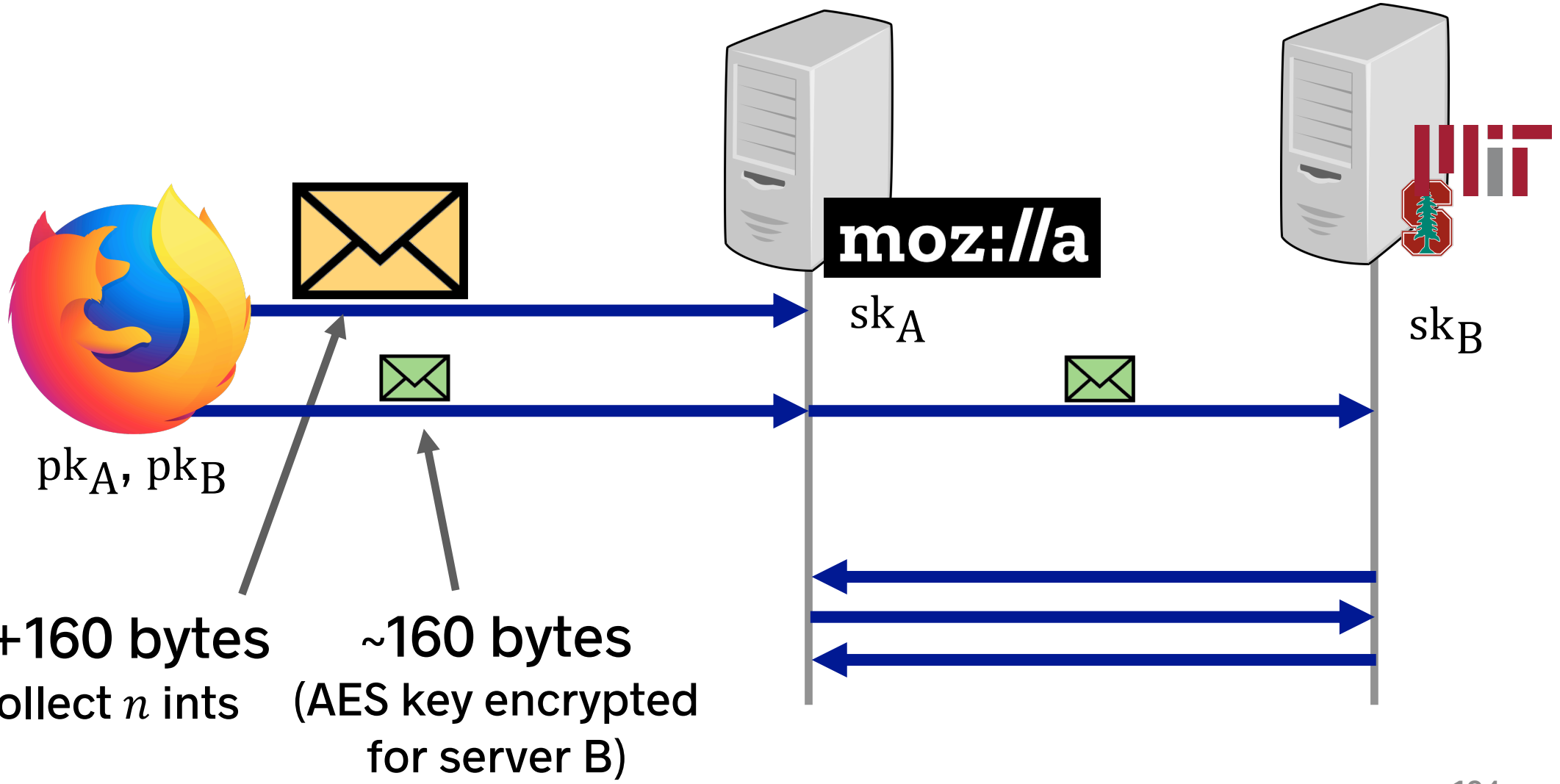


sk_A

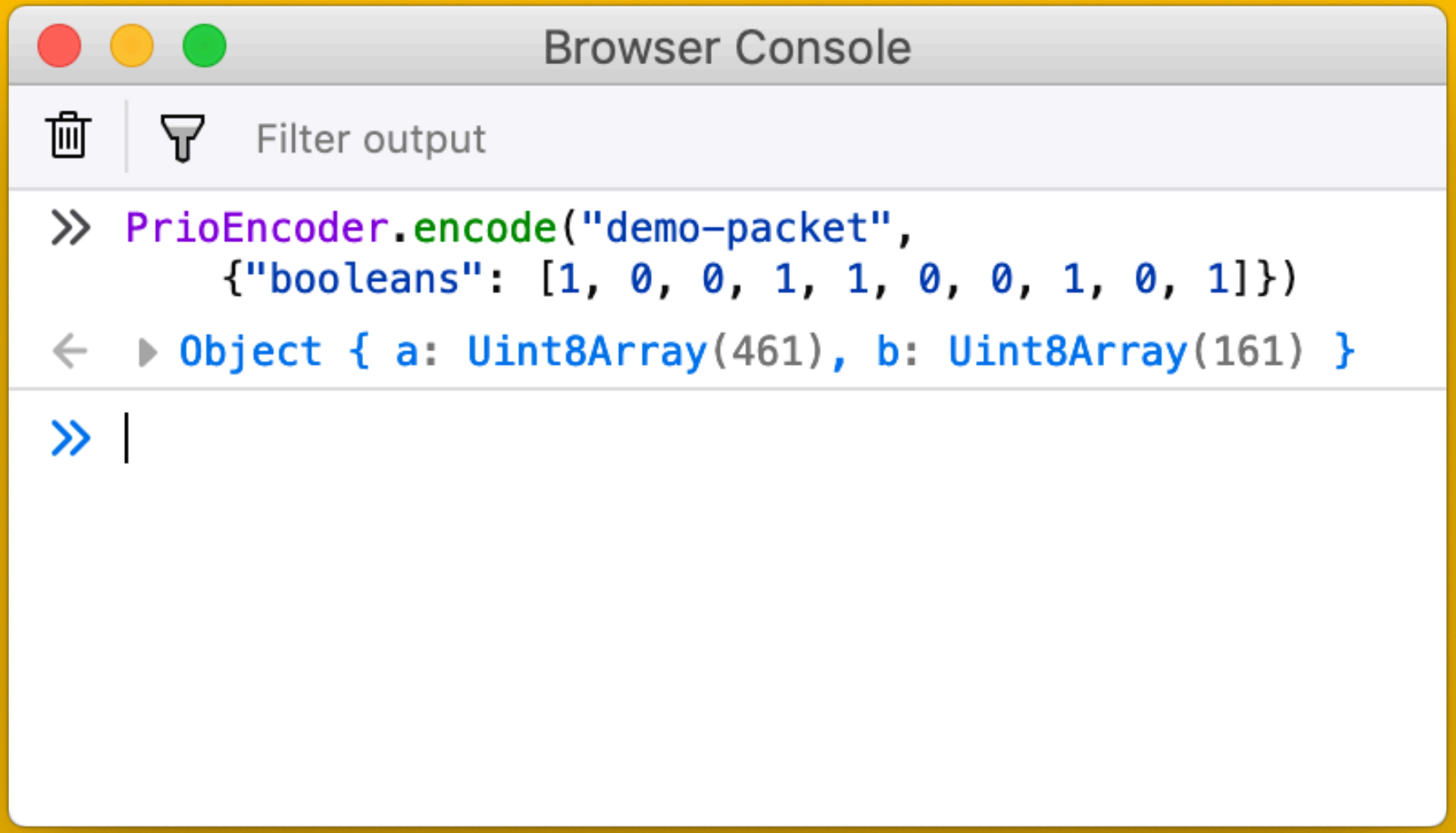


sk_B

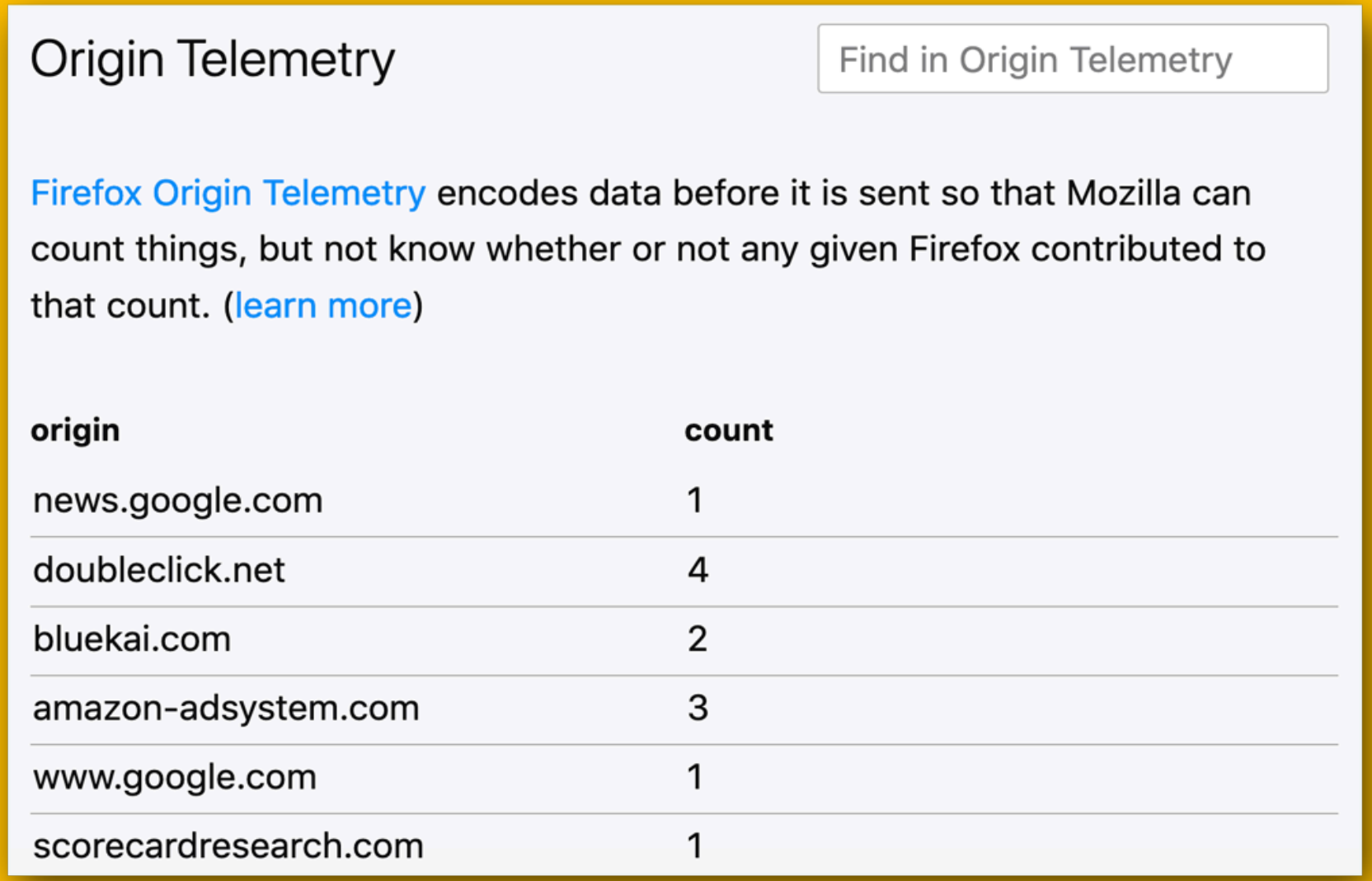
Firefox deployment

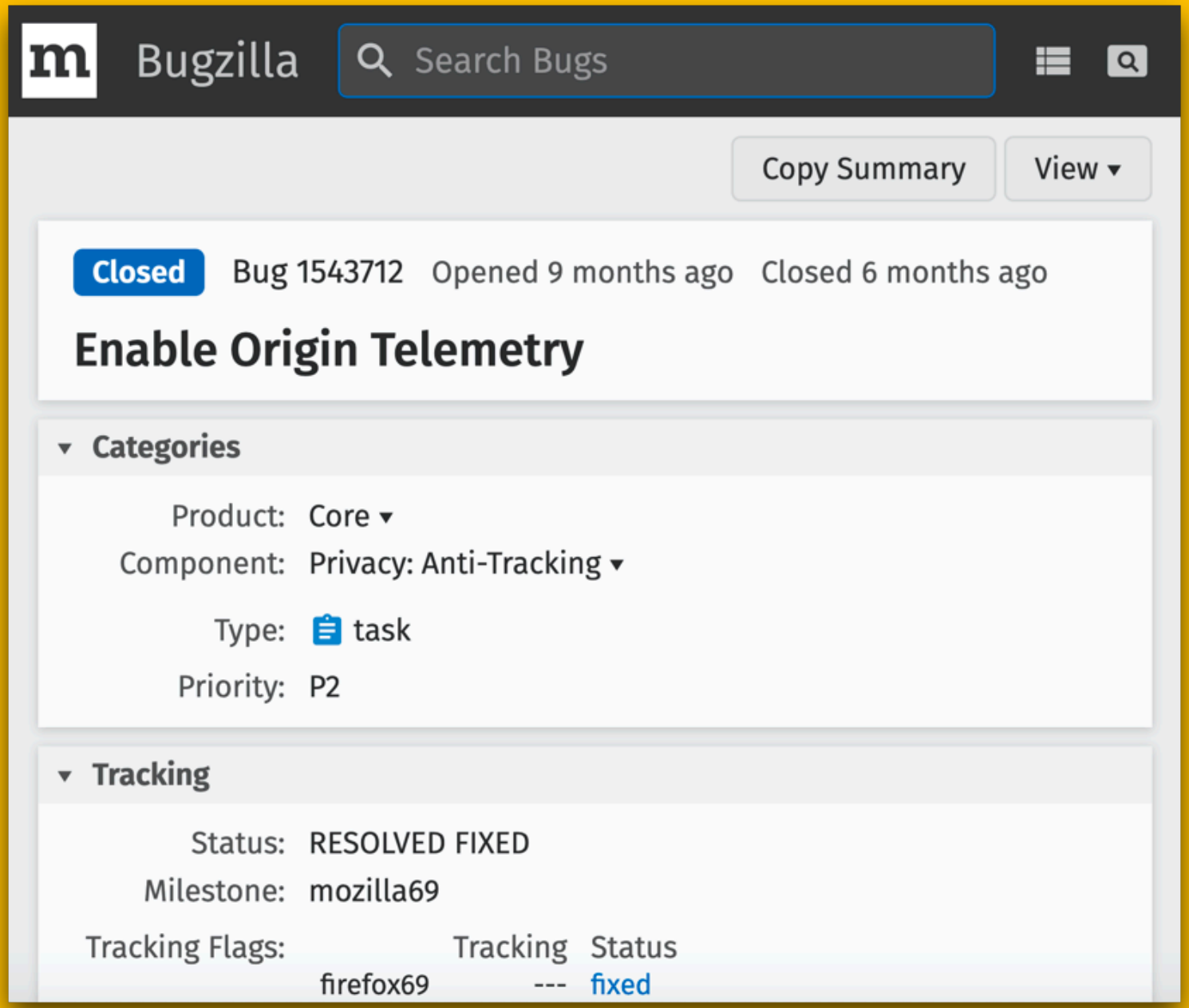


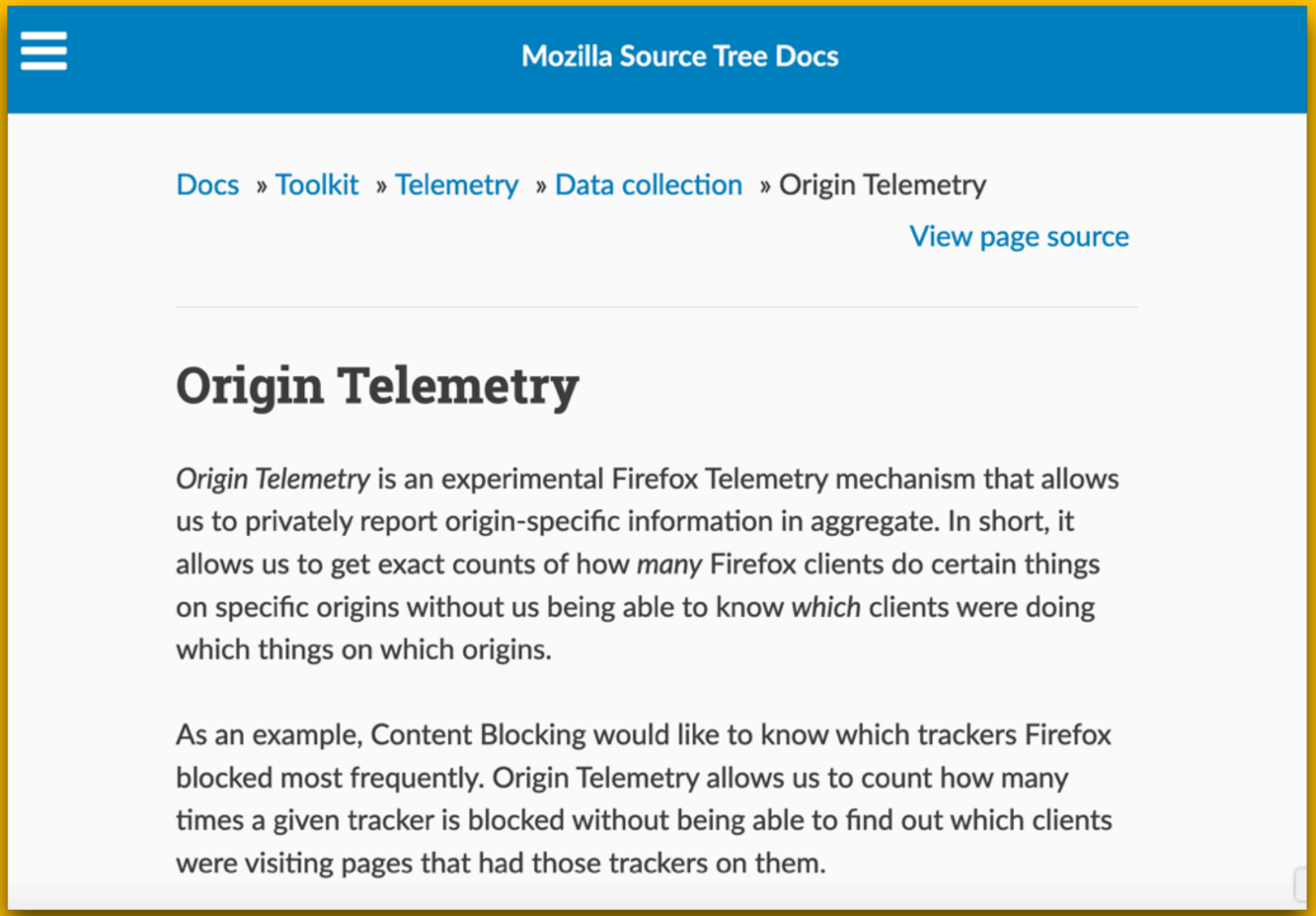
In Firefox, set preference `devtools.chrome.enabled`, then in browser console...



In Nightly, set pref. `telemetry.origin_telemetry_test_mode.enabled`, browse for a while, then visit `about:telemetry`.




Bugzilla
Search Bugs
Copy Summary
View
Closed
Bug 1543712
Opened 9 months ago
Closed 6 months ago
Enable Origin Telemetry
Categories
Product:
Core
Component:
Privacy: Anti-Tracking
Type:
task
Priority:
P2
Tracking
Status:
RESOLVED FIXED
Milestone:
mozilla69
Tracking Flags:
Tracking
Status
firefox69
---
fixed



https://firefox-source-docs.mozilla.org/toolkit/components/telemetry/collection/origin.html

Testing Privacy-Preserving Telemetry with Prio

By Robert Helmer, Anthony Miyaguchi, Eric Rescorla

Posted on October 29, 2018 in Firefox and Privacy

Building a browser is hard; building a good browser inevitably requires gathering a lot of data to make sure that things that work in the lab work in the field. But as soon as you gather data, you have to make sure you protect user privacy. We're always looking at ways to improve the security of our data collection, and lately we've been experimenting with a really cool technique called Prio.

CHUTTENBLOG

Posted on 2019-04-26 in mozilla

Firefox Origin Telemetry: Putting Prio in Practice

Prio is neat. It allows us to learn counts of things that happen across the Firefox population without ever being able to learn which Firefox sent us which pieces of information.

For example, Content Blocking will soon be using this to count how

Deployment stats

- Initially, collecting data on ~2,500 blocklist rules
  fb.com, google-analytics.com, adwords.google.com, ...
- Data collected on 0.014% of pageloads for 1% of clients
- Expect to process ~200m telemetry submissions per day
  - Submission from client every 24h or on shutdown
    = Tens of gigabytes of data per day to the second server

The second server

- Prio requires 2+ non-colluding servers, maintained ideally
  - by independent organizations,
  - on independent infrastructure (not both on AWS), and
  - in different countries (under independent legal jurisdictions).
- Serious non-technical challenge, but reasons for optimism
  - Infrastructure costs are modest
  - ∃ multiple candidate orgs with privacy-centric mission
  - If Org2 uses Prio, Mozilla can be the “second server” for Org2

→ Mozilla is working to sign up a partner org in 2020.

You can help!

github.com/mozilla/libprio/

Small things

- Add support for aggregating a wider range of data types
- Implement client- and server-side optimizations
- Implement differential-privacy features

Big things

- Rewrite parts of libprio in Rust
- Be the external org that runs the second server

→ Eligible for Mozilla's bug-bounty program. ←

Conclusion

- Prio is a new system for privacy-preserving telemetry
- Firefox is using Prio to collect data to improve the browser's new tracking-protection feature
- Deployment is ongoing!
  - Ask if you're interested in helping out.

Henry Corrigan-Gibbs (EPFL & MIT CSAIL), henrycg@csail.mit.edu
Dan Boneh (Stanford), Gary Chen, Steven Englehardt, Robert Helmer,
Chris Hutten-Czapski, Anthony Miyaguchi, Eric Rescorla, and Peter Saint-Andre (Mozilla)

Details: bugzilla.mozilla.org/show_bug.cgi?id=1543712
Code: github.com/mozilla/libprio/
Paper: crypto.stanford.edu/prio/

Conclusion

- Prio is a new system for privacy-preserving telemetry
- Firefox is using Prio to collect data to improve the browser's new tracking-protection feature
- Deployment is ongoing!
  - Ask if you're interested in helping out.

Henry Corrigan-Gibbs (EPFL & MIT CSAIL), henrycg@csail.mit.edu
Dan Boneh (Stanford), Gary Chen, Steven Englehardt, Robert Helmer,
Chris Hutten-Czapski, Anthony Miyaguchi, Eric Rescorla, and Peter Saint-Andre (Mozilla)

Details: bugzilla.mozilla.org/show_bug.cgi?id=1543712
Code: github.com/mozilla/libprio/
Paper: crypto.stanford.edu/prio/

Thank you!