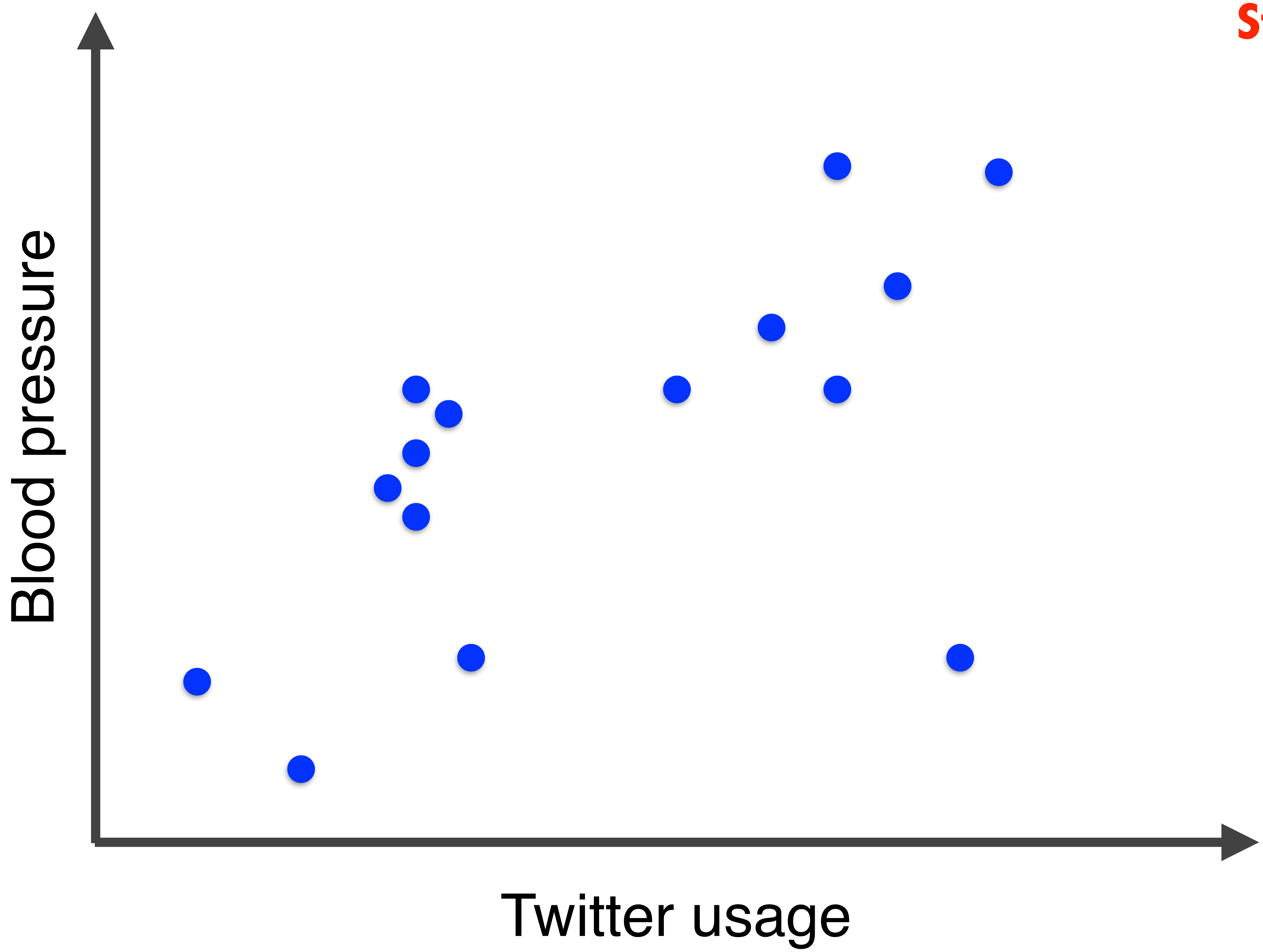
Prio: Private, Robust, and Efficient Computation of Aggregate Statistics

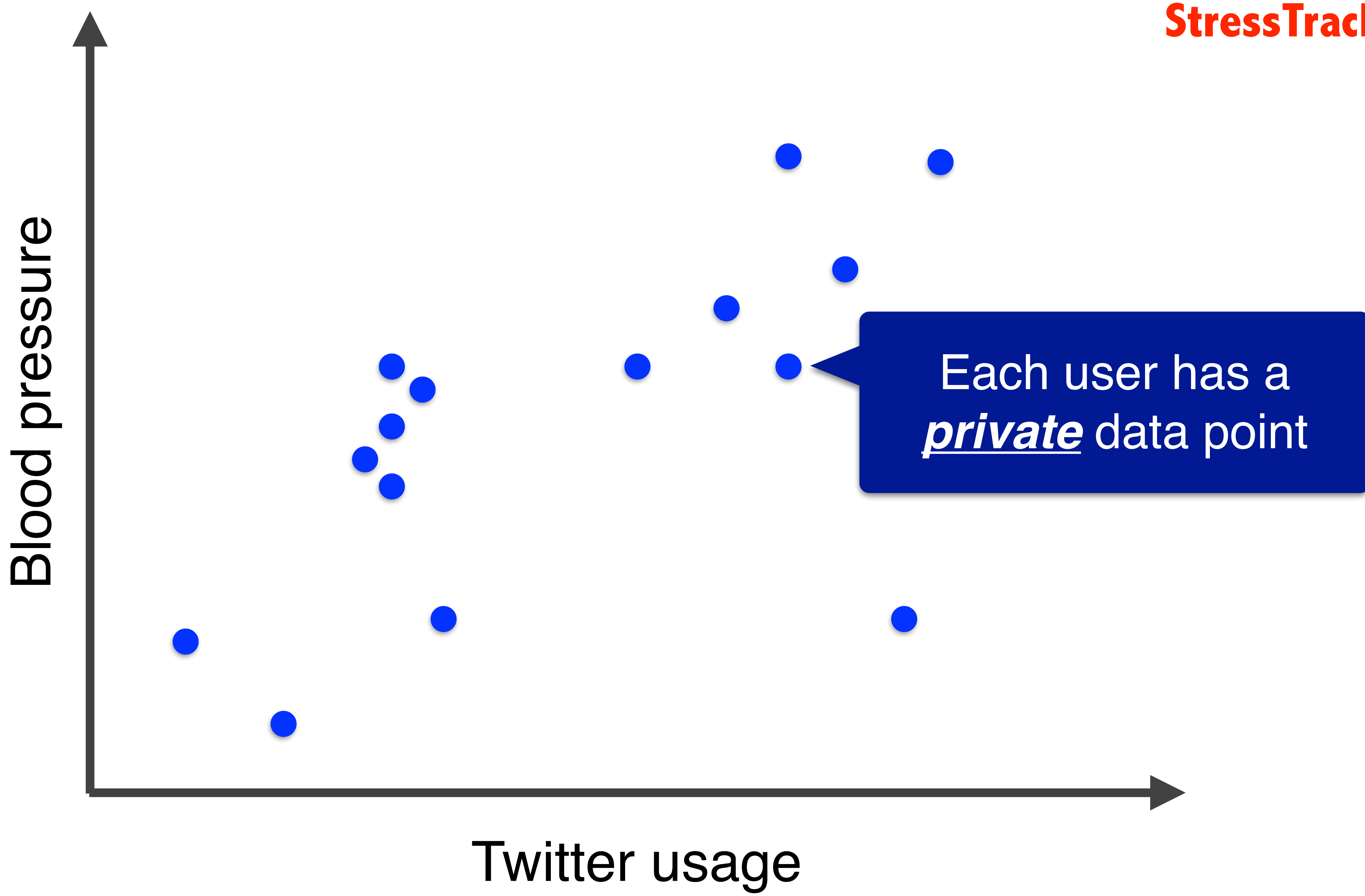
Henry Corrigan-Gibbs and Dan Boneh
Stanford University

Appeared at NSDI 2017

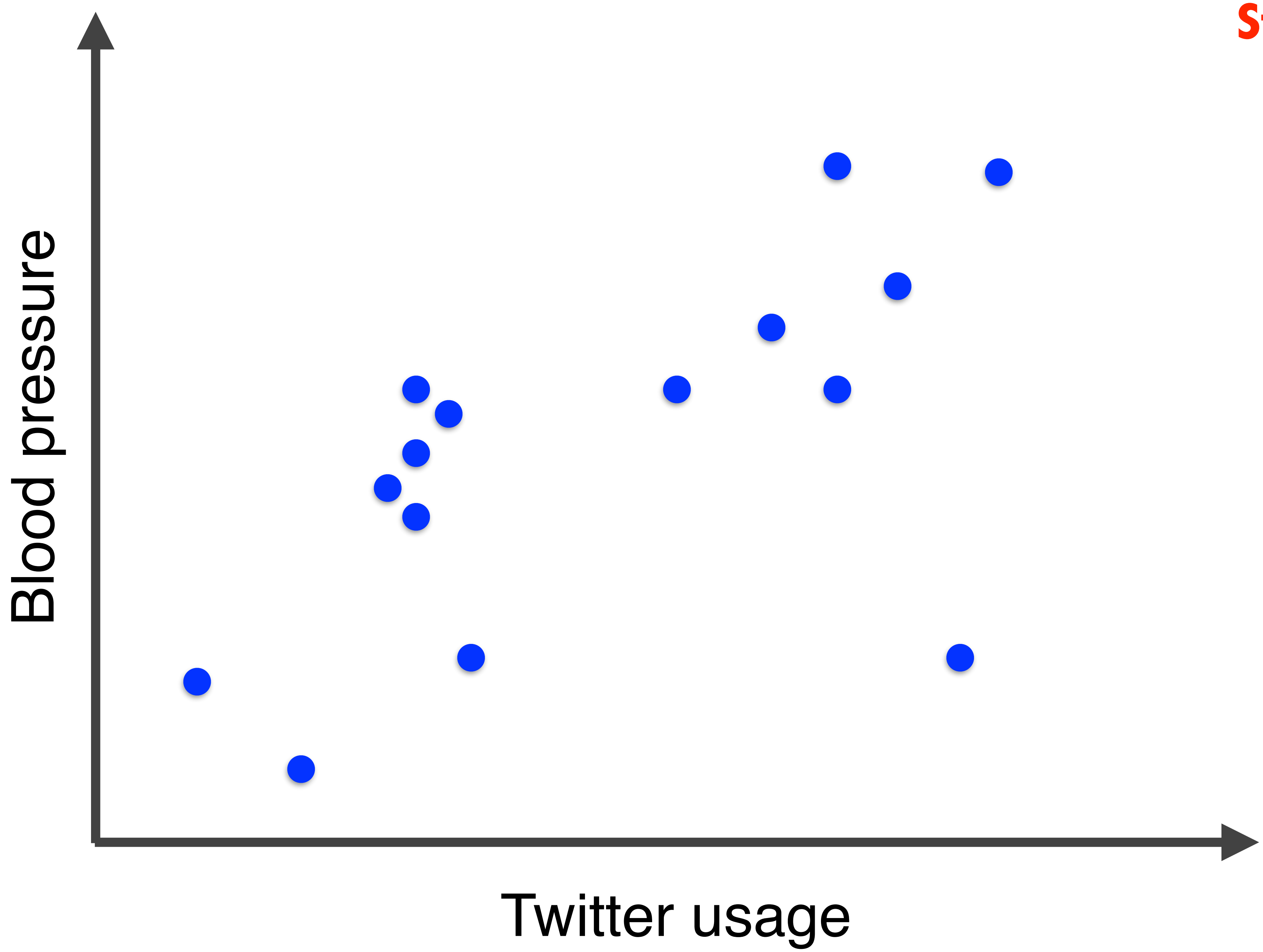
Today: Non-private aggregation



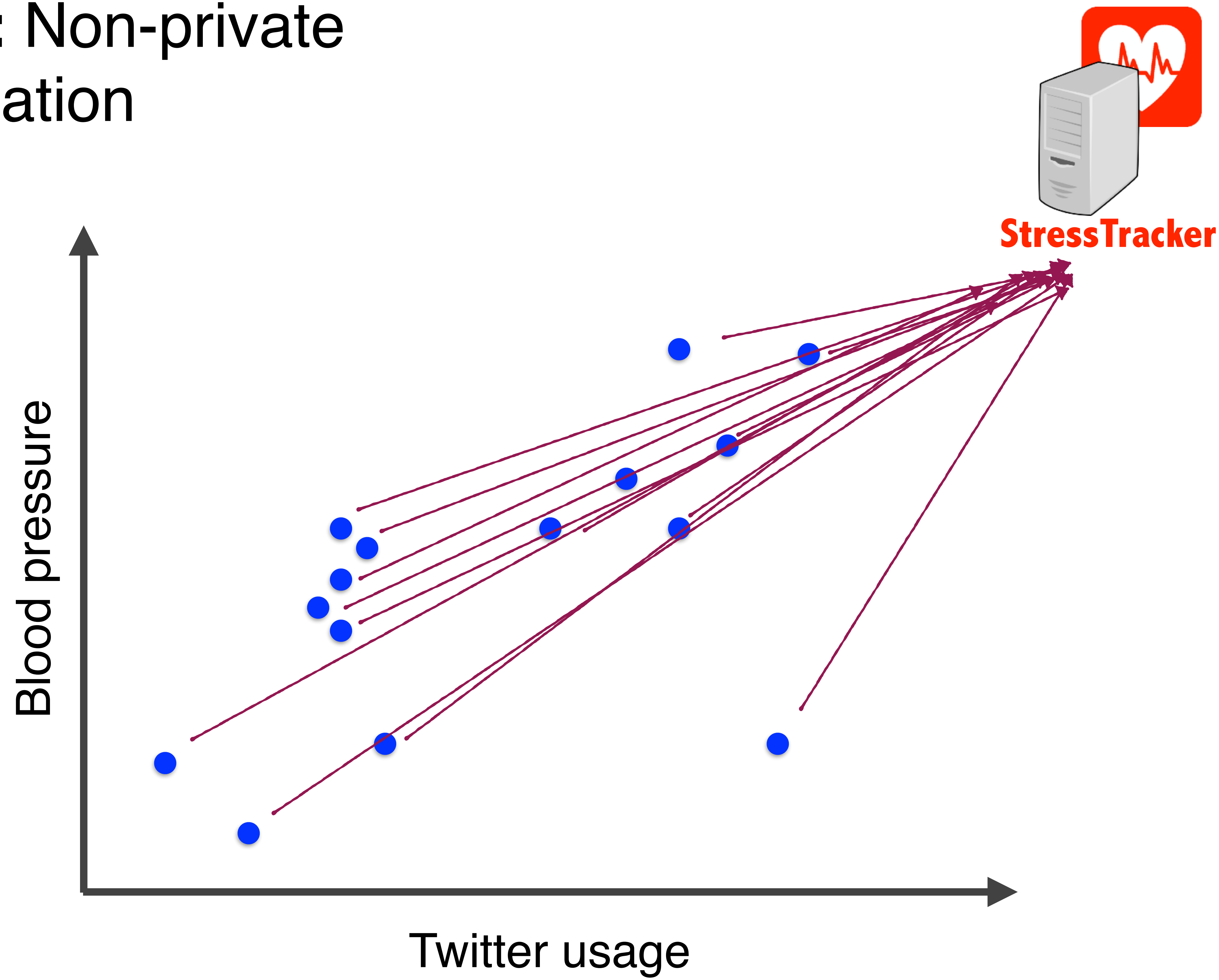
Today: Non-private aggregation



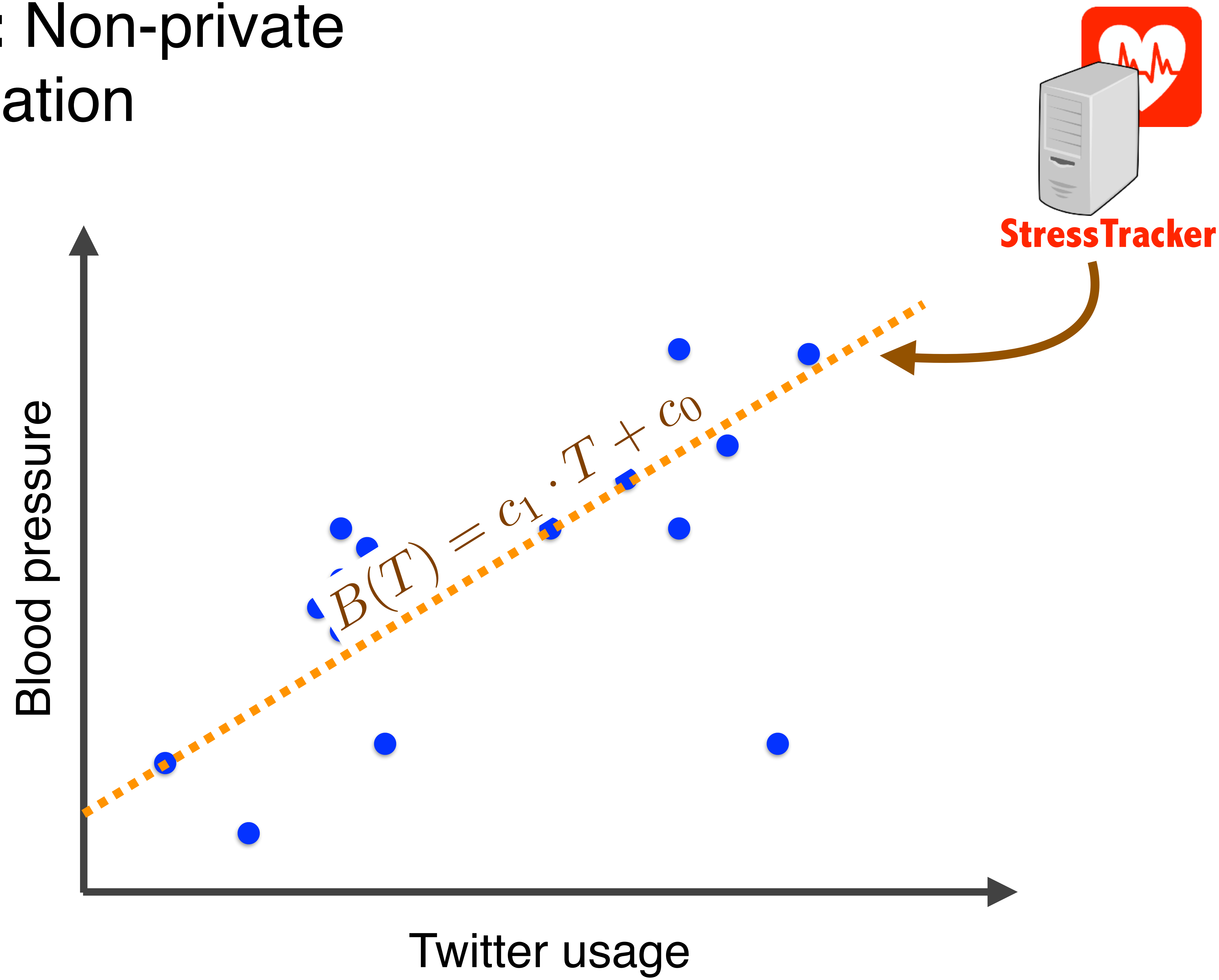
Today: Non-private aggregation



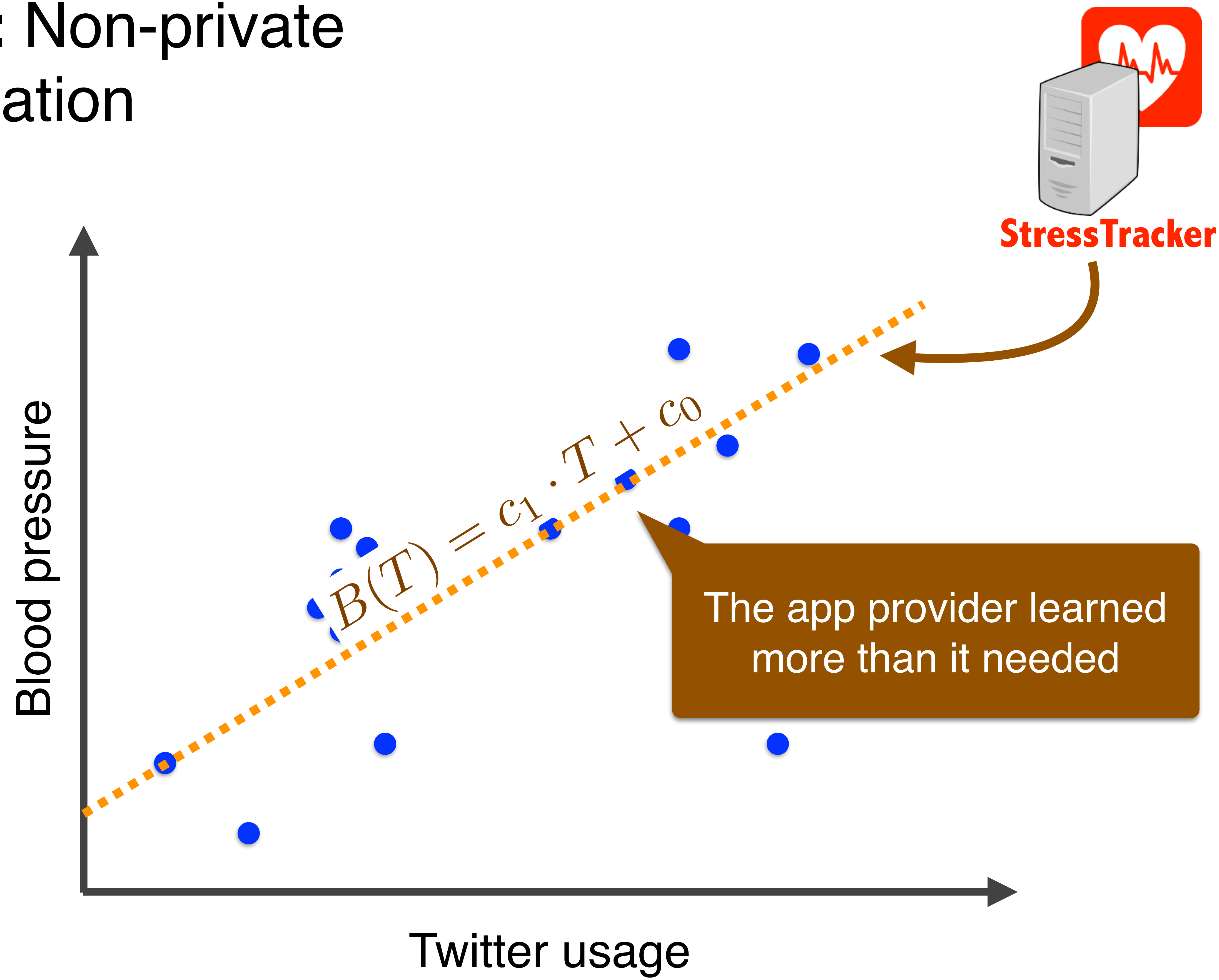
Today: Non-private aggregation



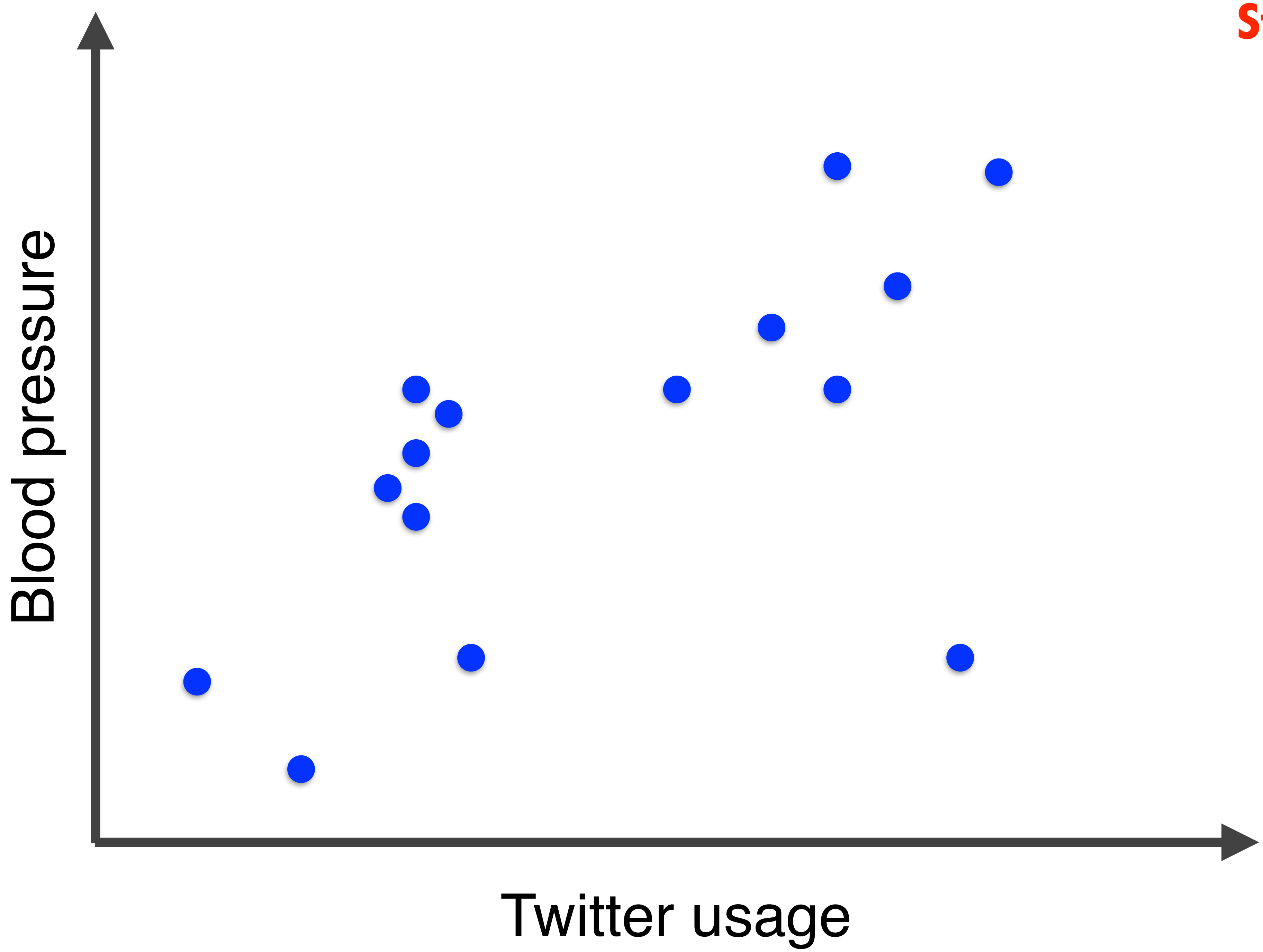
Today: Non-private aggregation



Today: Non-private aggregation

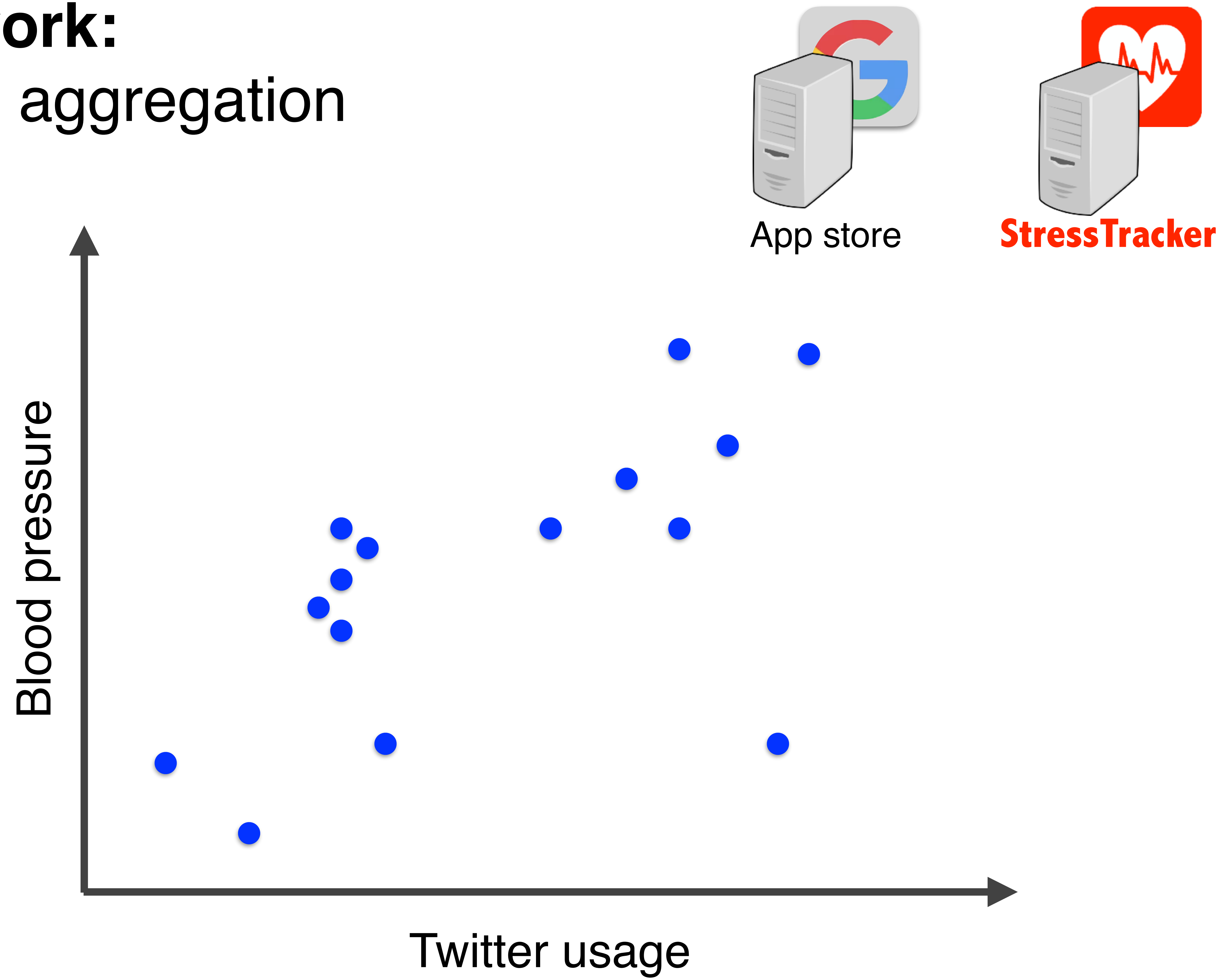


Today: Non-private aggregation

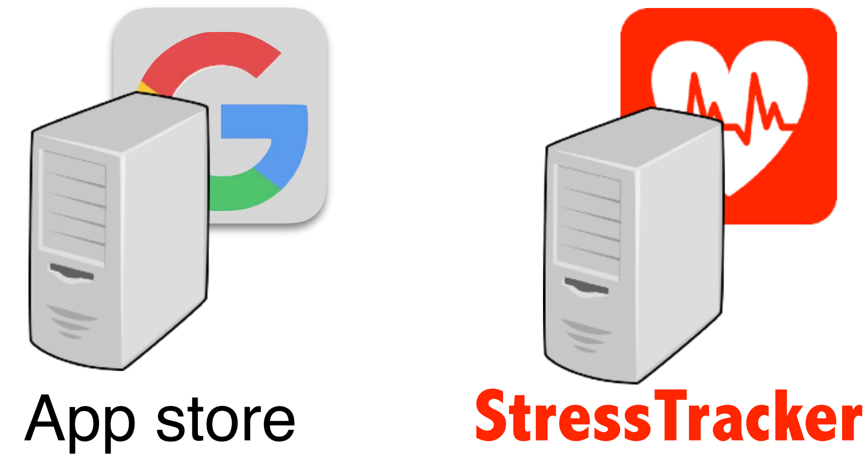


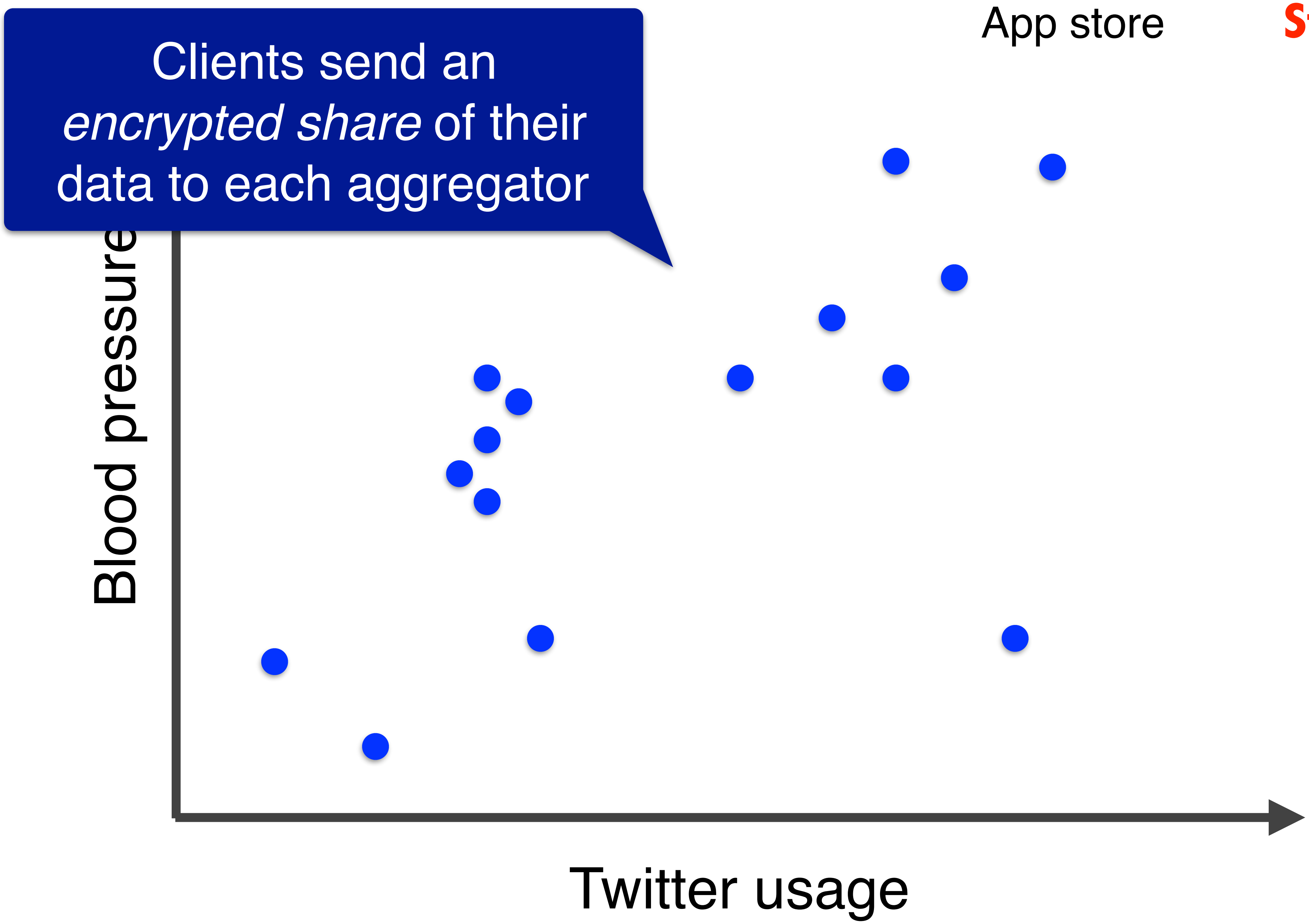
This work:

Private aggregation



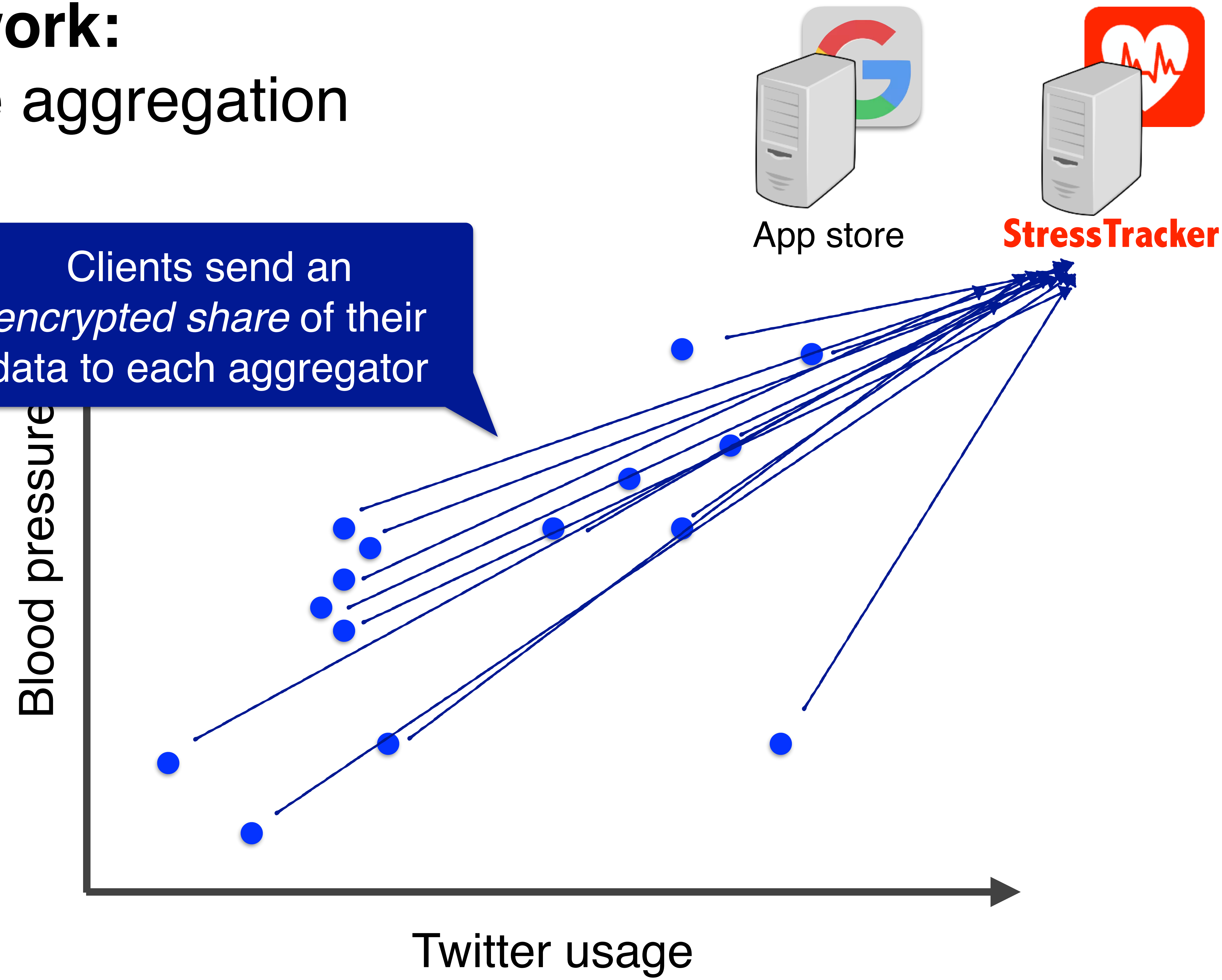
This work:
Private aggregation



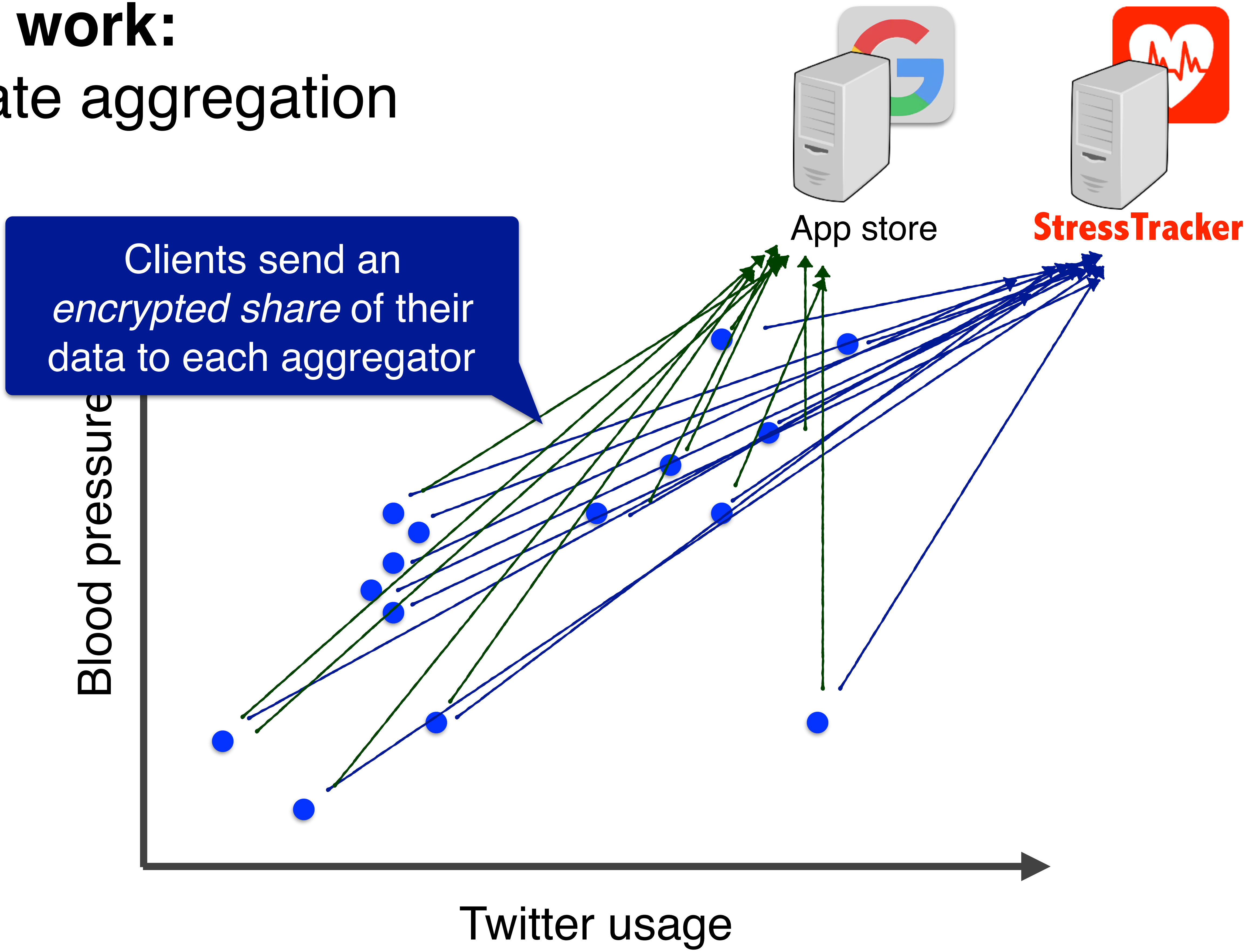


This work:
Private aggregation

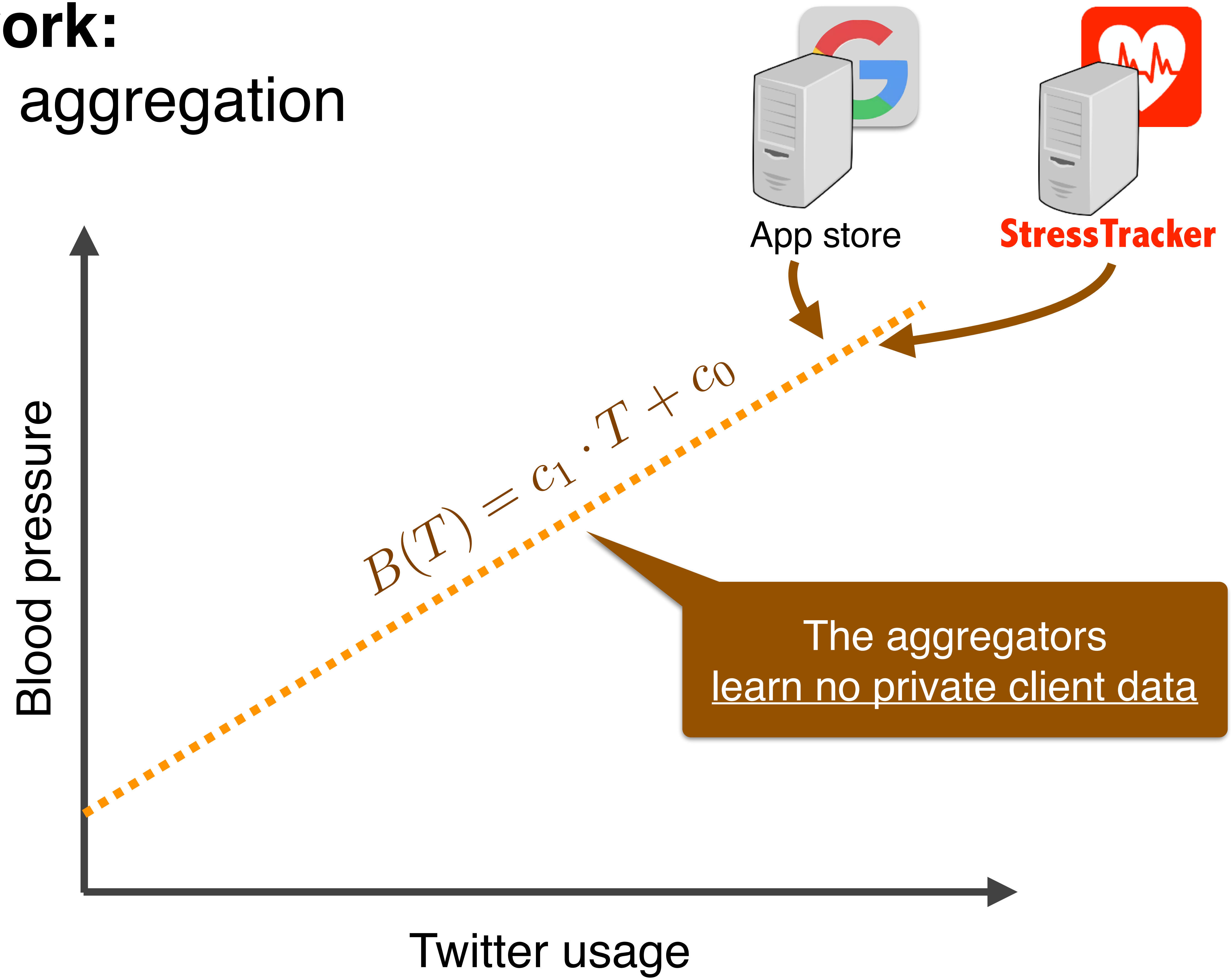
Clients send an *encrypted share* of their data to each aggregator



This work:
Private aggregation

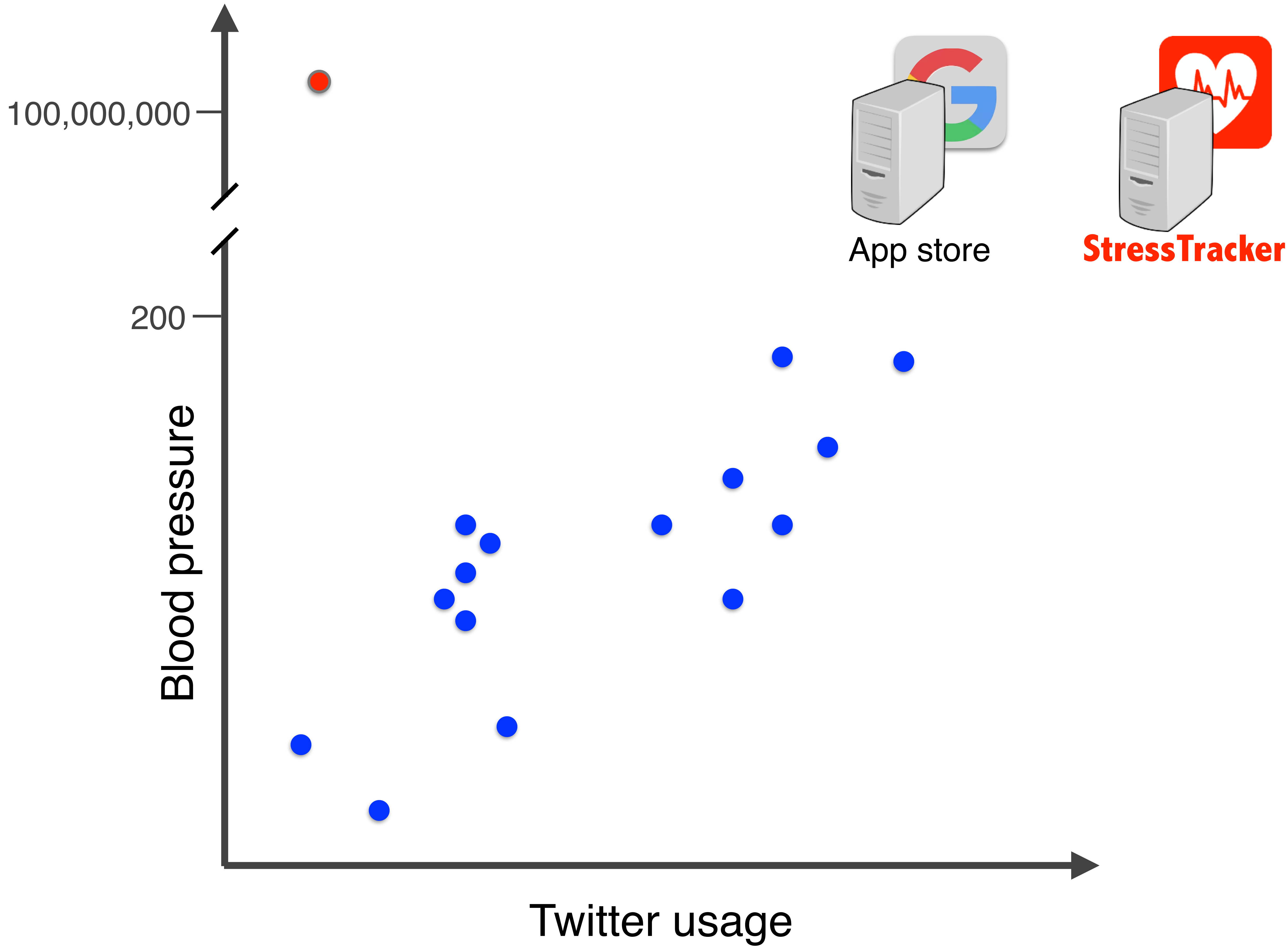


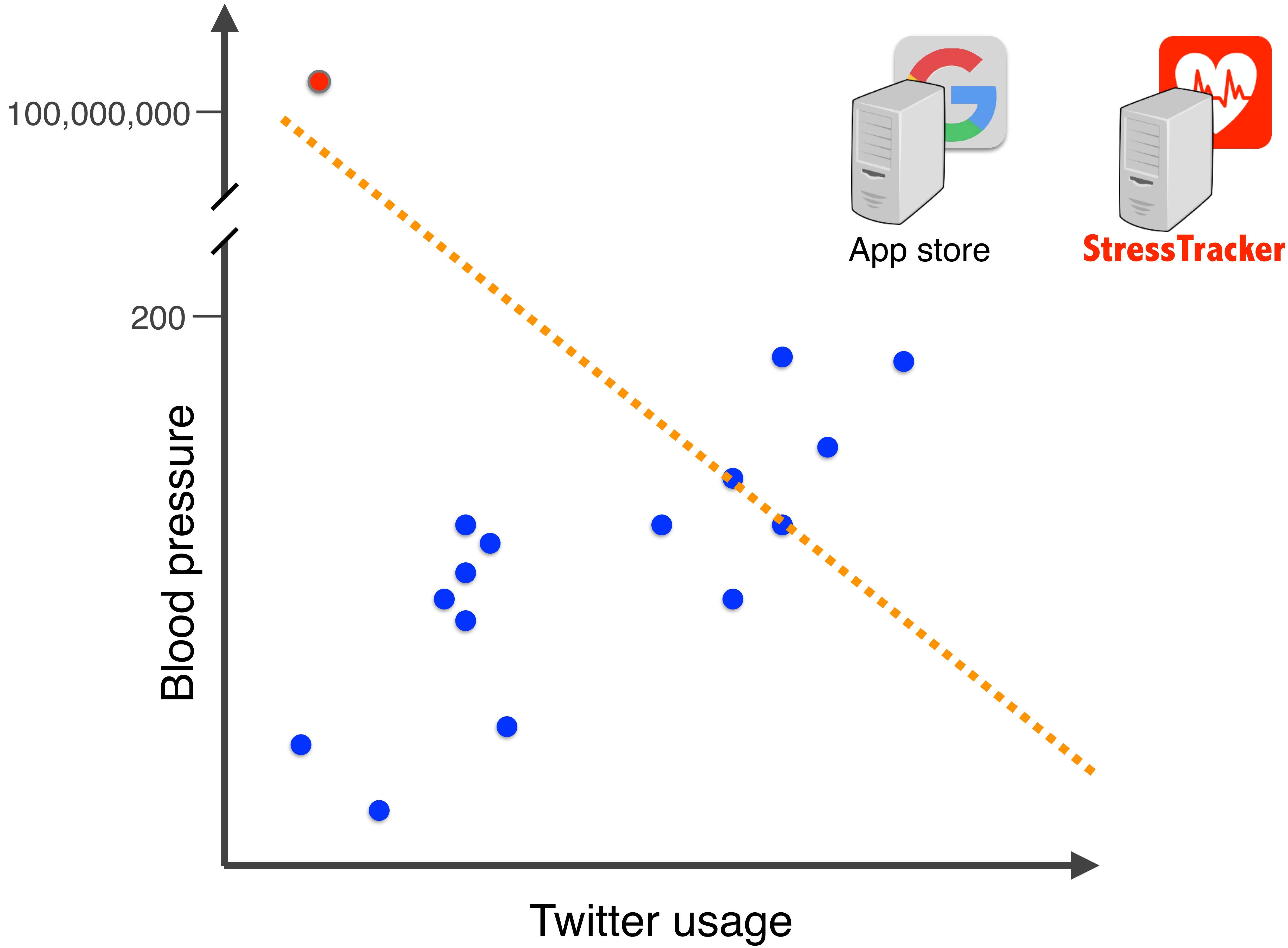
This work:
Private aggregation



Private aggregation

1. **Exact correctness** — If all servers are honest, servers learn $f(\cdot)$
2. **Privacy** — If one server is honest, servers learn only* $f(\cdot)$
3. **Robustness** — Malicious clients have bounded influence
4. **Efficiency** — No public-key crypto (apart from TLS)
   1000s of submissions per second


100,000,000
200
Blood pressure
Twitter usage
App store
StressTracker



100,000,000
200
Blood pressure
Twitter usage
App store
StressTracker

Private aggregation

1. **Exact correctness** If all servers are honest, servers learn $f(\cdot)$
2. **Privacy** If one server is honest, servers learn only* $f(\cdot)$
3. **Robustness** Malicious clients have bounded influence
4. **Efficiency** No public-key crypto (apart from TLS)

Prio is the first system to achieve all four.

Private aggregation

1. **Exact correctness** If <u>all servers</u> are honest, servers learn $f(\cdot)$
2. **Privacy** If <u>one server</u> is honest, servers learn only* $f(\cdot)$
3. **Robustness** Malicious clients hav
4. **Efficiency** No public-key crypto

…and Prio supports a wide range of aggregation functions $f(\cdot)$

Prio is the first system to achieve all four.

Private aggregation

1. **Exact correctness** — If all servers are honest, servers learn $f(\cdot)$
2. **Privacy** — If one server is honest, servers learn only* $f(\cdot)$
3. **Robustness** — Malicious clients have bounded influence
4. **Efficiency** — No public-key crypto (apart from TLS)

Prio is the first system to achieve all four.

Contributions

1. **Secret-shared non-interactive proofs (SNIPs)**
   – Client proves that its encoded submission is well-formed
   – We do not need the power of traditional “heavy” crypto tools

2. **Aggregatable encodings**
   Can compute sums privately $\implies$ Can compute $f(\cdot)$ privately
   …for many f’s of interest

Contributions

1. **Secret-shared non-interactive proofs (SNIPs)**
   – Client proves that its encoded submission is well-formed
   – We do not need the power of traditional “heavy” crypto tools

2. **Aggregatable encodings**
   Can compute sums privately $\Longrightarrow$ Can compute $f(\cdot)$ privately
   …for many f’s of interest

See the paper

Related systems

- Additively homomorphic encryption
  P4P (2010), Private stream aggregation (2011), Grid aggregation (2011), PDDP (2012), SplitX (2013), PrivEx (2014), PrivCount (2016), Succinct sketches (2016), …
- Multi-party computation [GMW87], [BGW88]
  FairPlay (2004), Brickell-Shmatikov (2006), FairplayMP (2008), SEPIA (2010), Private matrix factorization (2013), JustGarble (2013), …
- Anonymous credentials/tokens
  VPriv (2009), PrivStats (2011), ANONIZE (2014), …
- Randomized response [W65], [DMNS06], [D06]
  RAPPOR (2014, 2016), …

Prio is the first system to achieve
exact correctness, privacy, robustness, efficiency.

Outline

- **Background: The private aggregation problem**
- A straw-man solution for private sums
- Providing robustness with SNIPs
- Evaluation
- Discussion: Real-world considerations

Outline

- Background: The private aggregation problem
- **A straw-man solution for private sums**
- Providing robustness with SNIPs
- Evaluation
- Discussion: Real-world considerations

Warm-up: Computing private sums

Warm-up: Computing private sums

- Every device i holds a value x_i
- We want to compute

$$f(x_1, \ldots, x_N) = x_1 + \ldots + x_N$$

without learning any users' private value x_i.

Warm-up: Computing private sums

- Every device i holds a value x_i
- We want to compute
$$f(x_1, \ldots, x_N) = x_1 + \ldots + x_N$$
without learning any users' private value x_i.

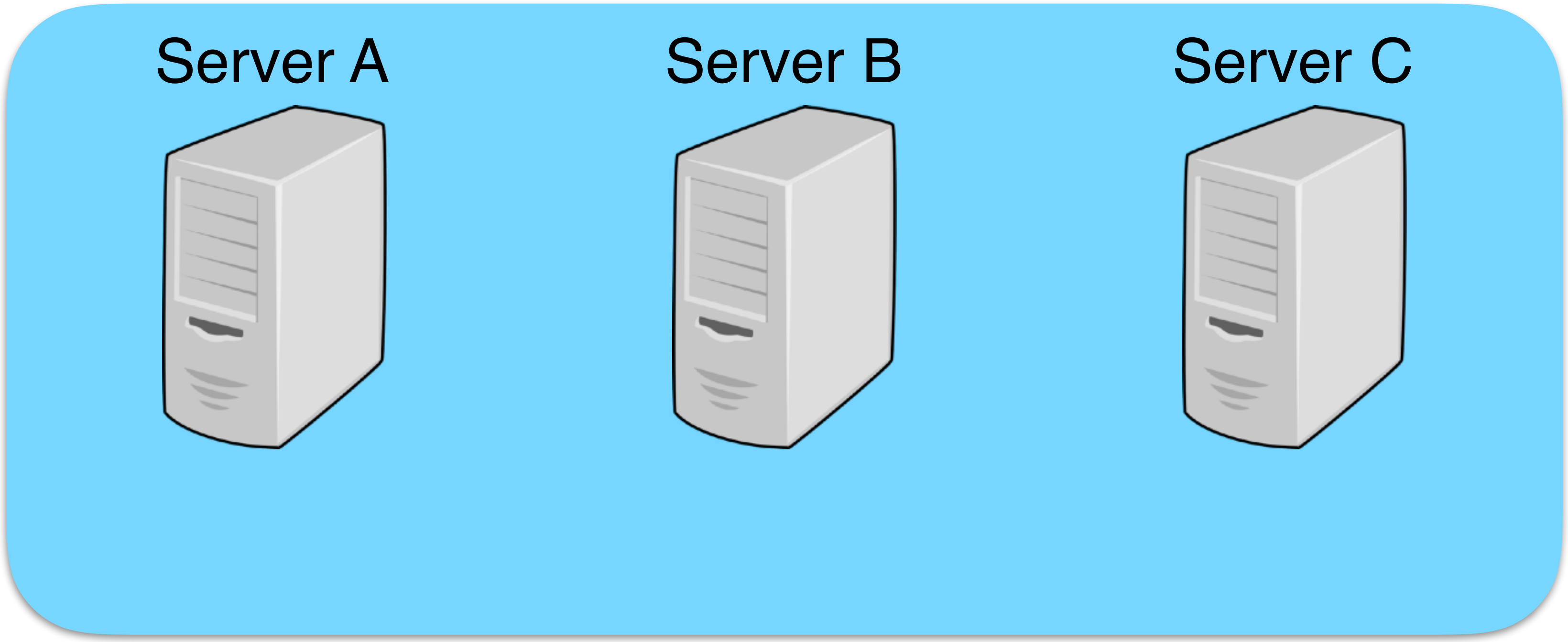
Example: Privately measuring traffic congestion.

$$x_i \quad = 1 \qquad \text{if user } i \text{ is on the Bay Bridge}$$
$$\quad\;\; = 0 \qquad \text{otherwise}$$

The sum $x_1 + \ldots + x_N$ yields the number of app users on the Bay Bridge.

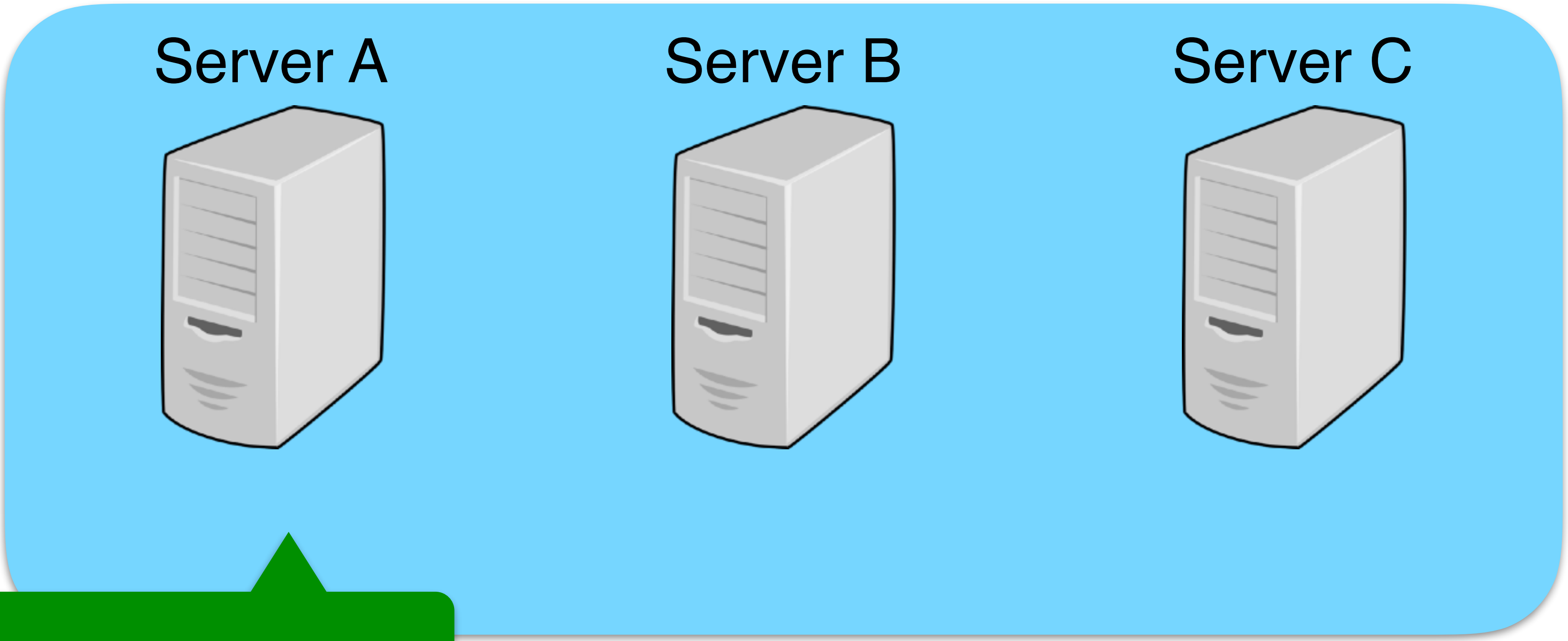
Private sums: A “straw-man” scheme

[Chaum88], [BGW88], …
[KDK11] [DFKZ13] [PrivEx14] …



Private sums: A "straw-man" scheme

[Chaum88], [BGW88], …
[KDK11] [DFKZ13] [PrivEx14] …

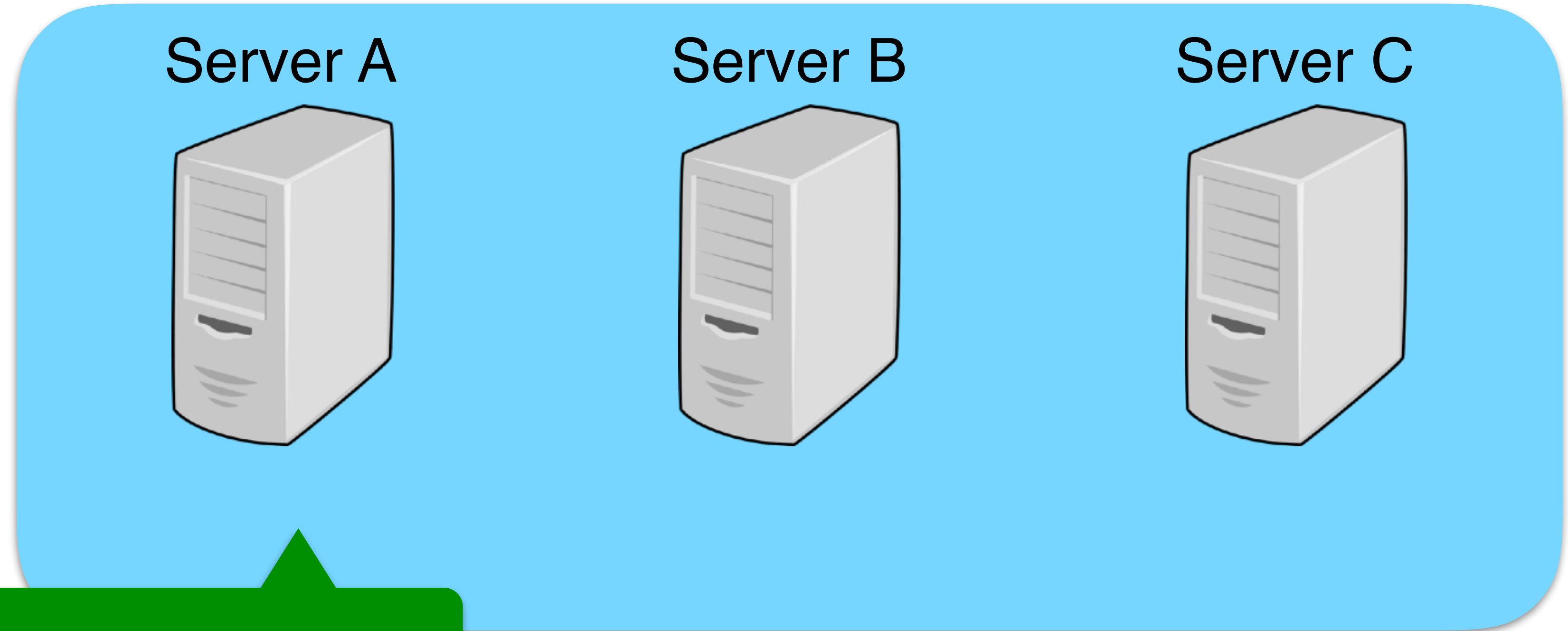


Assume that the servers are non-colluding.

Equivalently: that at least one server is honest.

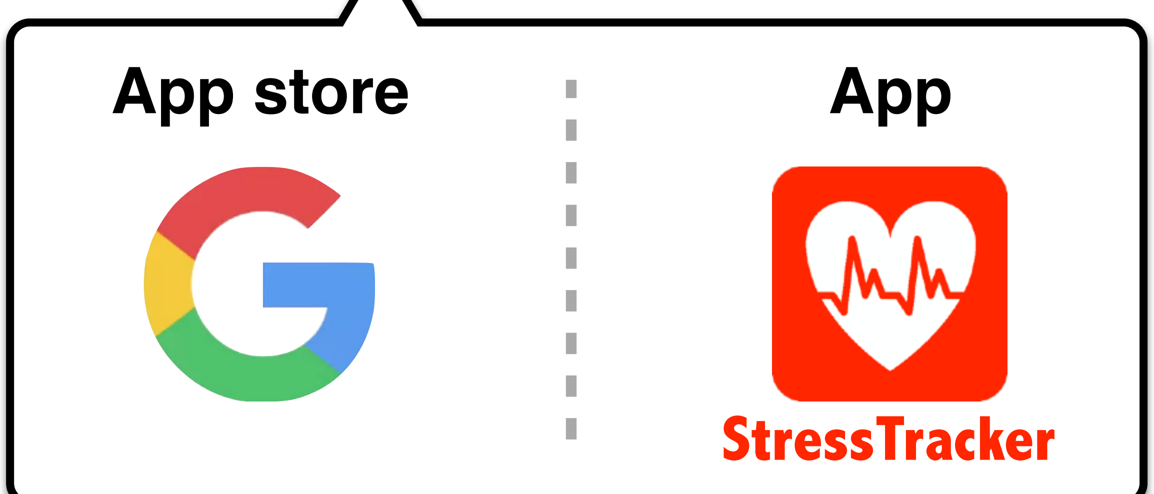
Private sums: A “straw-man” scheme

[Chaum88], [BGW88], …
[KDK11] [DFKZ13] [PrivEx14] …



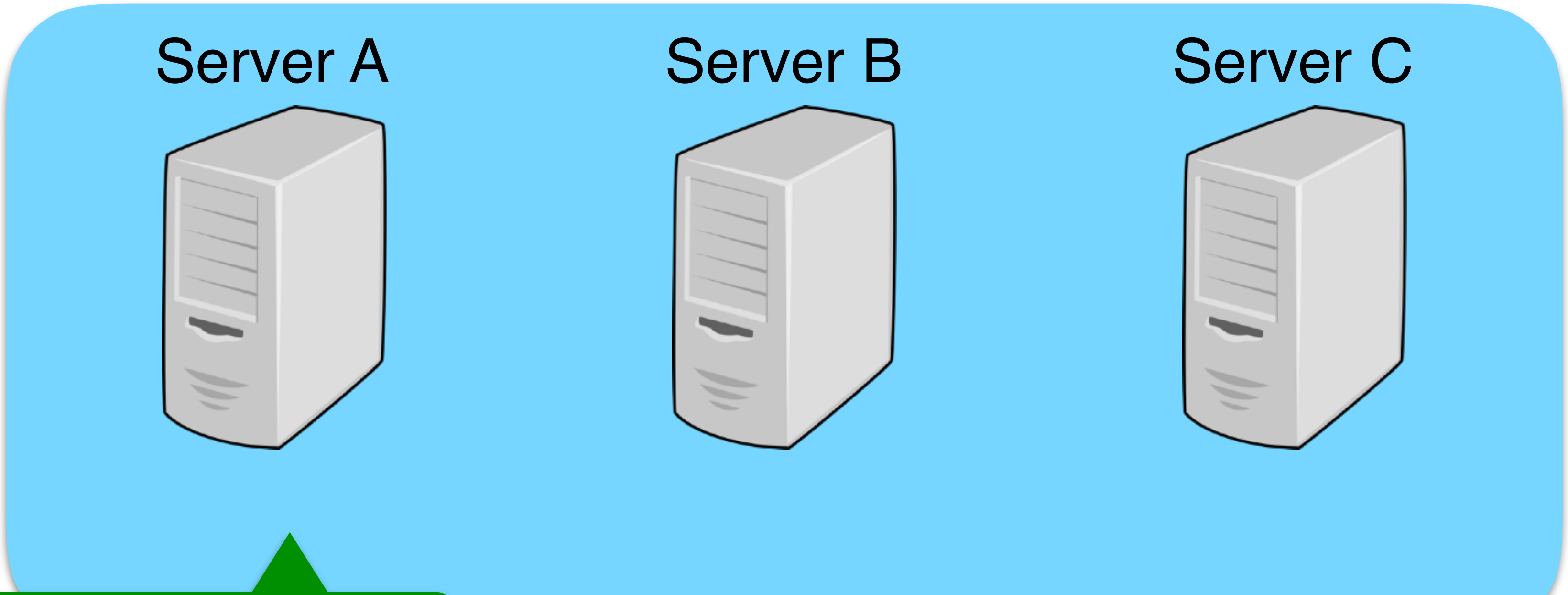
Assume that the servers are non-colluding.

Equivalently: that at least one server is honest.



Private sums: A “straw-man” scheme

[Chaum88], [BGW88], …
[KDK11] [DFKZ13] [PrivEx14] …

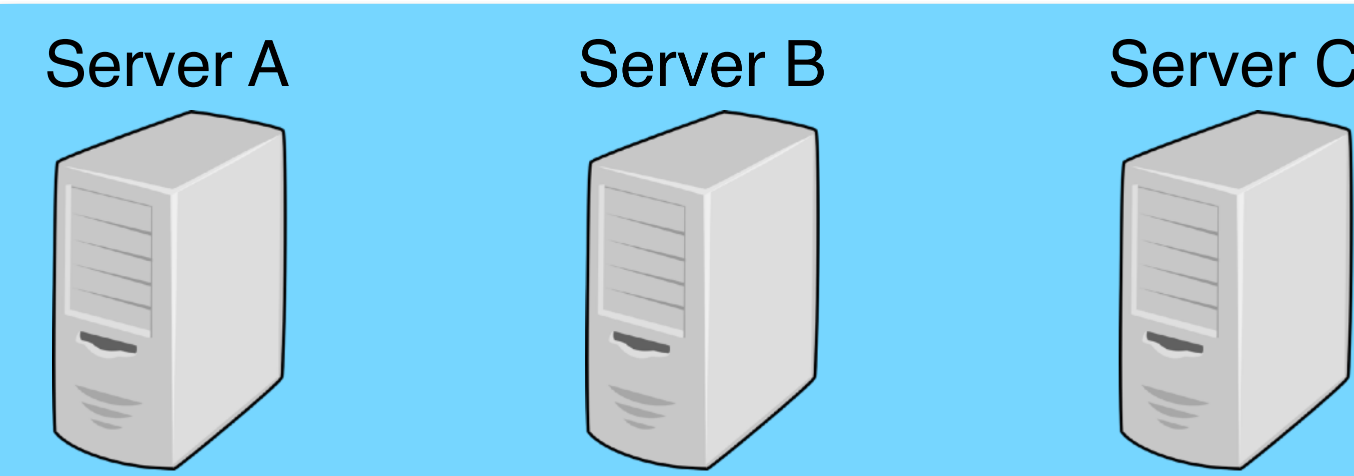


Assume that the servers are non-colluding.

Equivalently: that at least one server is honest.

Private sums: A "straw-man" scheme

[Chaum88], [BGW88], …
[KDK11] [DFKZ13] [PrivEx14] …



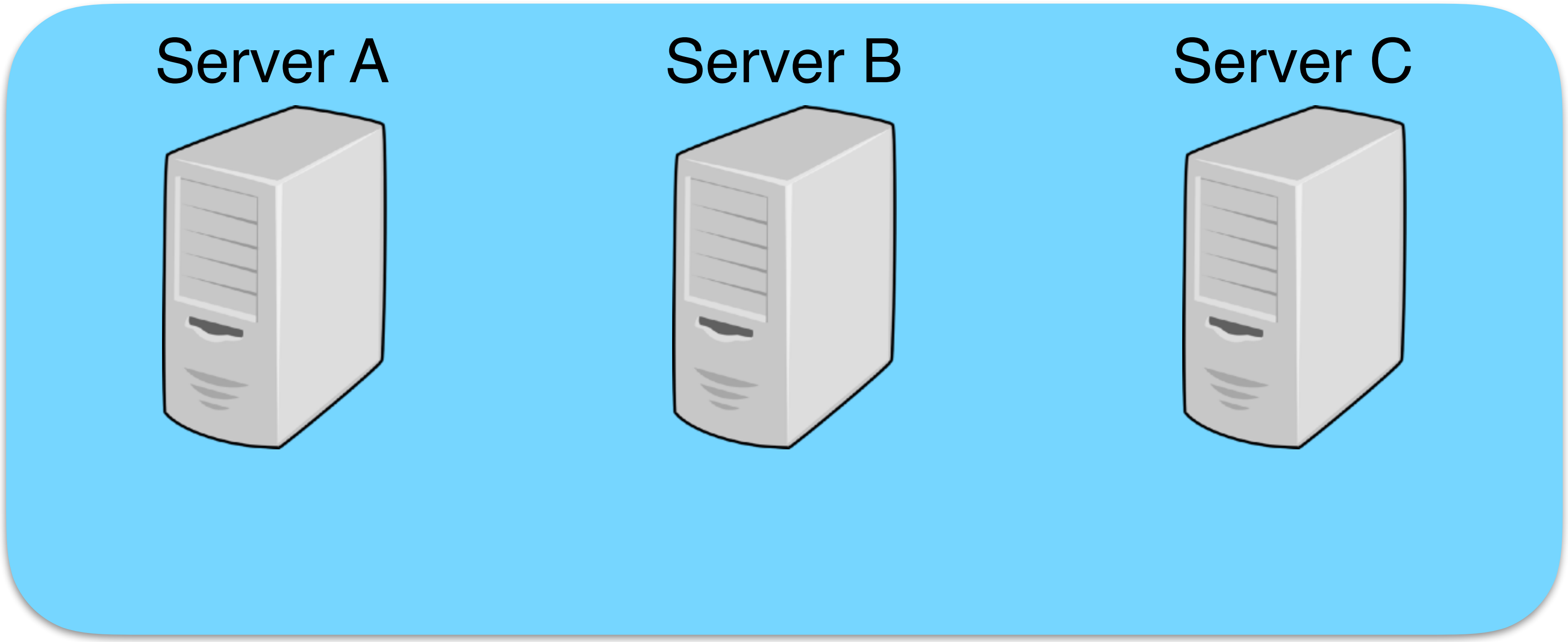
Assume that the servers are non-colluding.

Equivalently: that at least one server is honest.

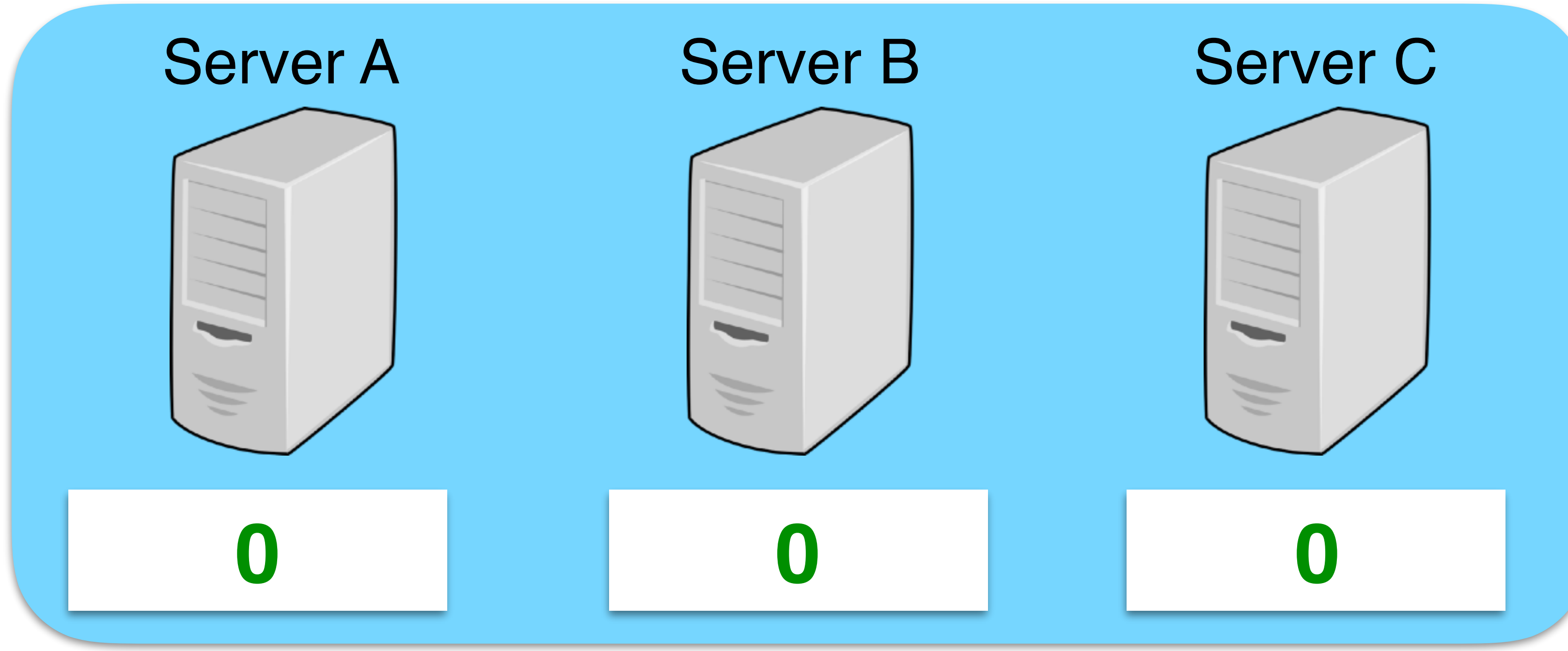
Spain **Germany** **Iceland**

Private sums: A “straw-man” scheme

[Chaum88], [BGW88], …
[KDK11] [DFKZ13] [PrivEx14] …



Private sums: A “straw-man” scheme



1

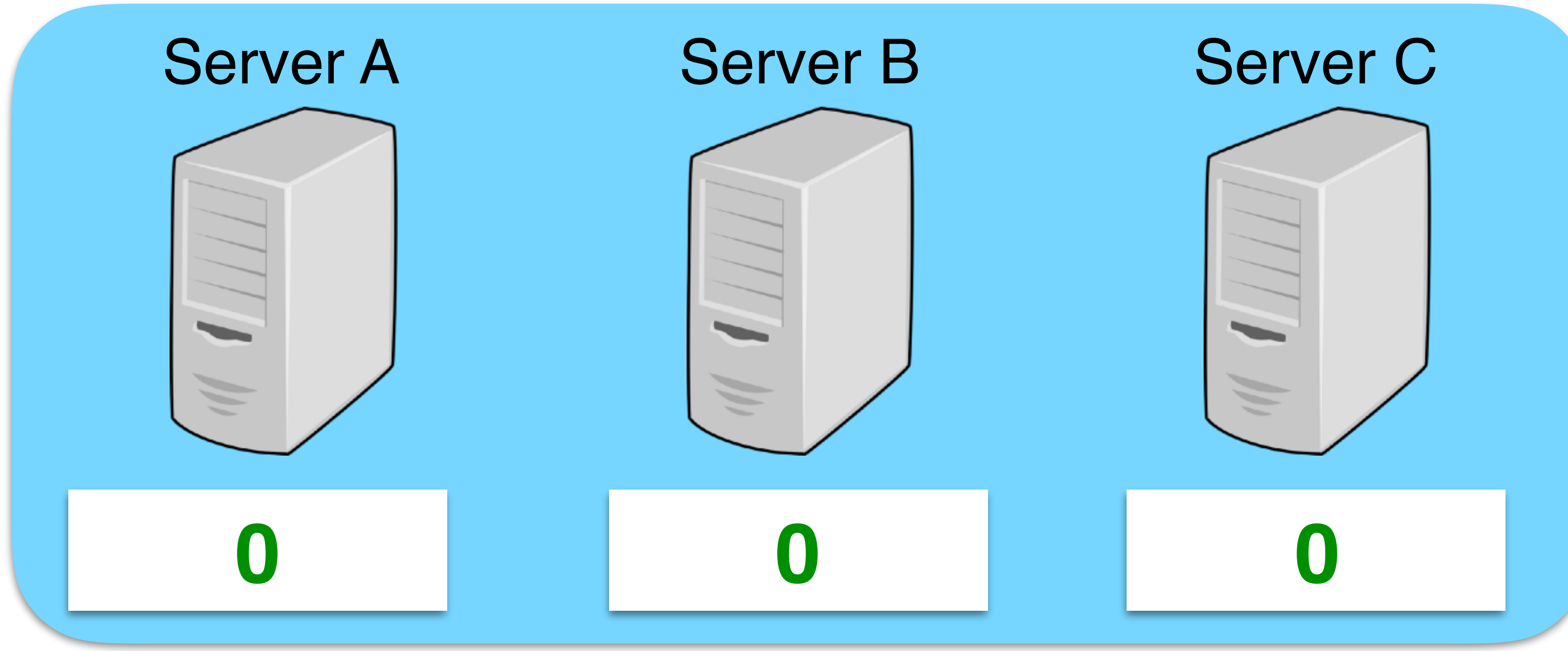
<u>Secret sharing</u>

Pick three random “shares” that sum to 1.

1 = 15 + (-12) + (- 2) (mod 31)

Need all three shares to recover the shared value.

Private sums: A "straw-man" scheme



1

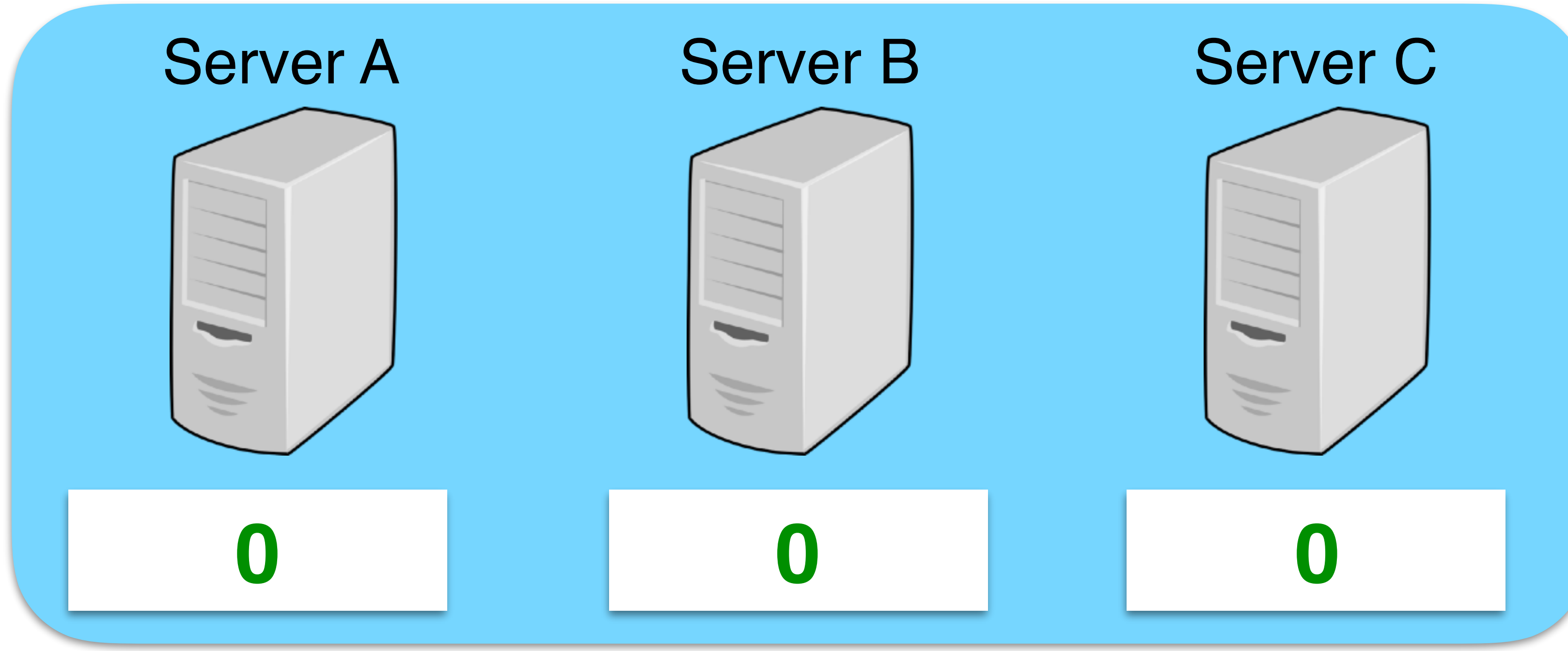
<u>Secret sharing</u>

Pick three random "shares" that sum to 1.

$1 = 15 + (-12) + (-2) \pmod{31}$

Need all three shares to recover the shared value.

Private sums: A “straw-man” scheme



15 -12 -2

1

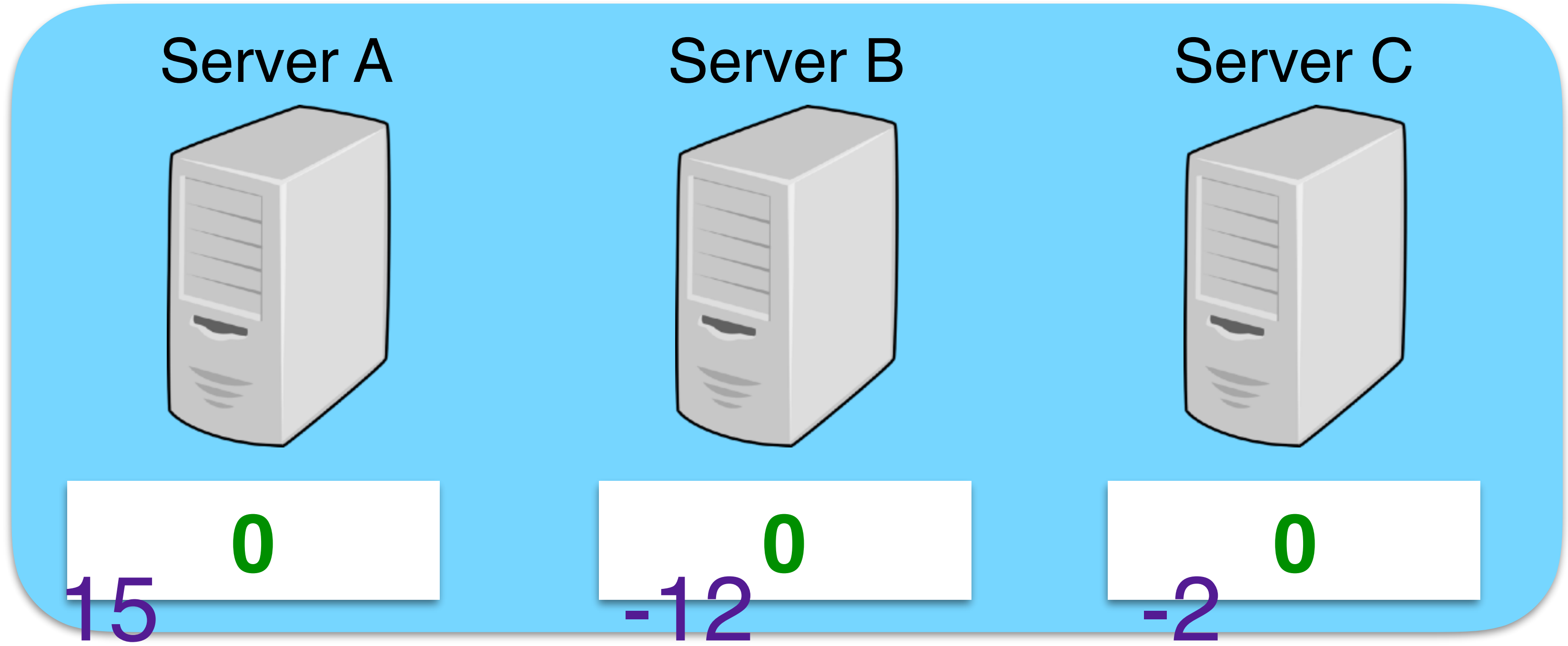
<u>Secret sharing</u>

Pick three random “shares” that sum to 1.

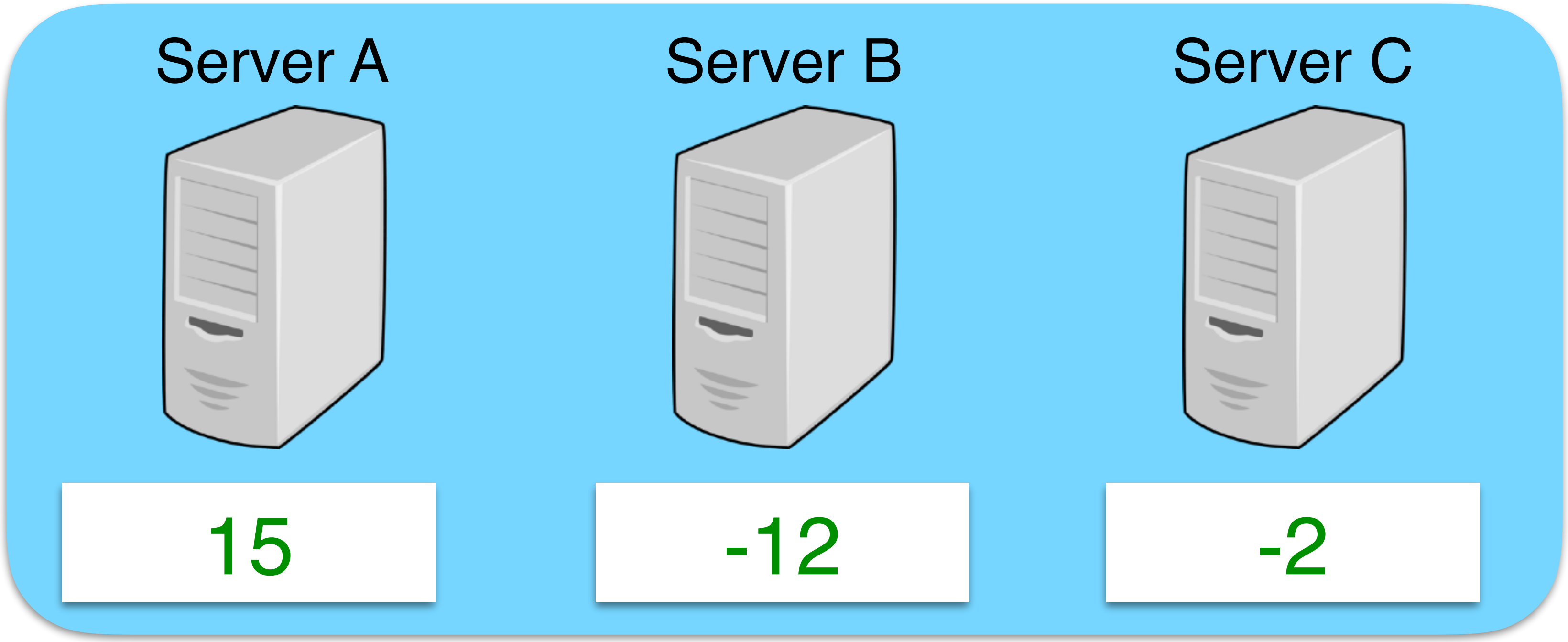
$1 = 15 + (-12) + (-2) \pmod{31}$

Need all three shares to recover the shared value.

Private sums: A “straw-man” scheme

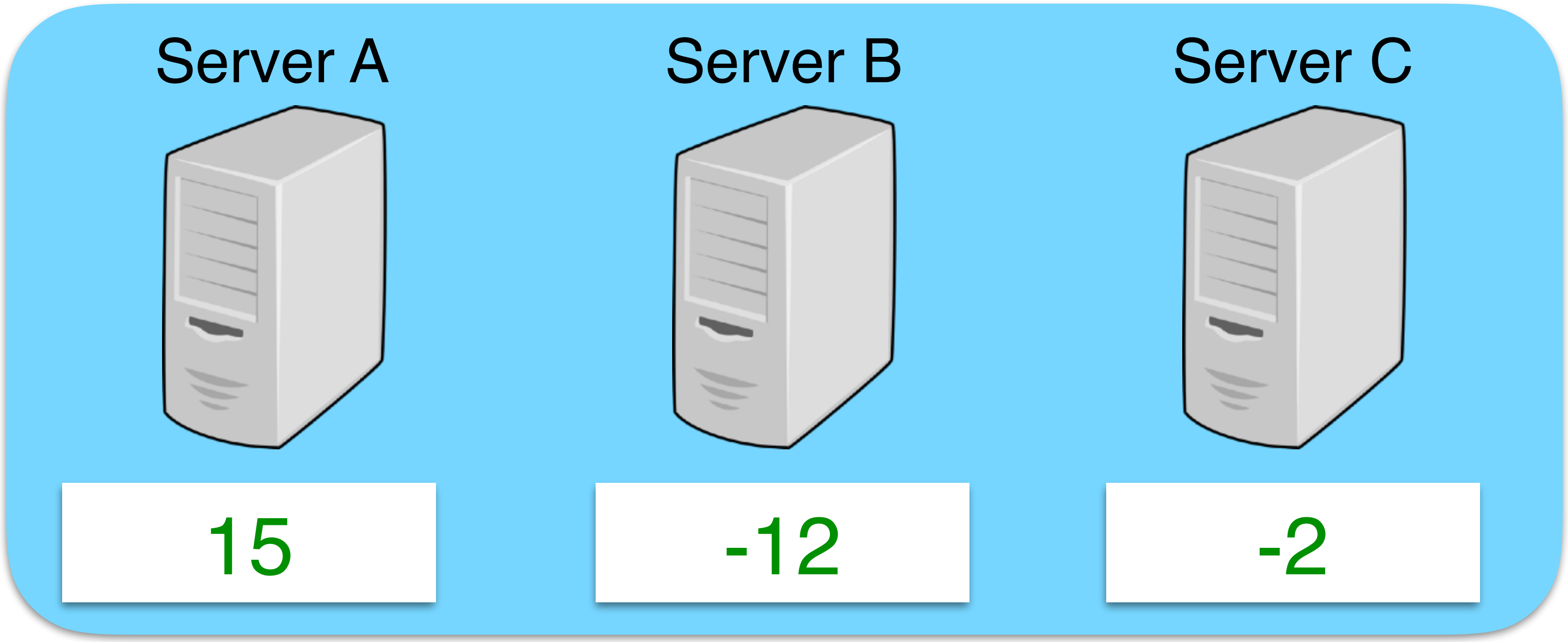


Private sums: A “straw-man” scheme

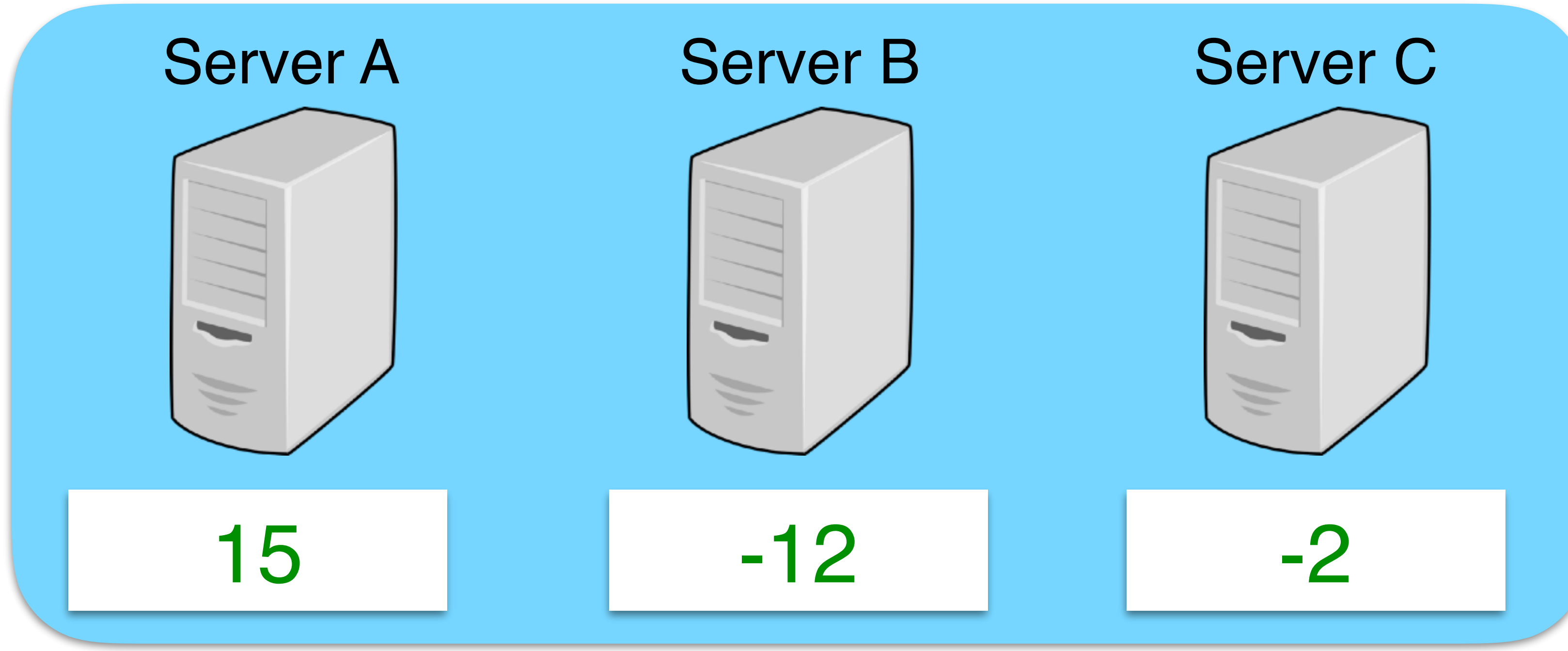


1

Private sums: A "straw-man" scheme

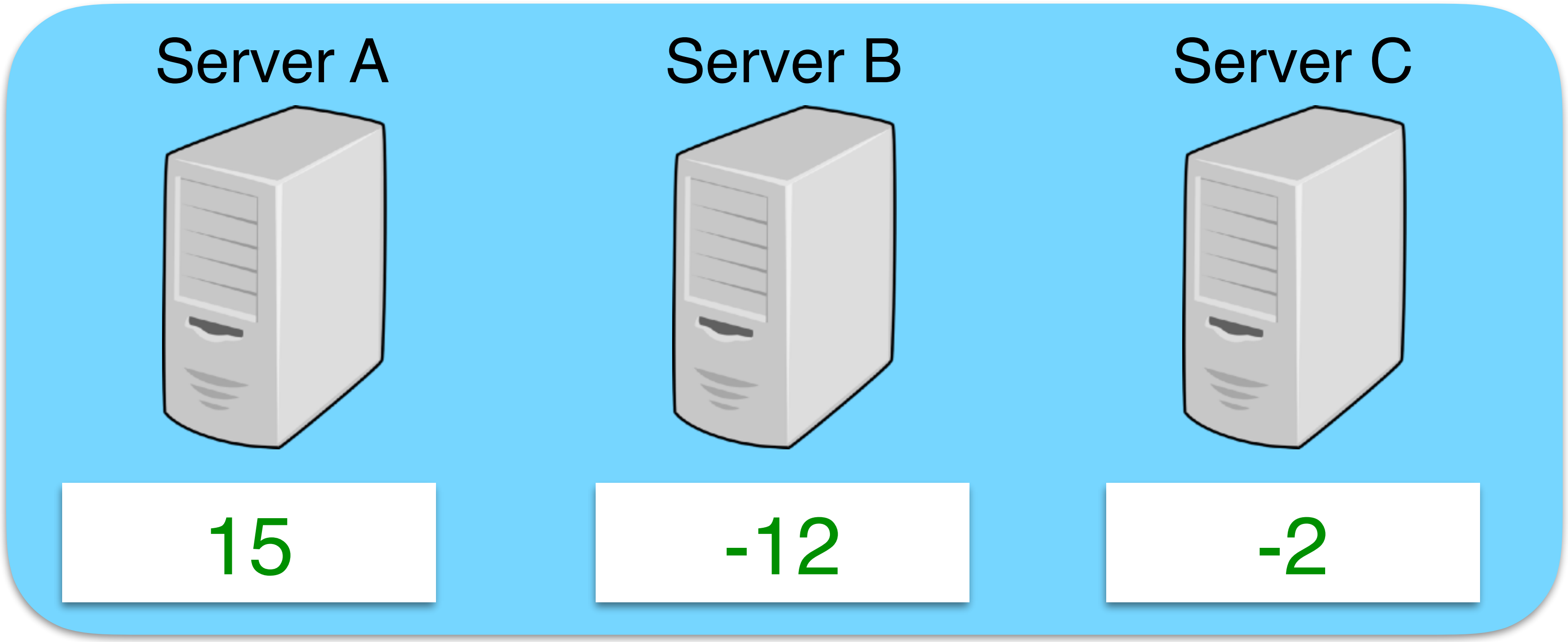


Private sums: A “straw-man” scheme



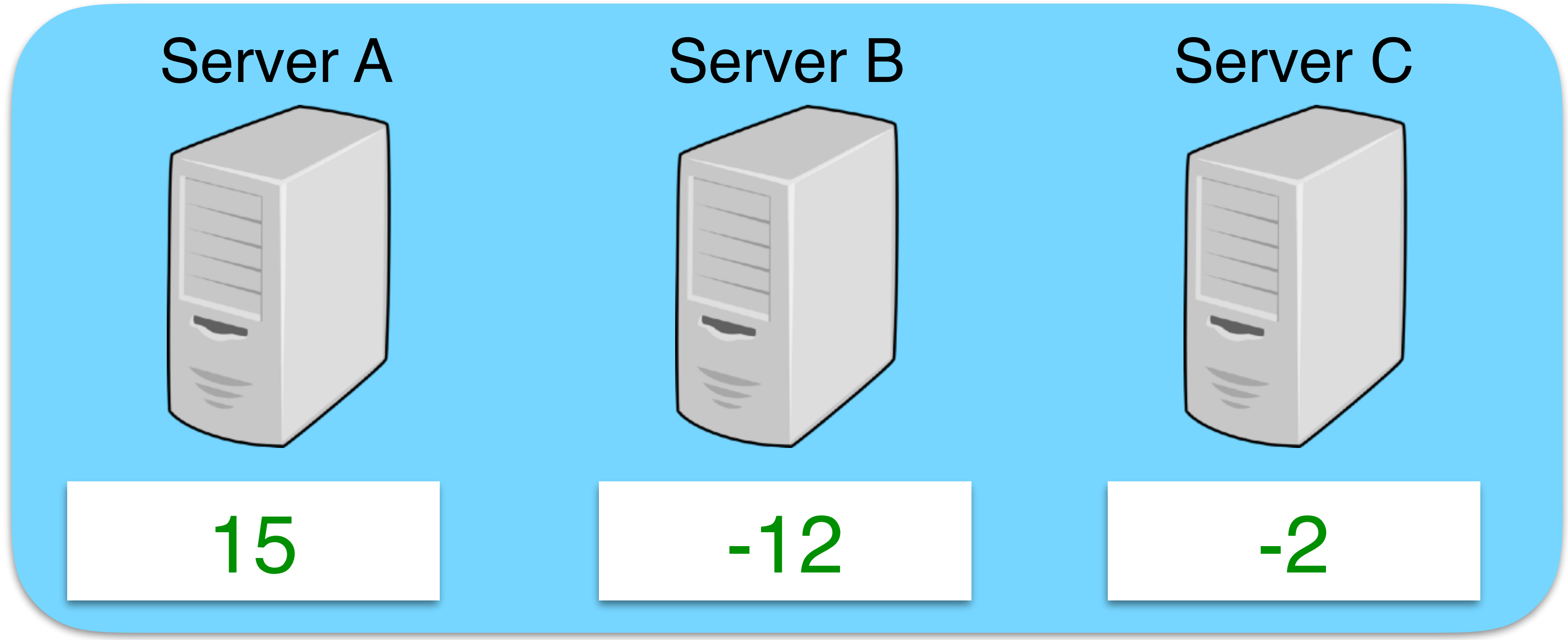
0 = (-10) + 7 + 3

Private sums: A "straw-man" scheme



$0 = (-10) + 7 + 3$

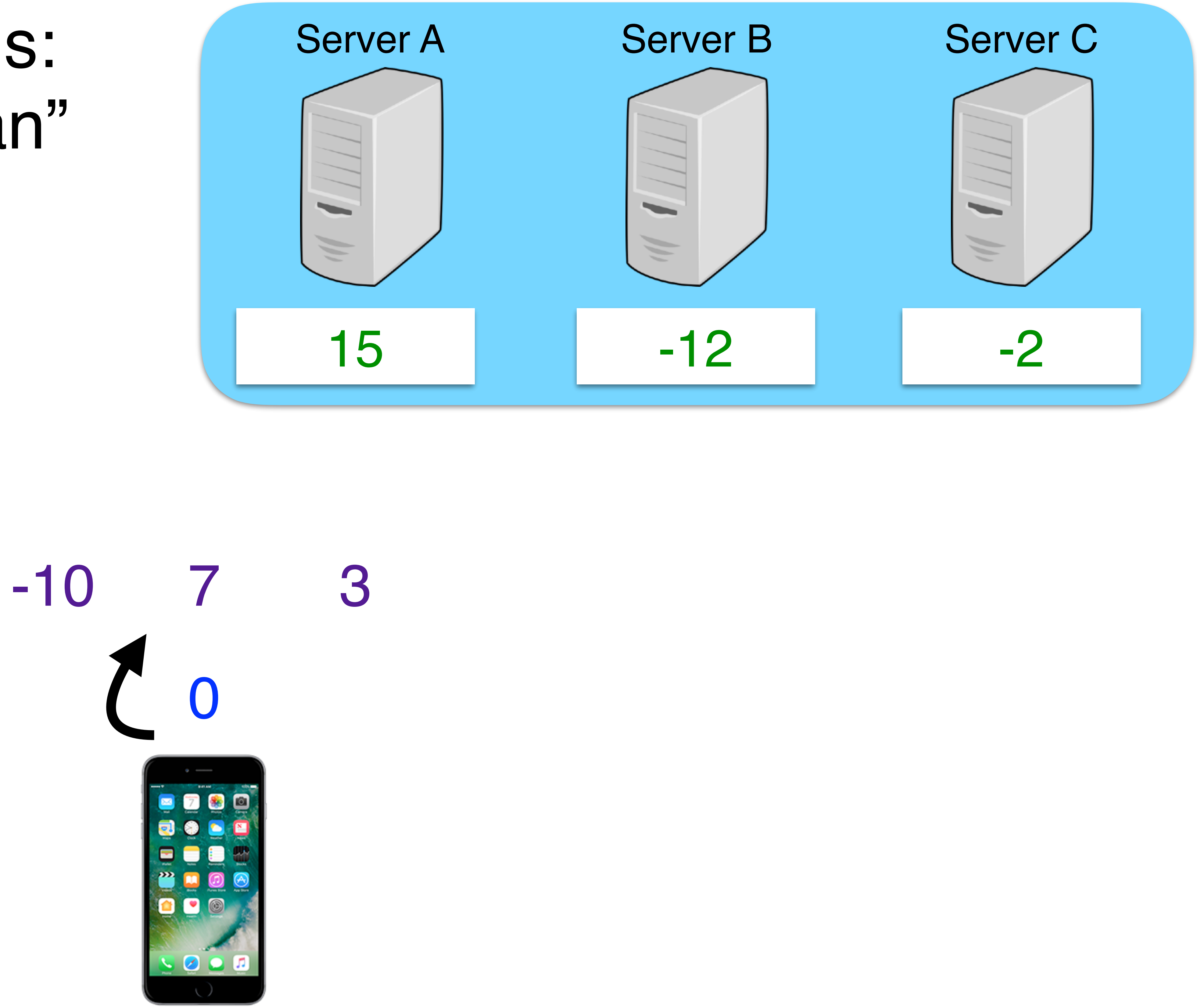
Private sums: A "straw-man" scheme



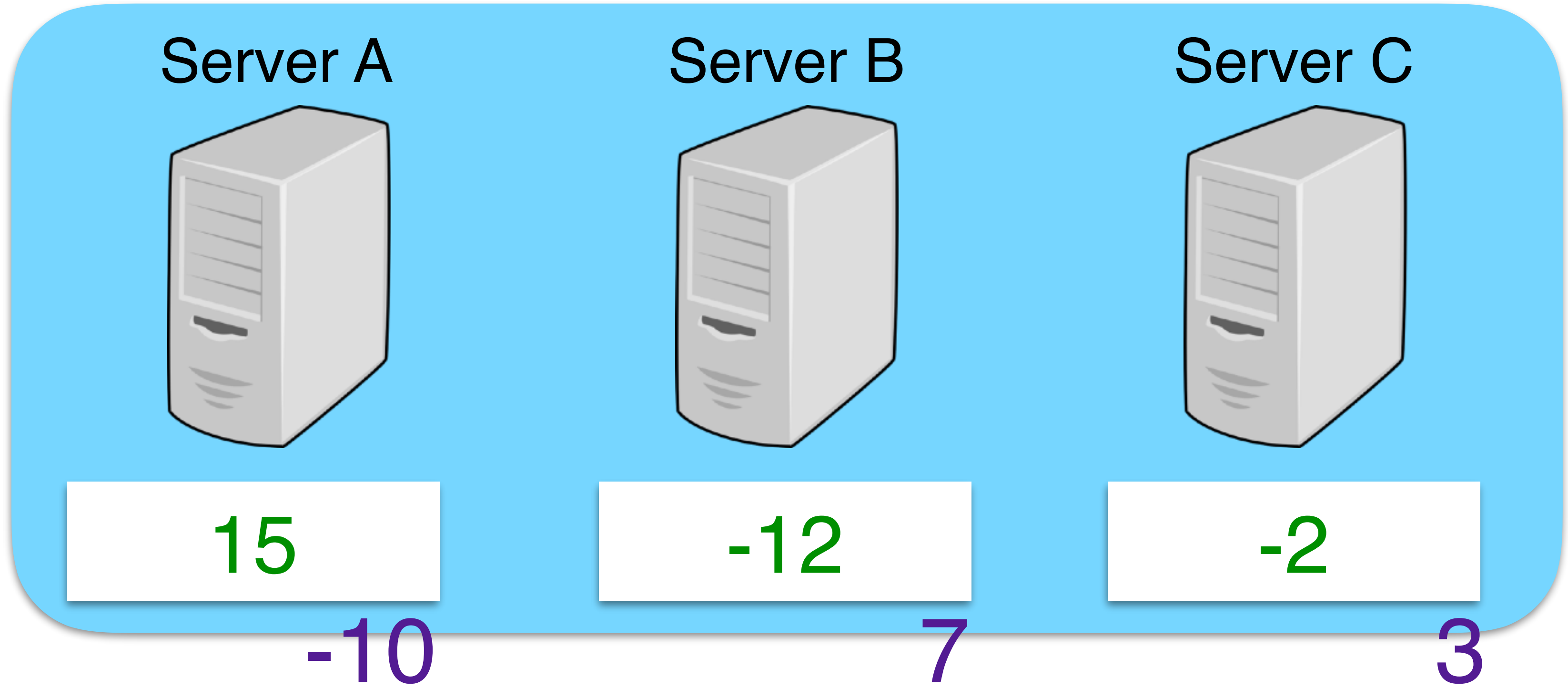
-10 7 3

$0 = (-10) + 7 + 3$

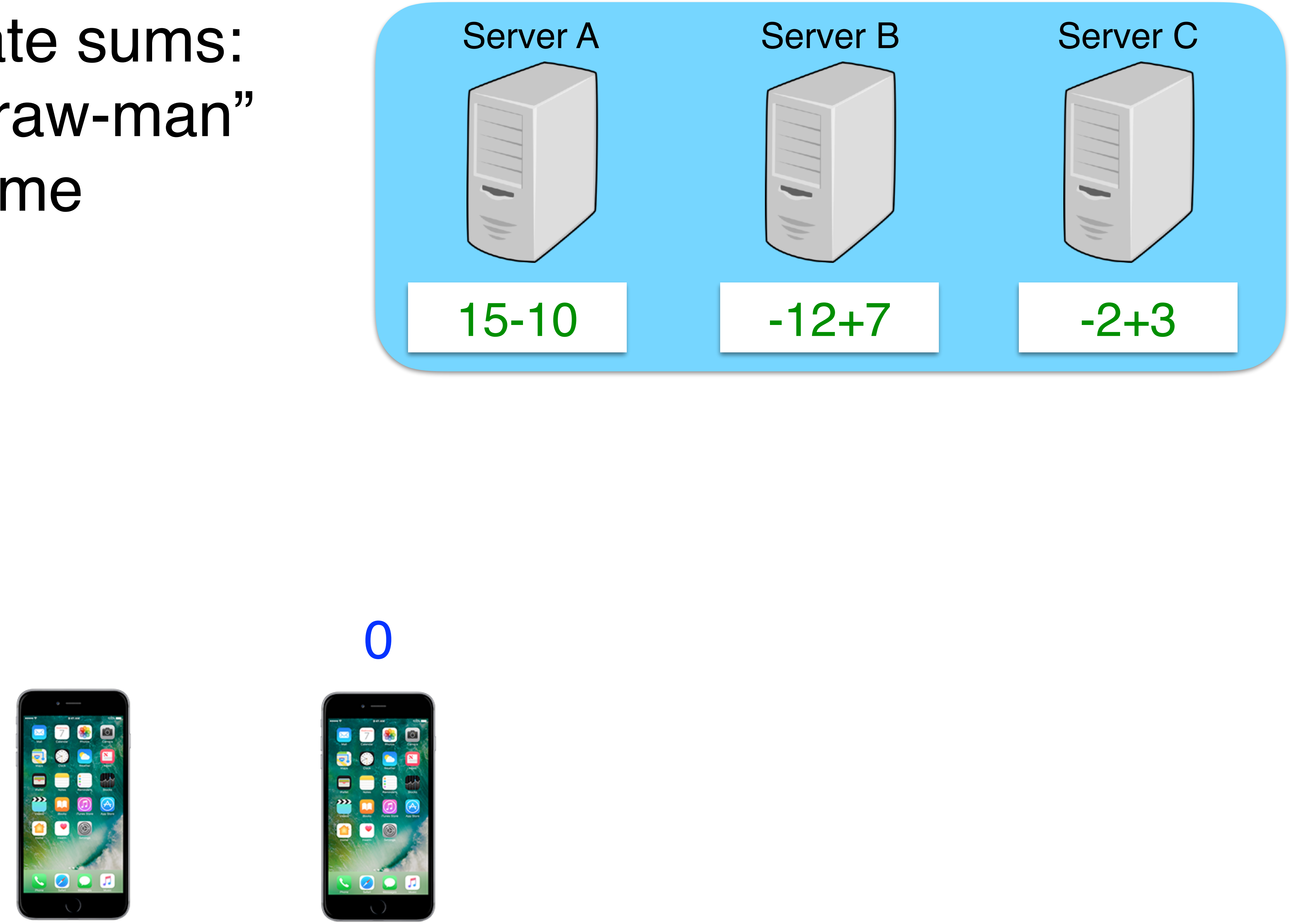
Private sums:
A “straw-man” scheme



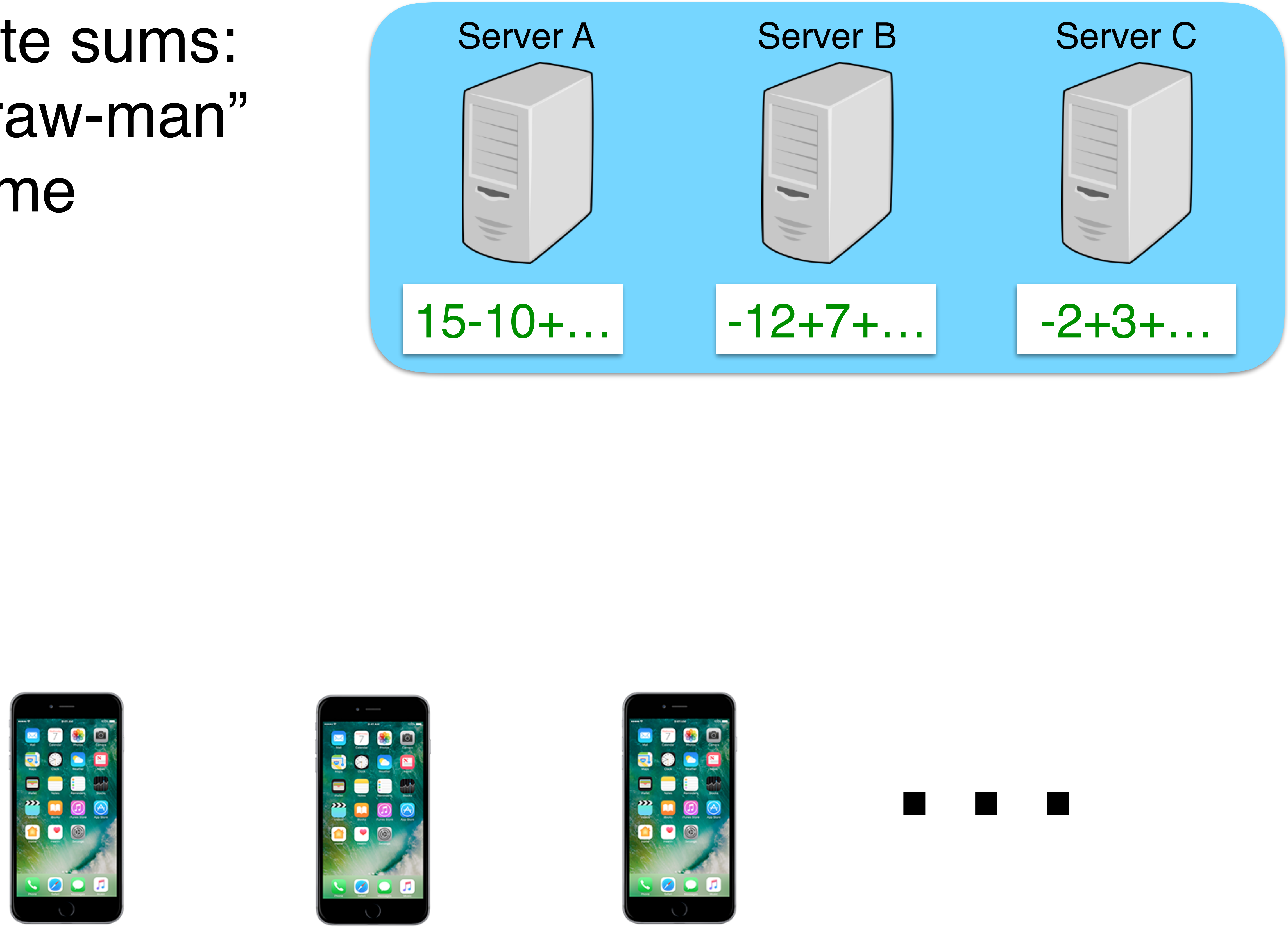
Private sums: A "straw-man" scheme



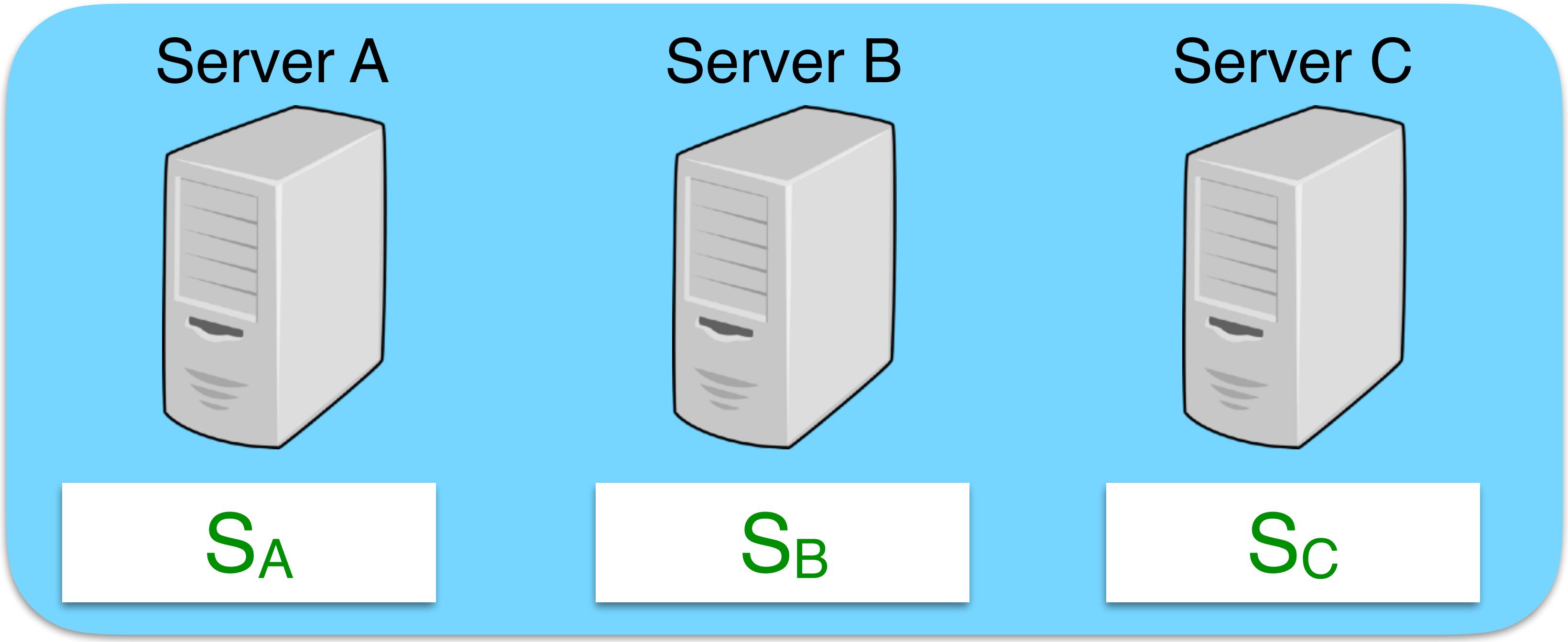
Private sums: A “straw-man” scheme



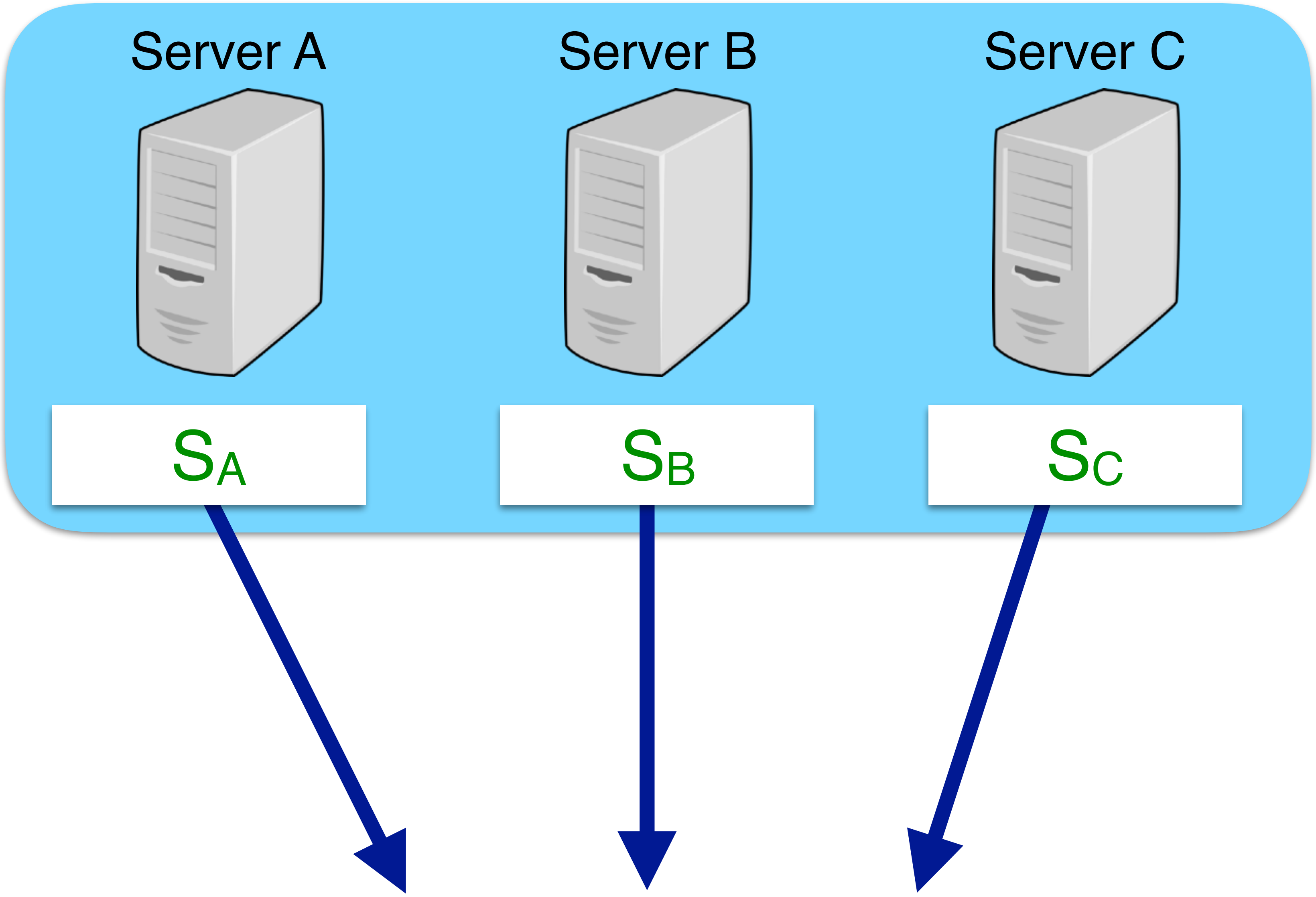
Private sums:
A “straw-man” scheme



Private sums: A "straw-man" scheme

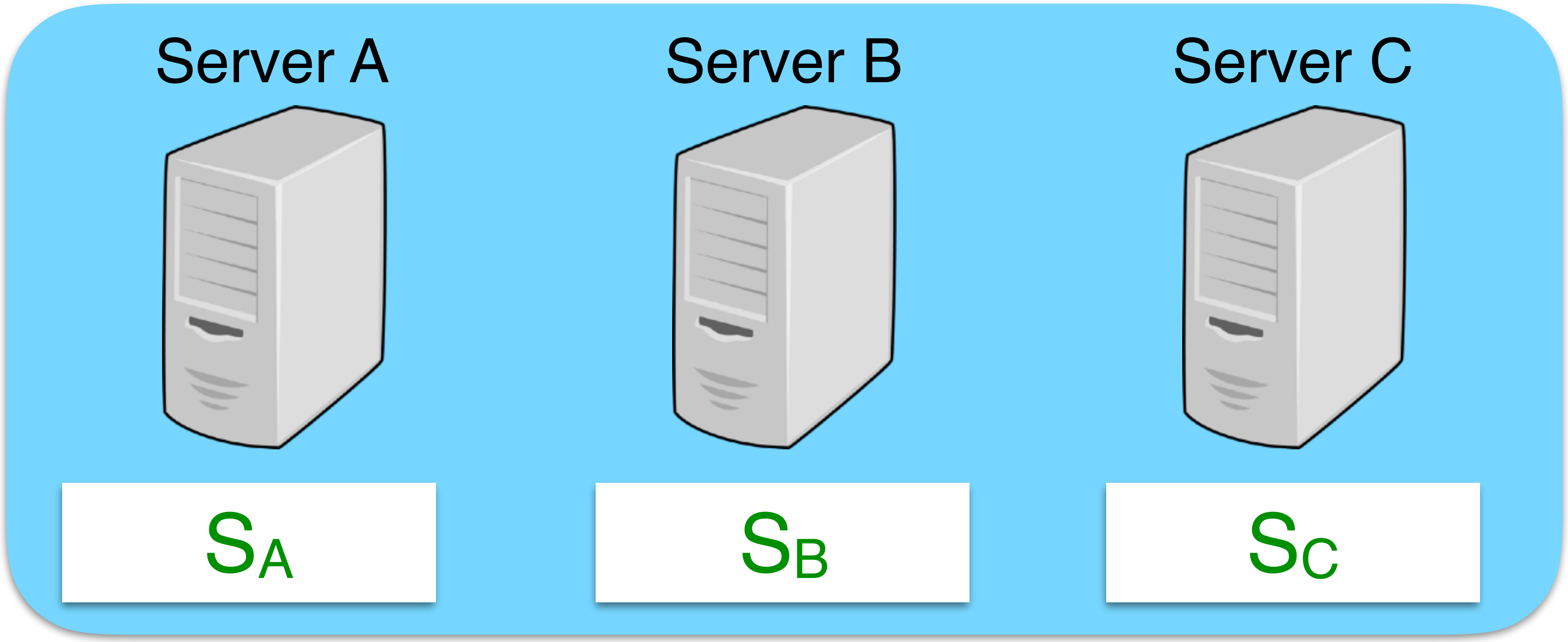


Private sums: A “straw-man” scheme



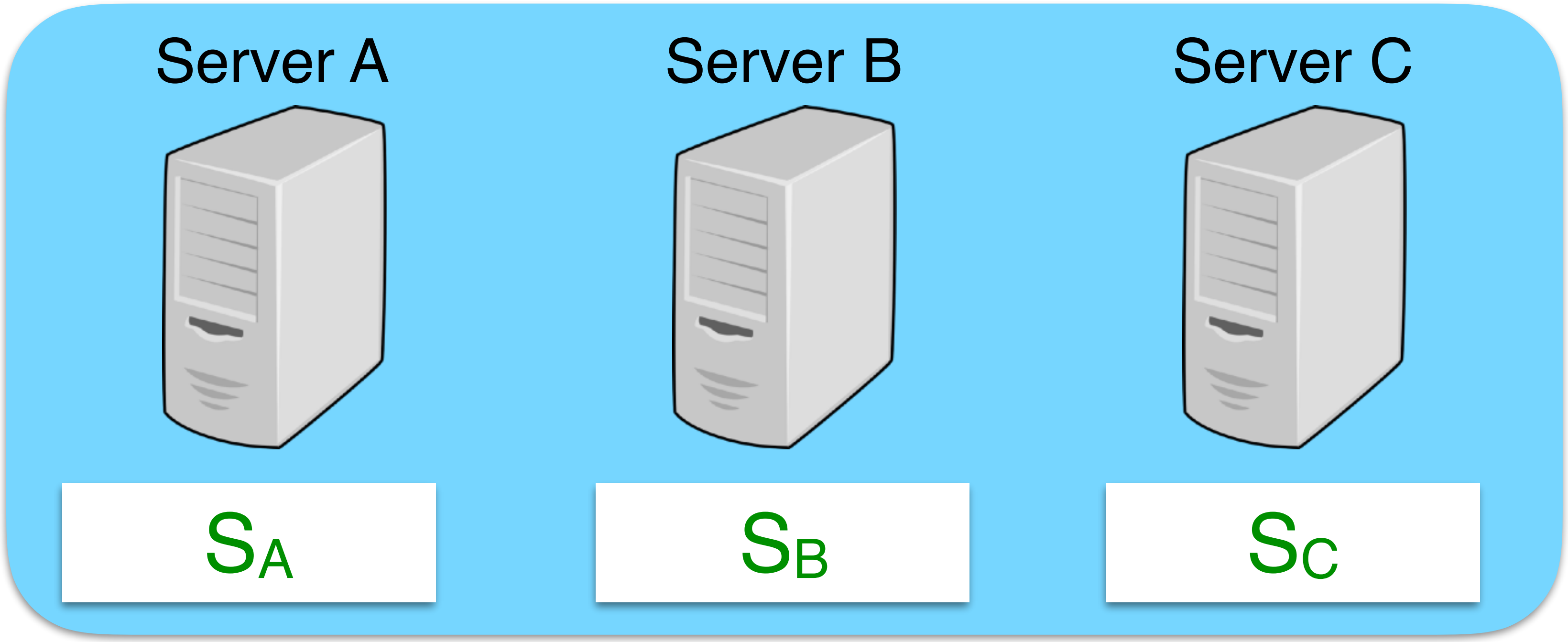
$S_A + S_B + S_C = 15 + -10 + \ldots$

Private sums: A “straw-man” scheme



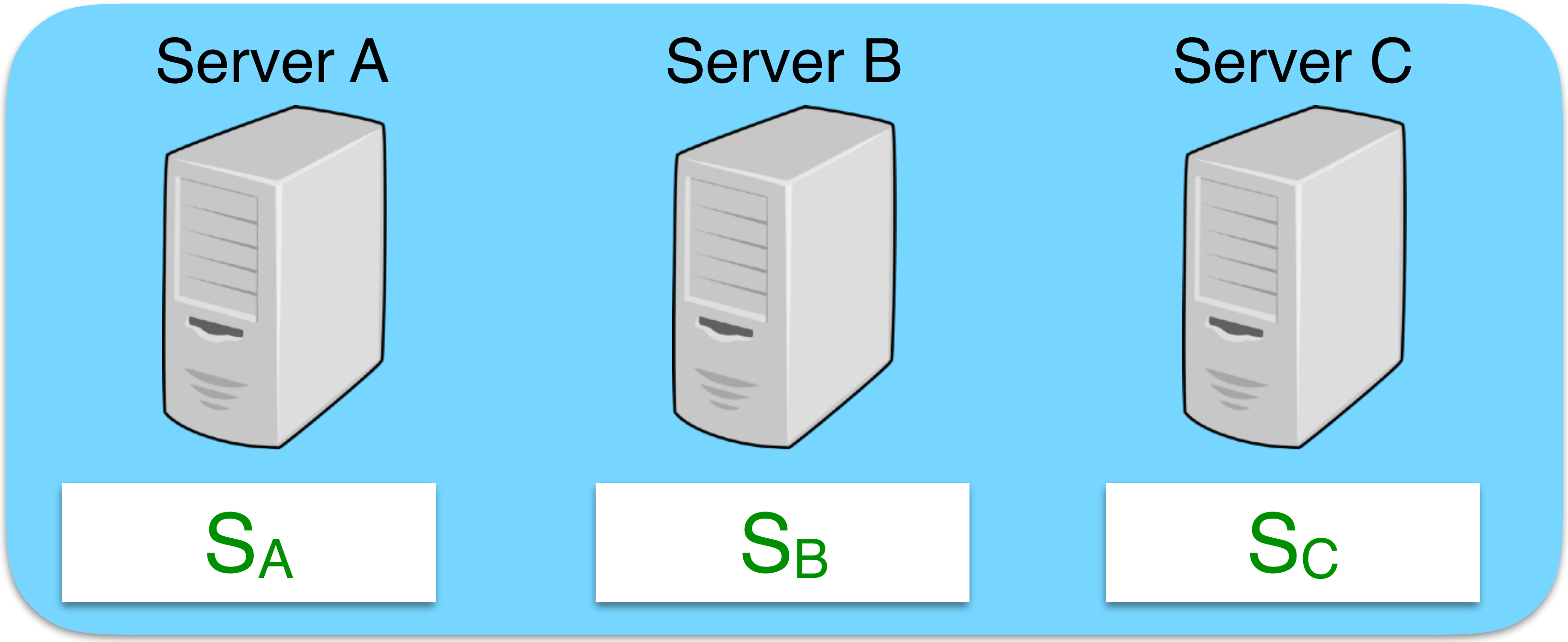
$S_A + S_B + S_C = 15 + -10 + \ldots$

Private sums: A “straw-man” scheme



$$
\begin{aligned}
S_A + S_B + S_C &= 15 + -10 + \ldots \\
&= 1 + 0 + \ldots + 1
\end{aligned}
$$

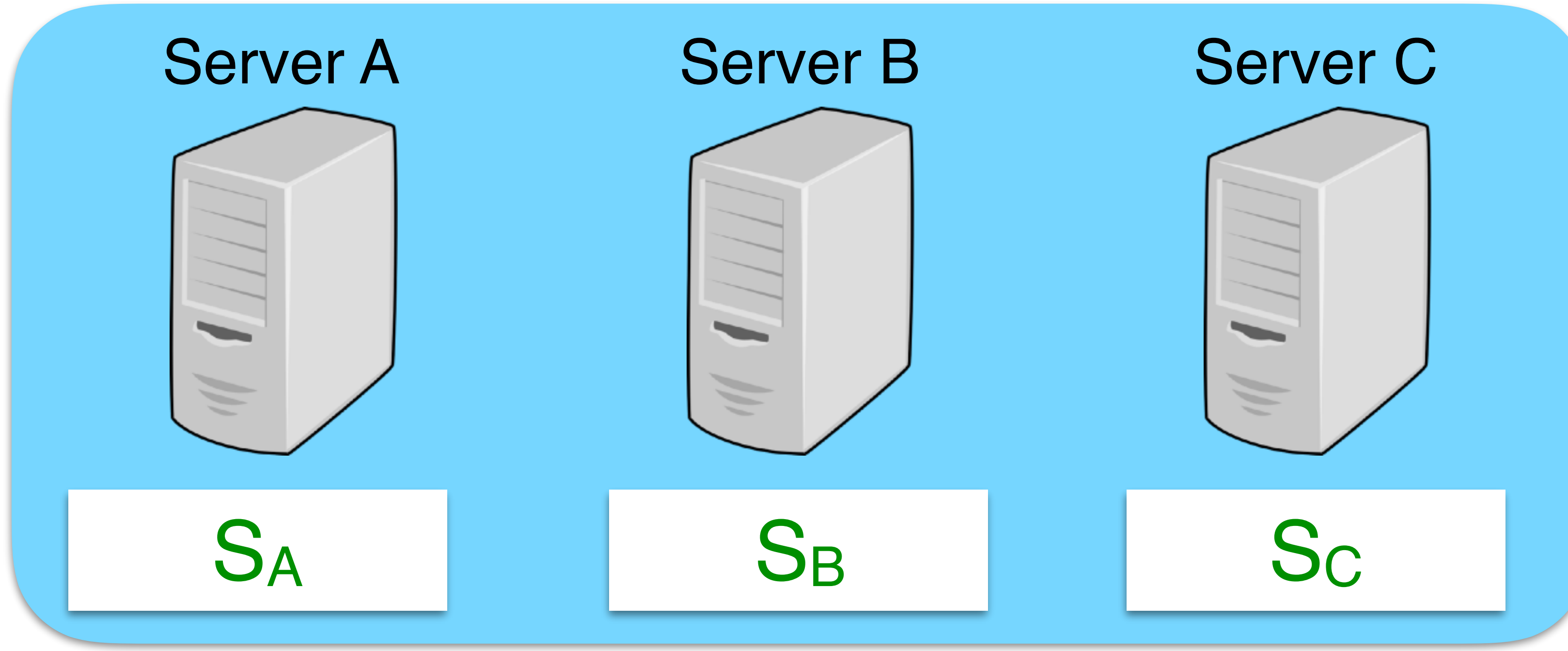
Private sums: A “straw-man” scheme



$S_A + S_B + S_C = 15 + -10 + \ldots$
$= 1 + 0 + \ldots + 1$

Servers learn the sum of client values and learn *nothing else*.

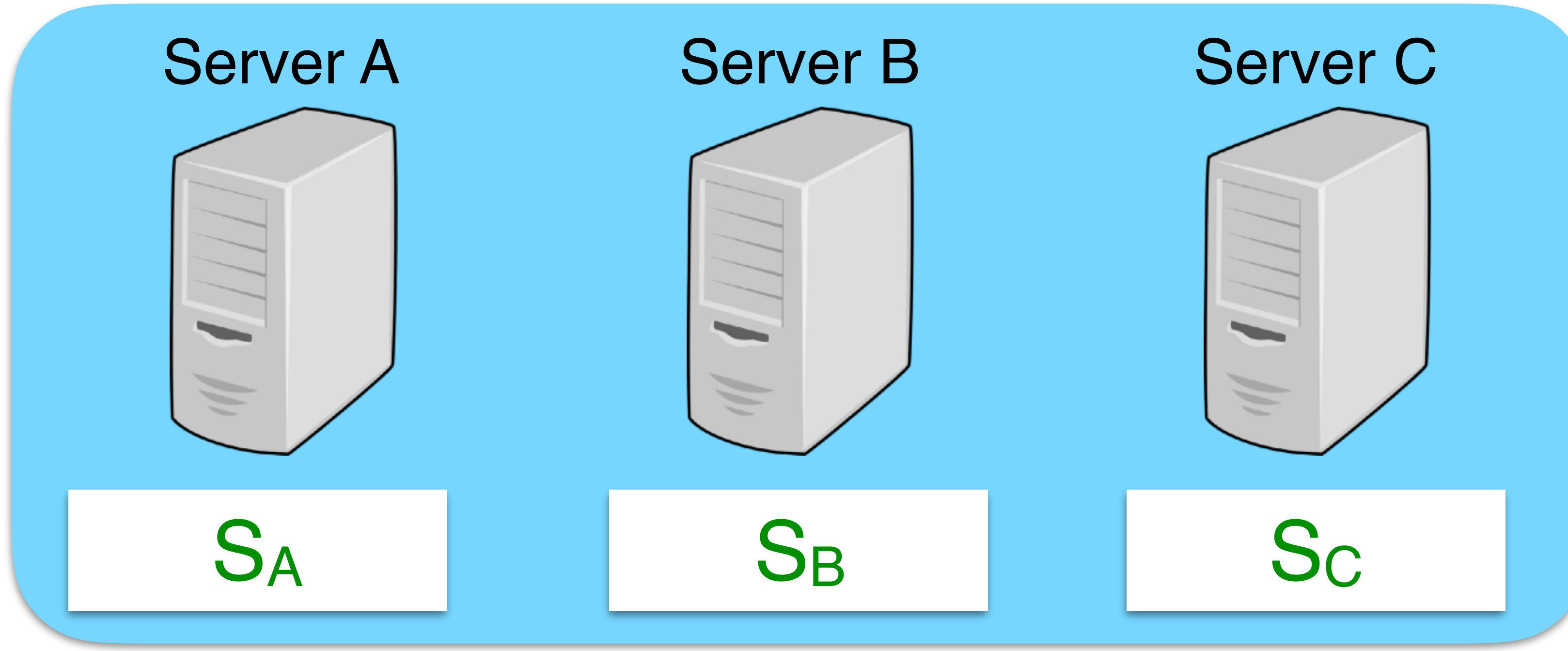
Private sums: A “straw-man” scheme



$$S_A + S_B + S_C = 15 + -10 + \ldots$$
$$= 1 + 0 + \ldots + 1$$

Servers learn the sum of client values and learn *nothing else*.

Private sums: A “straw-man” scheme



$$S_A + S_B + S_C = 15 + -10 + \ldots$$
$$= 1 + 0 + \ldots + 1$$

Learn that three phones are on the Bay Bridge—don’t know which three

Computing private sums

Computing private sums

Exact correctness: If everyone follows the protocol, servers compute the sum of all x_is.

Privacy: Any proper subset of the servers learns nothing but the sum of the x_is.

Efficiency: Follows by inspection.

Computing private sums

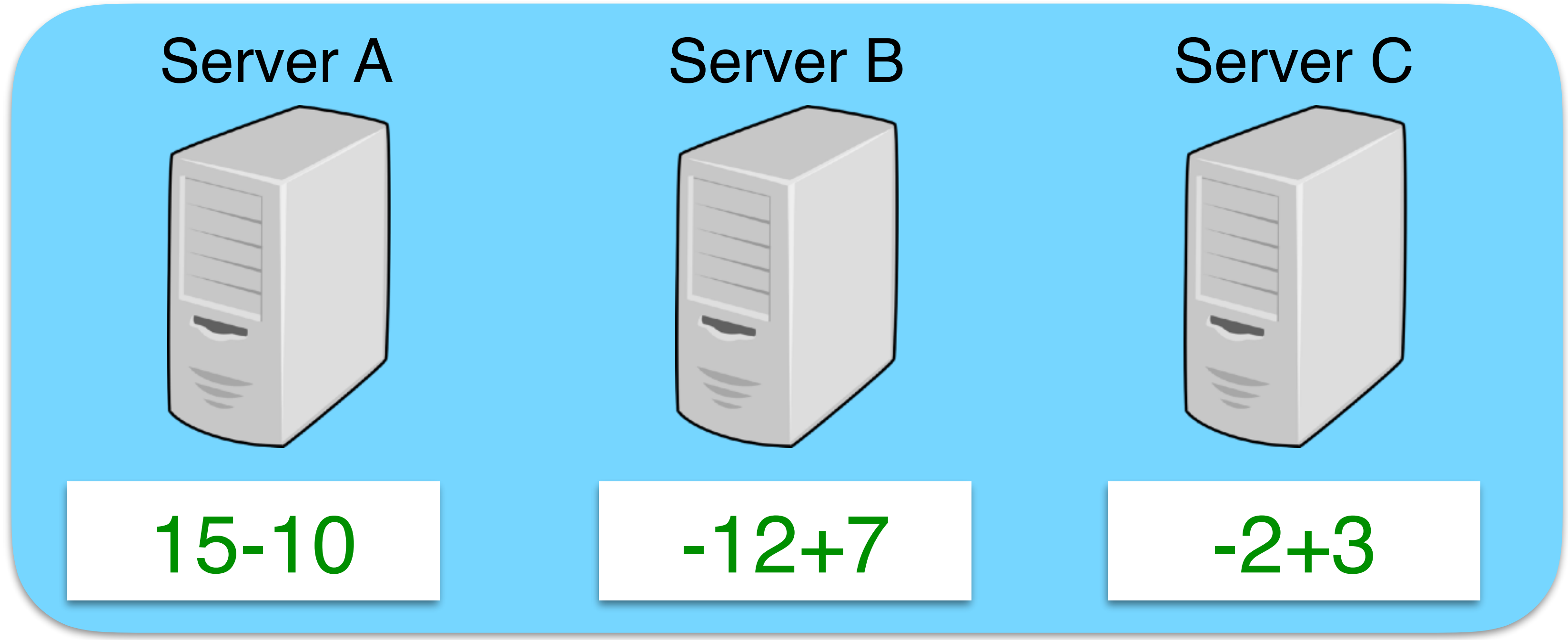
Exact correctness: If everyone follows the protocol, servers compute the sum of all x_is.

Privacy: Any proper subset of the servers learns nothing but the sum of the x_is.

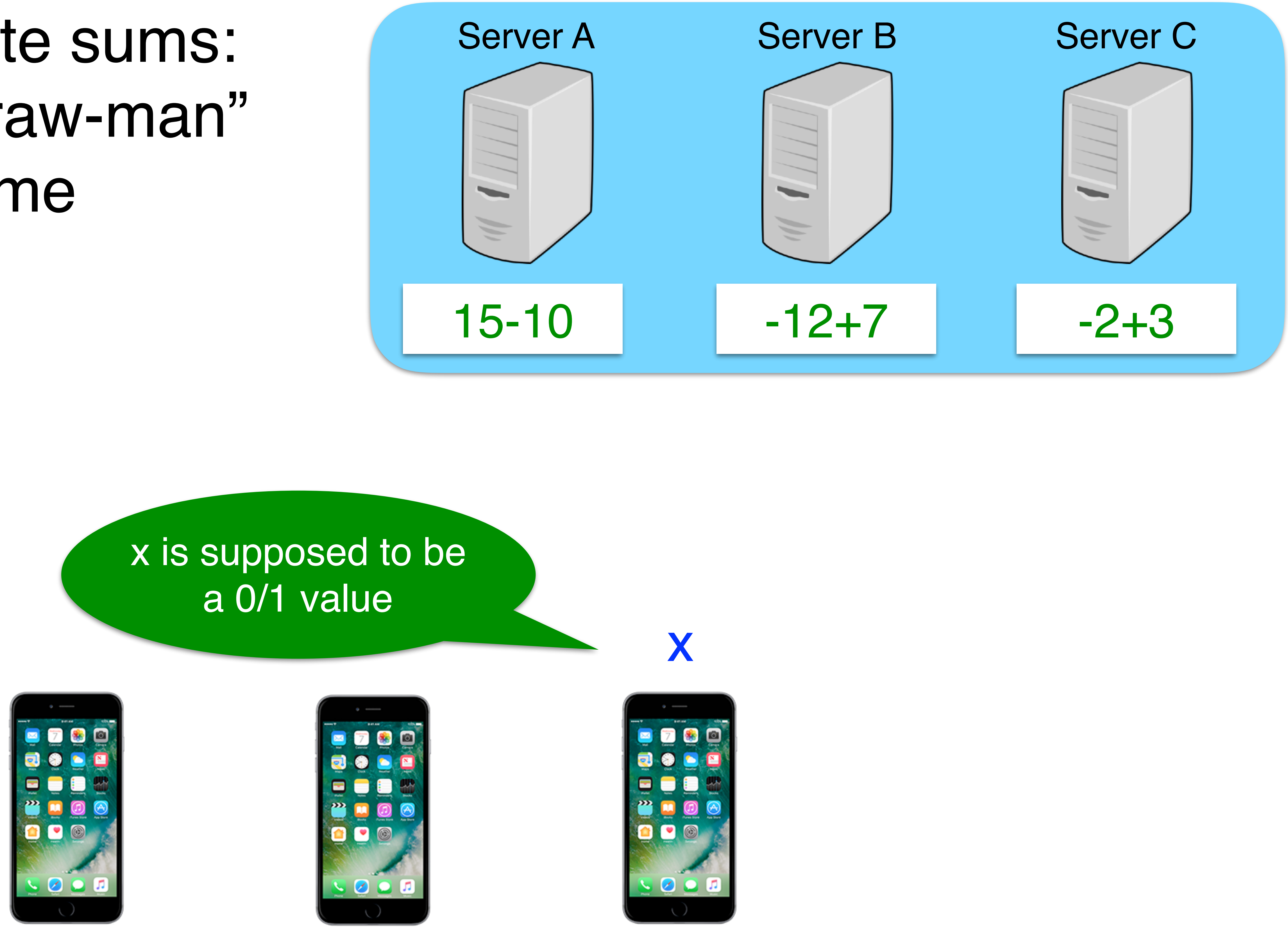
Efficiency: Follows by inspection.

Robustness: ???

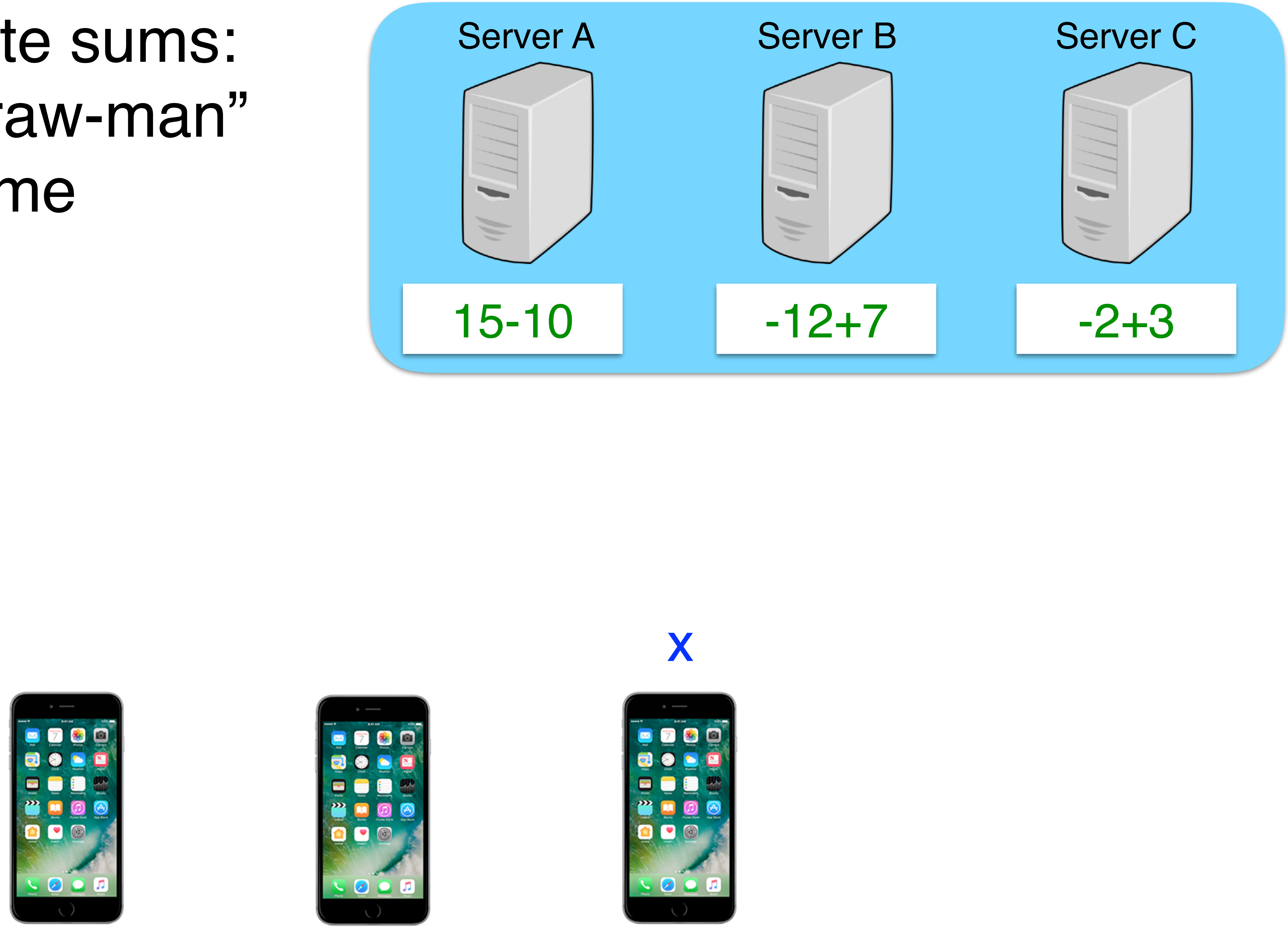
Private sums: A “straw-man” scheme



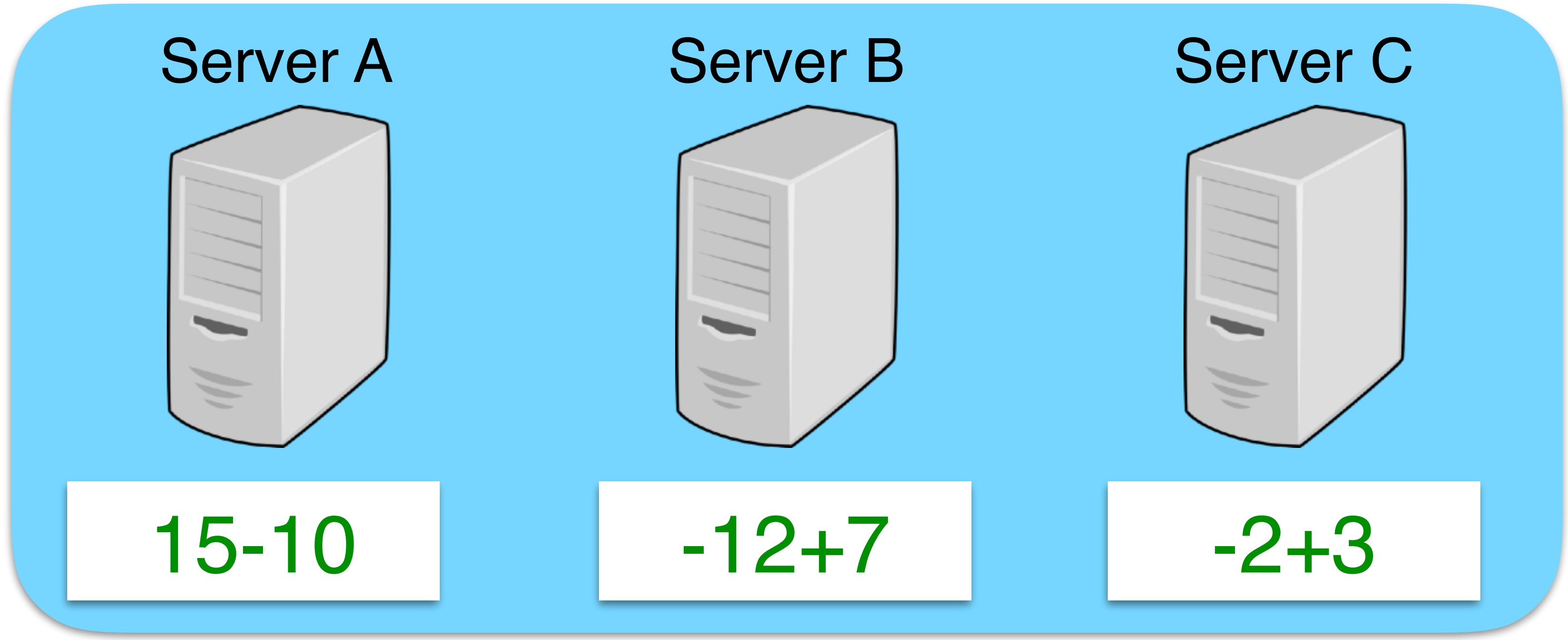
Private sums: A “straw-man” scheme



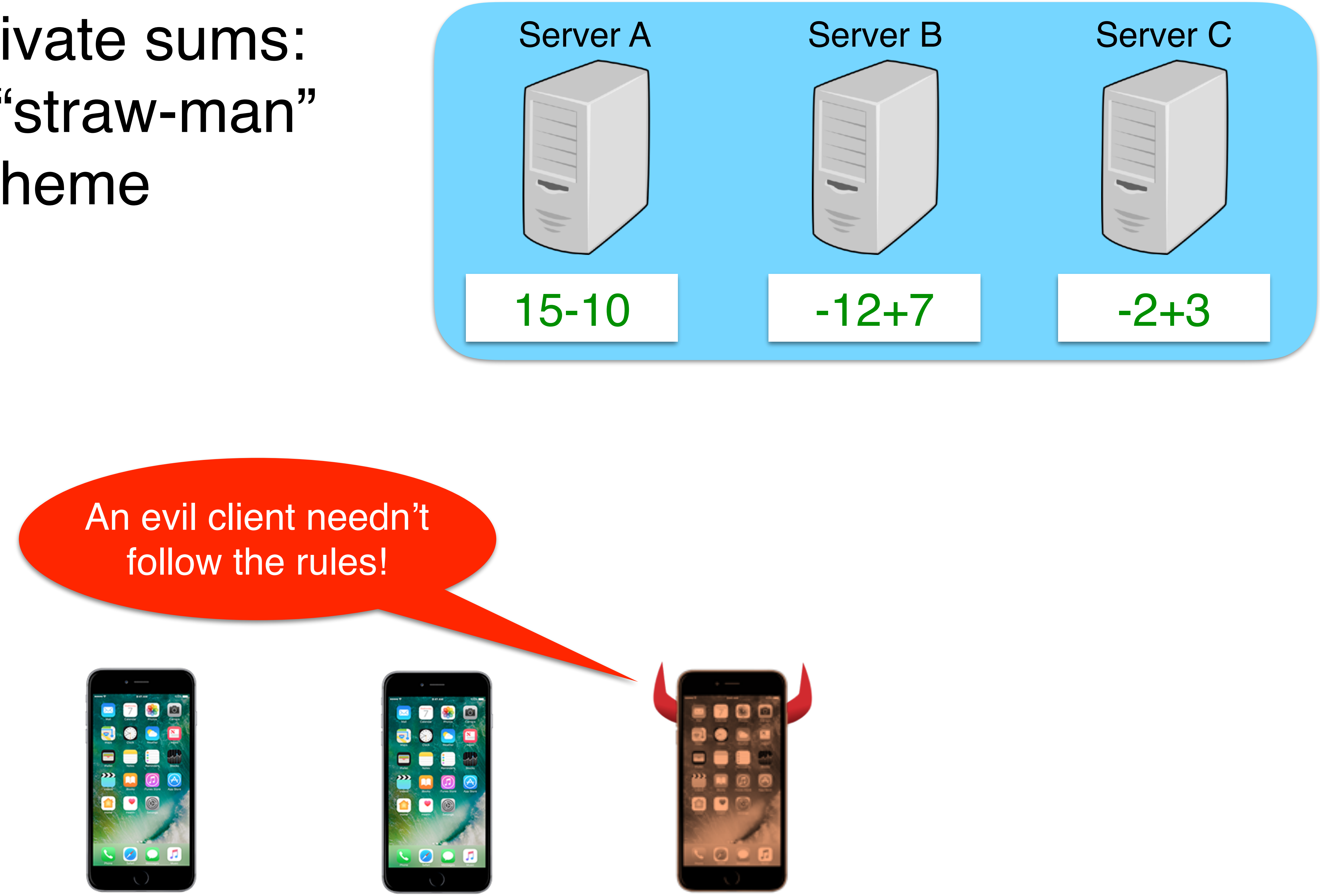
Private sums:
A “straw-man” scheme



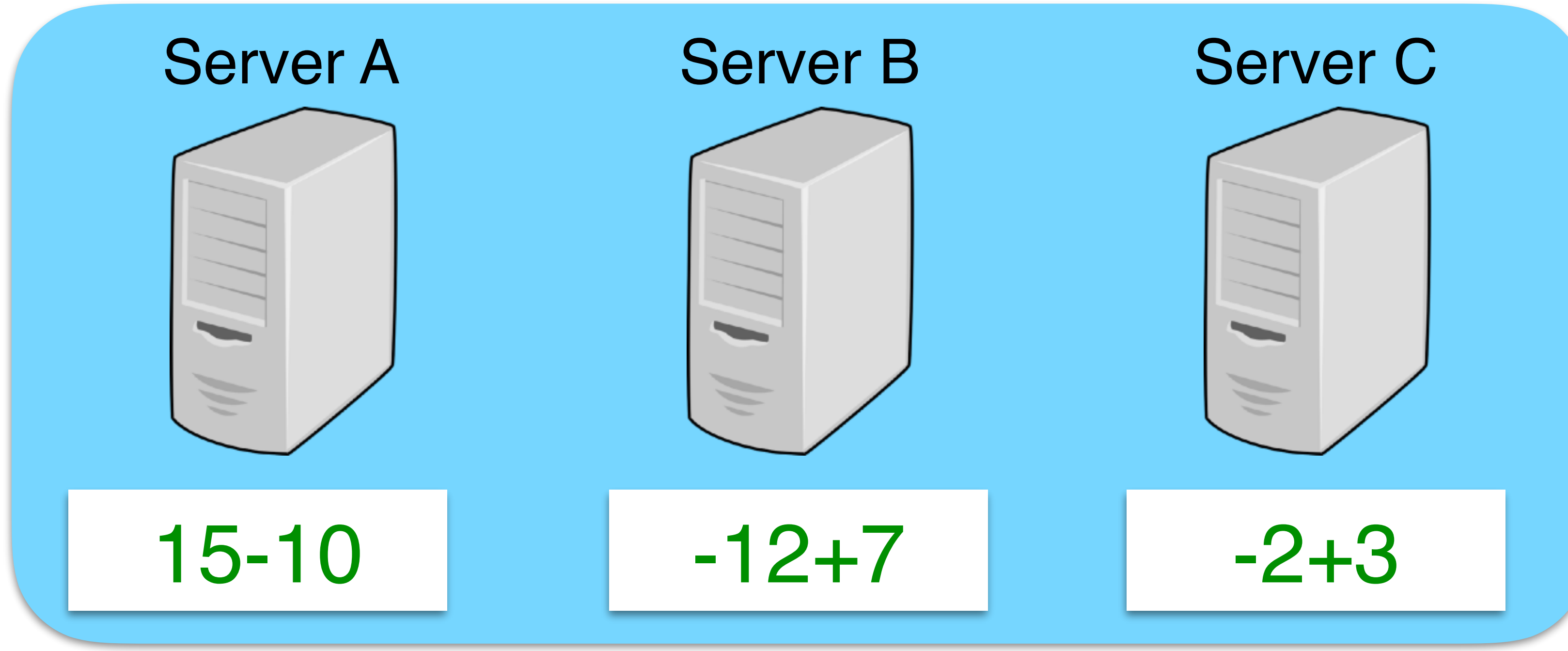
Private sums: A “straw-man” scheme



Private sums: A “straw-man” scheme



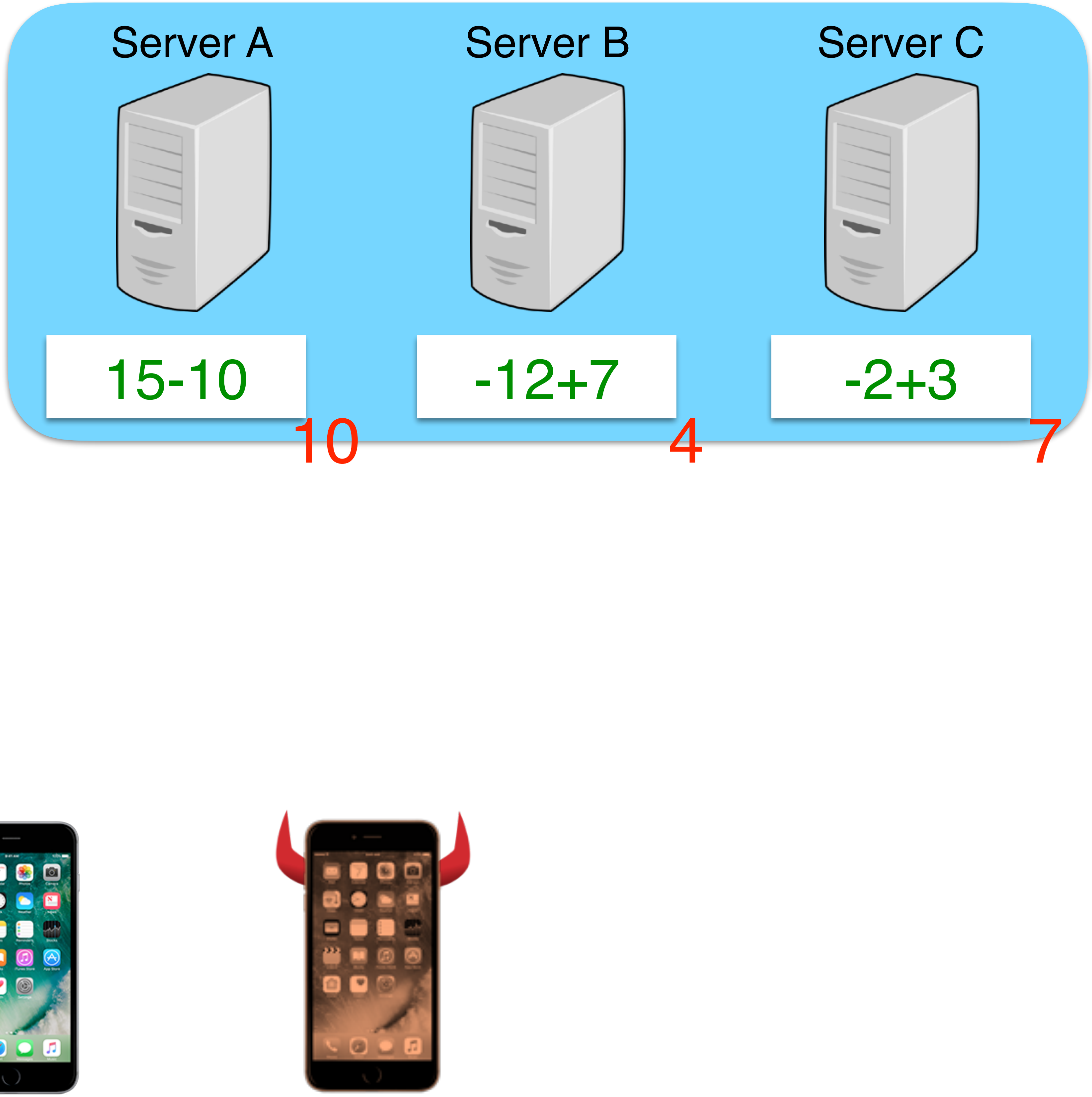
Private sums: A “straw-man” scheme



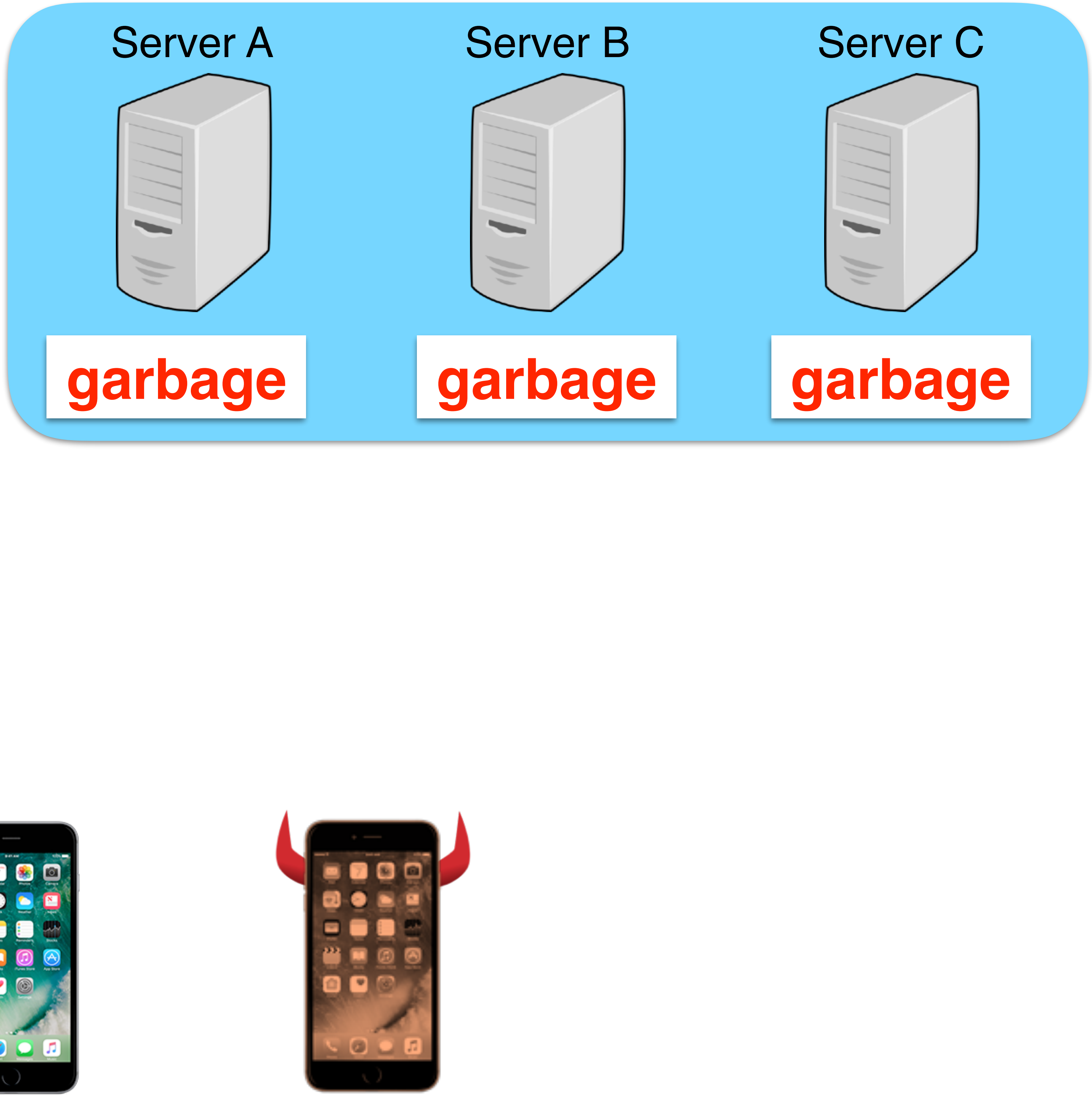
An evil client needn’t follow the rules!

$$10 + 4 + 7 = \mathbf{21}$$

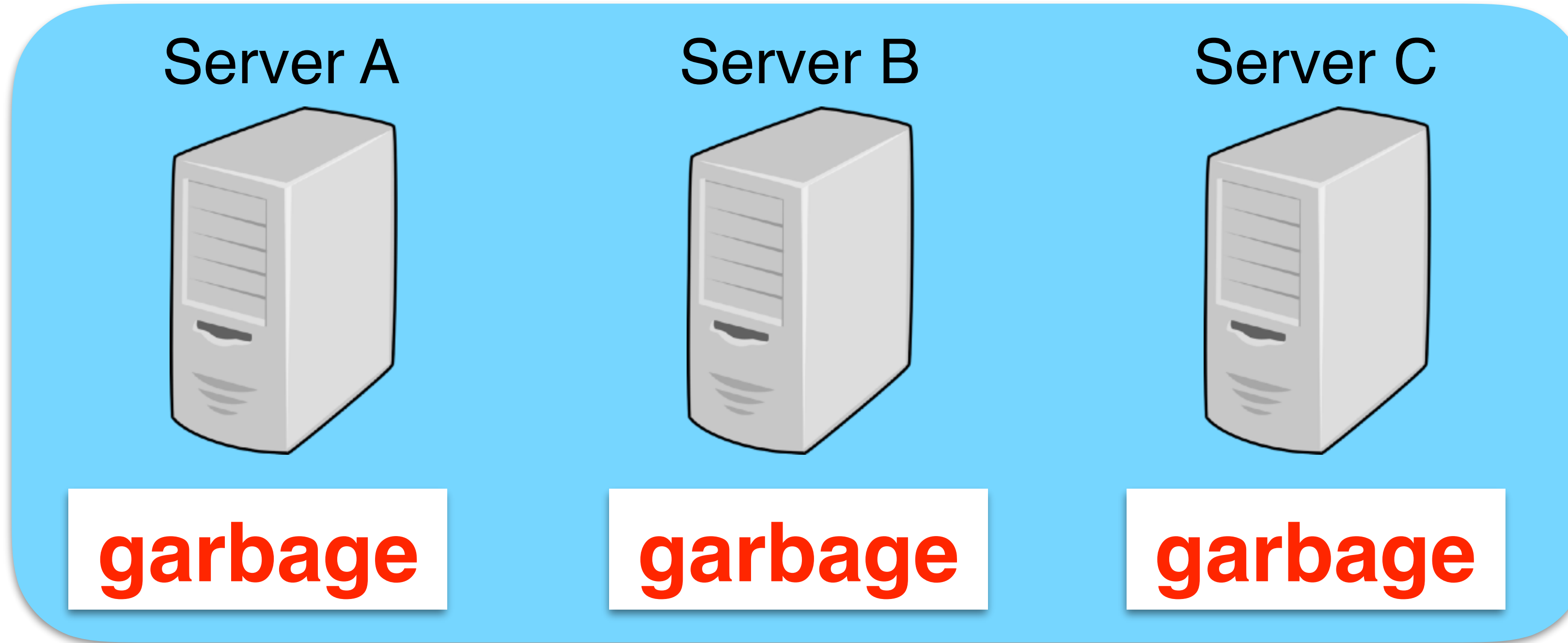
Private sums: A “straw-man” scheme



Private sums: A “straw-man” scheme



Private sums: A “straw-man” scheme



A single bad client **can undetectably corrupt** the sum

Users have **incentives** to cheat

Typical defenses (NIZKs) are costly

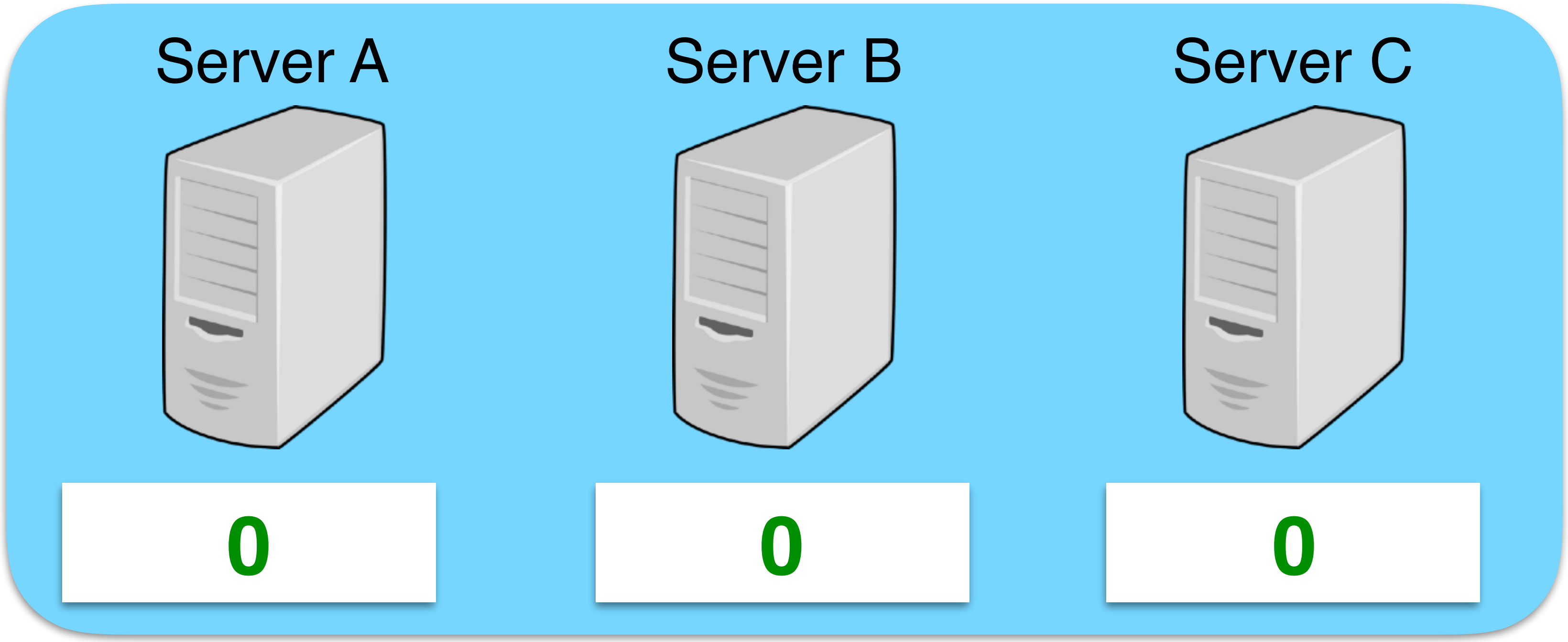
Outline

- Background: The private aggregation problem
- **A straw-man solution for private sums**
- Providing robustness with SNIPs
- Evaluation
- Discussion: Real-world considerations

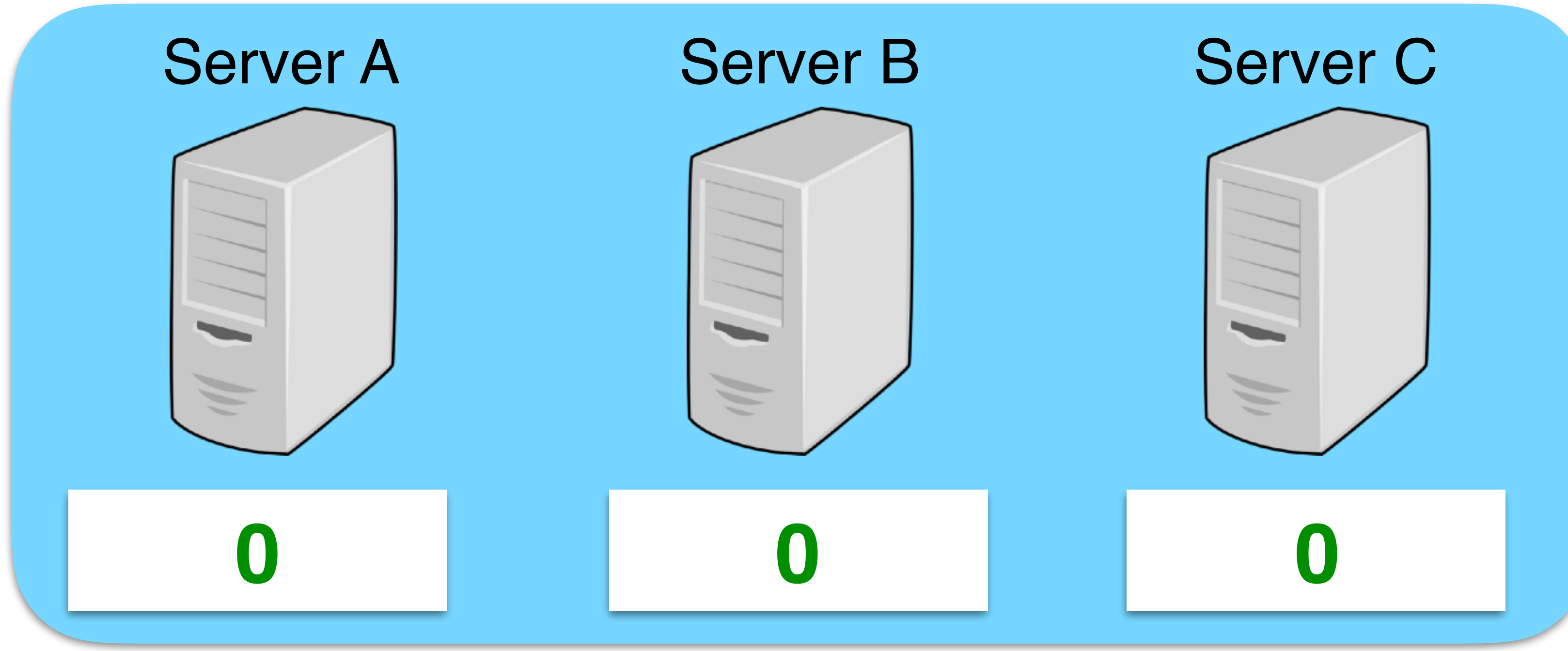
Outline

- Background: The private aggregation problem
- A straw-man solution for private sums
- **Providing robustness with SNIPs**
- Evaluation
- Discussion: Real-world considerations

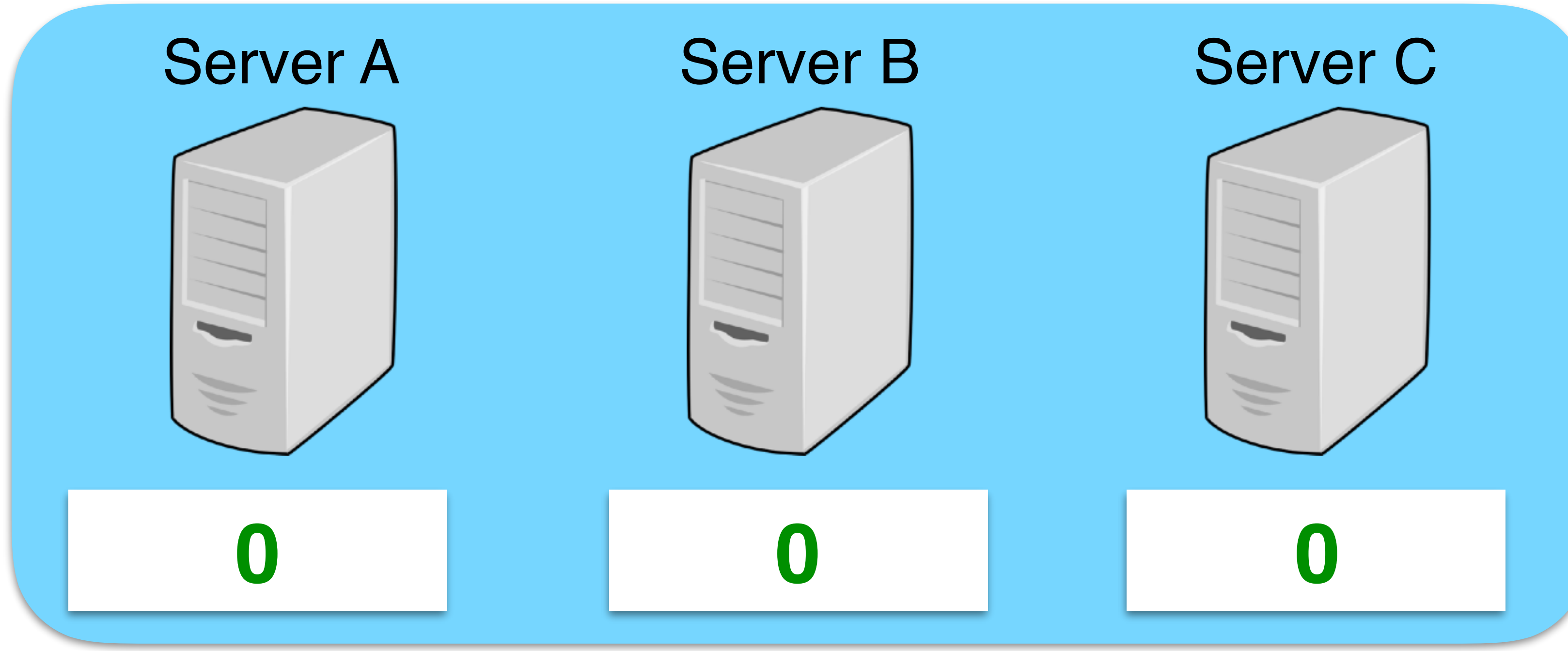
Contribution 1
Secret-shared non-interactive proofs (SNIPs)



Contribution 1
Secret-shared non-interactive proofs (SNIPs)



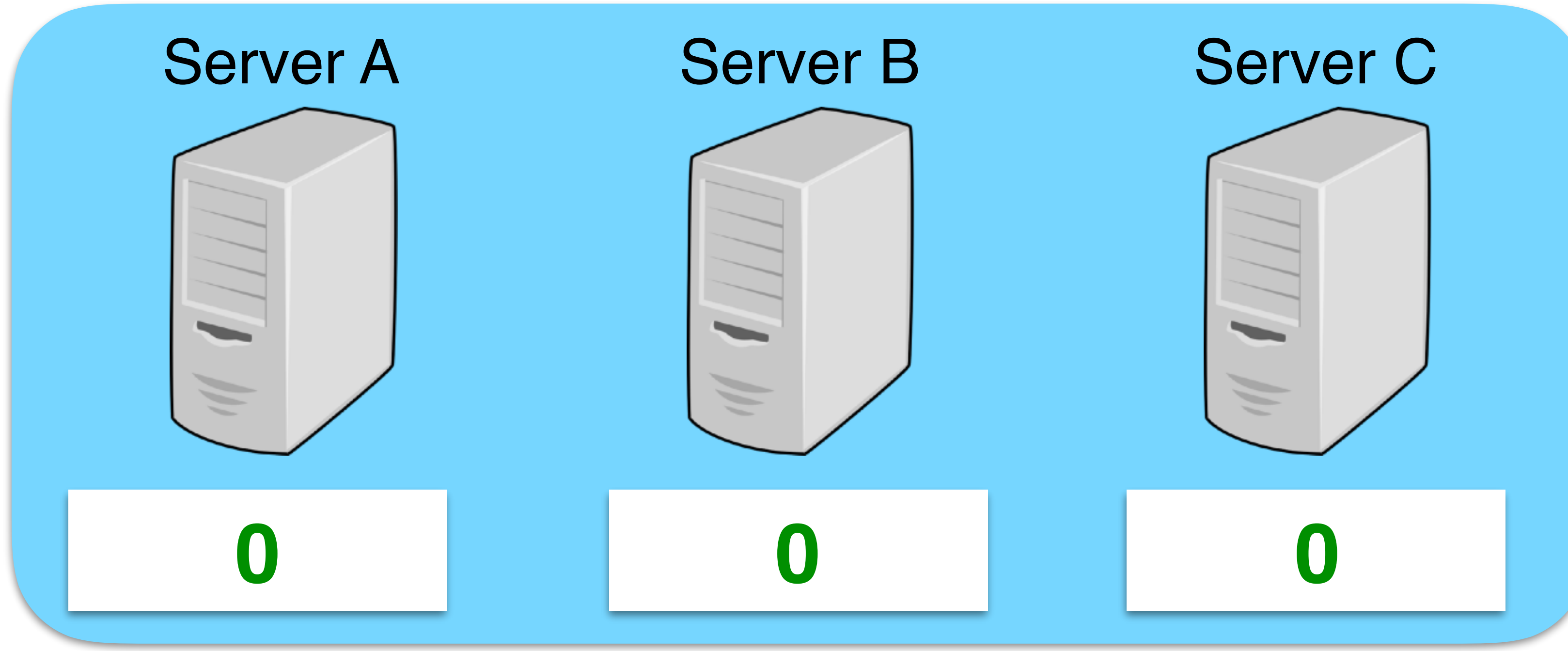
Contribution 1
Secret-shared non-interactive proofs (SNIPs)



$$15 + (-12) + (-2) = 1$$

$x = 1$

Contribution 1
Secret-shared non-interactive proofs (SNIPs)

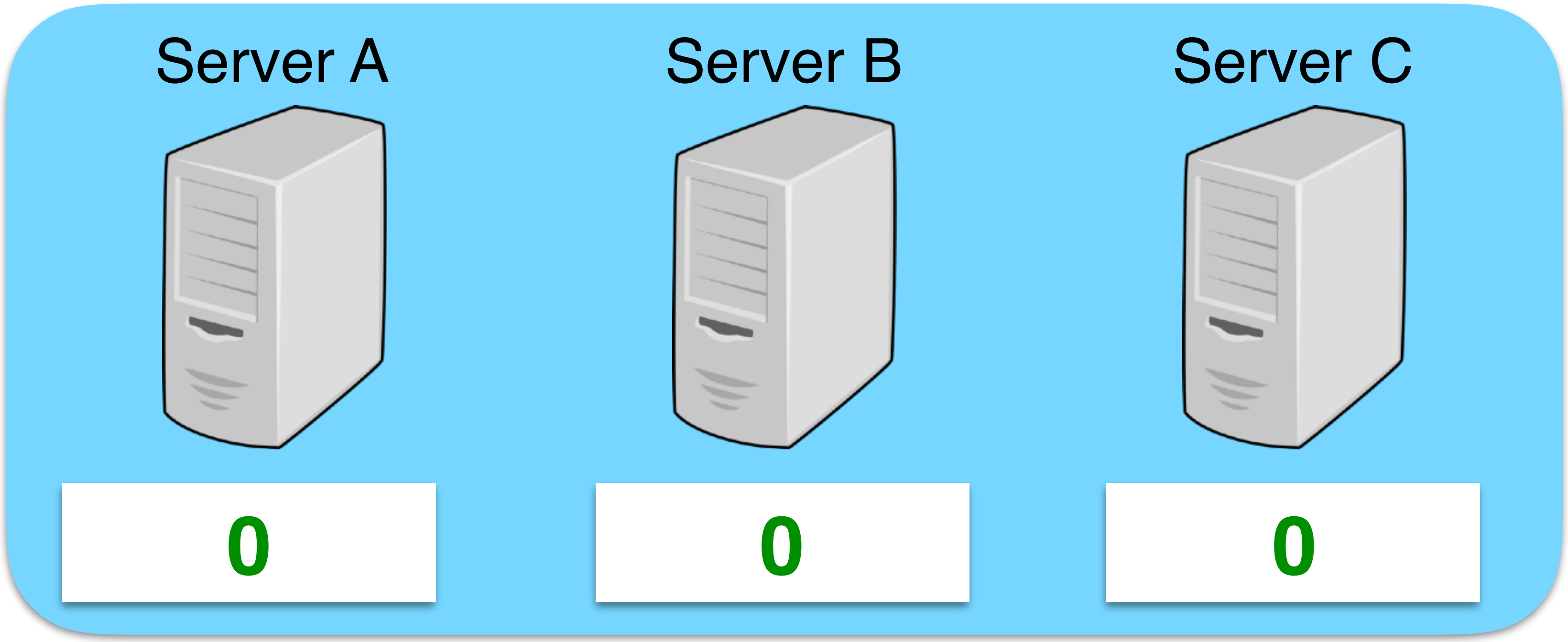


15 -12 -2

x = 1

Contribution 1

Secret-shared non-interactive proofs (SNIPs)

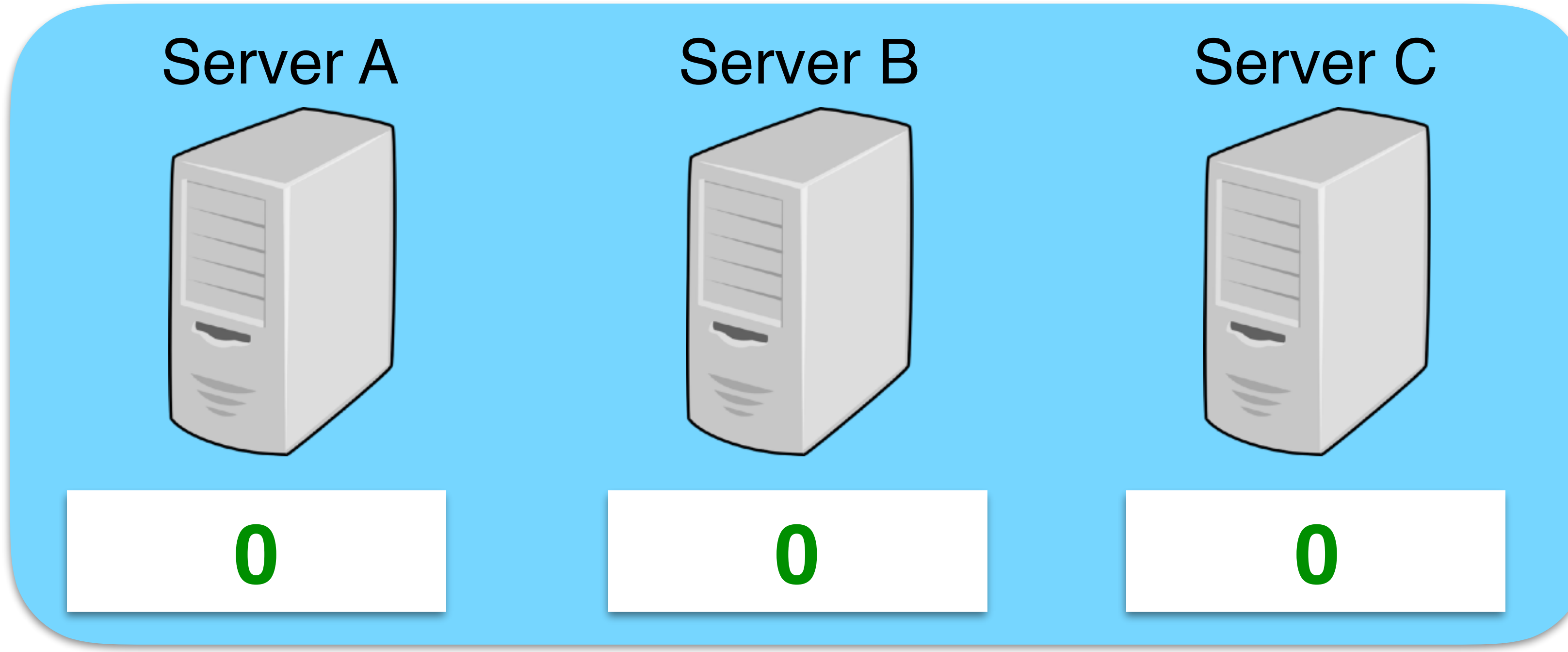


15 -12 -2

x = 1

Contribution 1

Secret-shared non-interactive proofs (SNIPs)



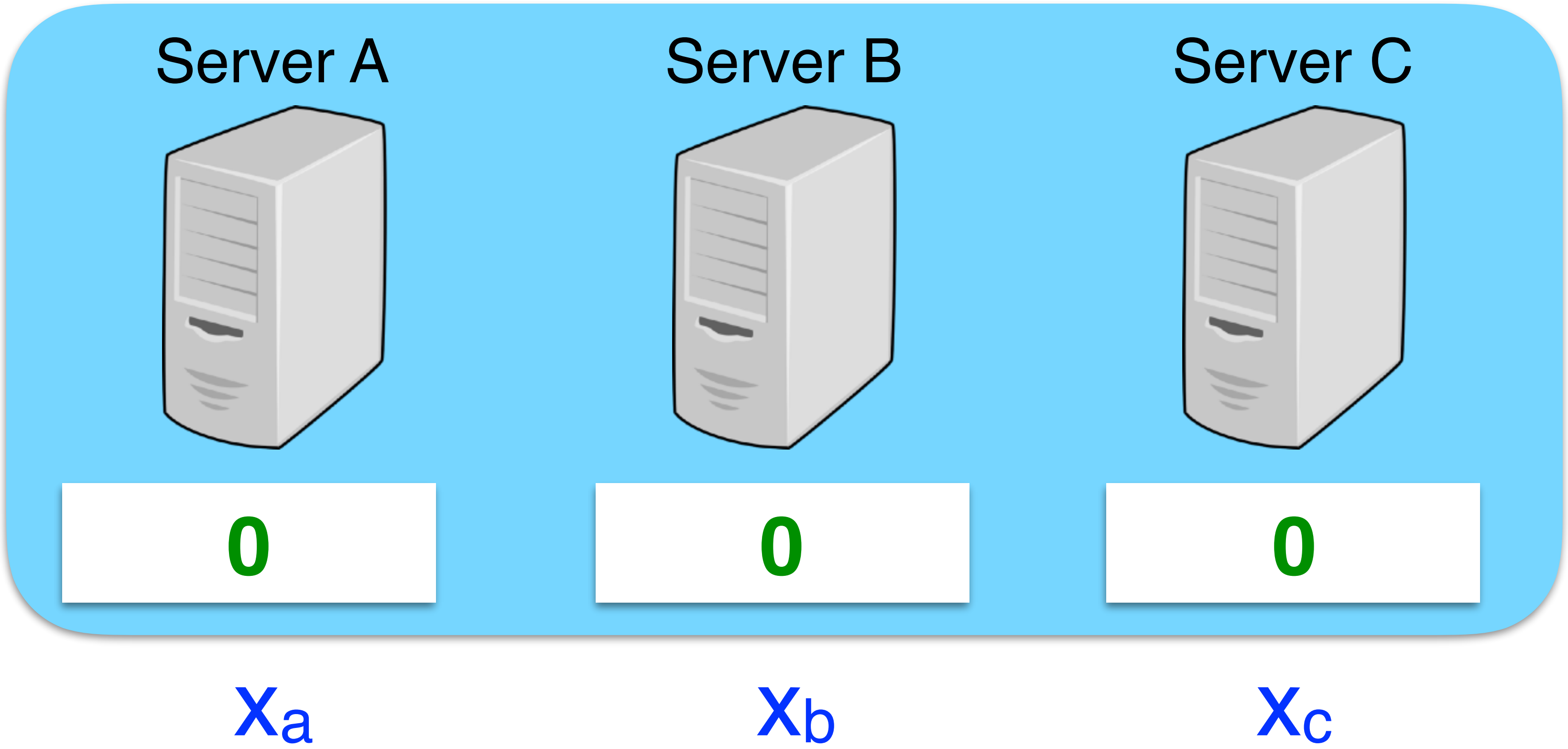
15 -12 -2

$x = 1$

In this example, the servers want to ensure that their shares sum to 0 or 1

…without learning x.

Contribution 1
Secret-shared non-interactive proofs (SNIPs)

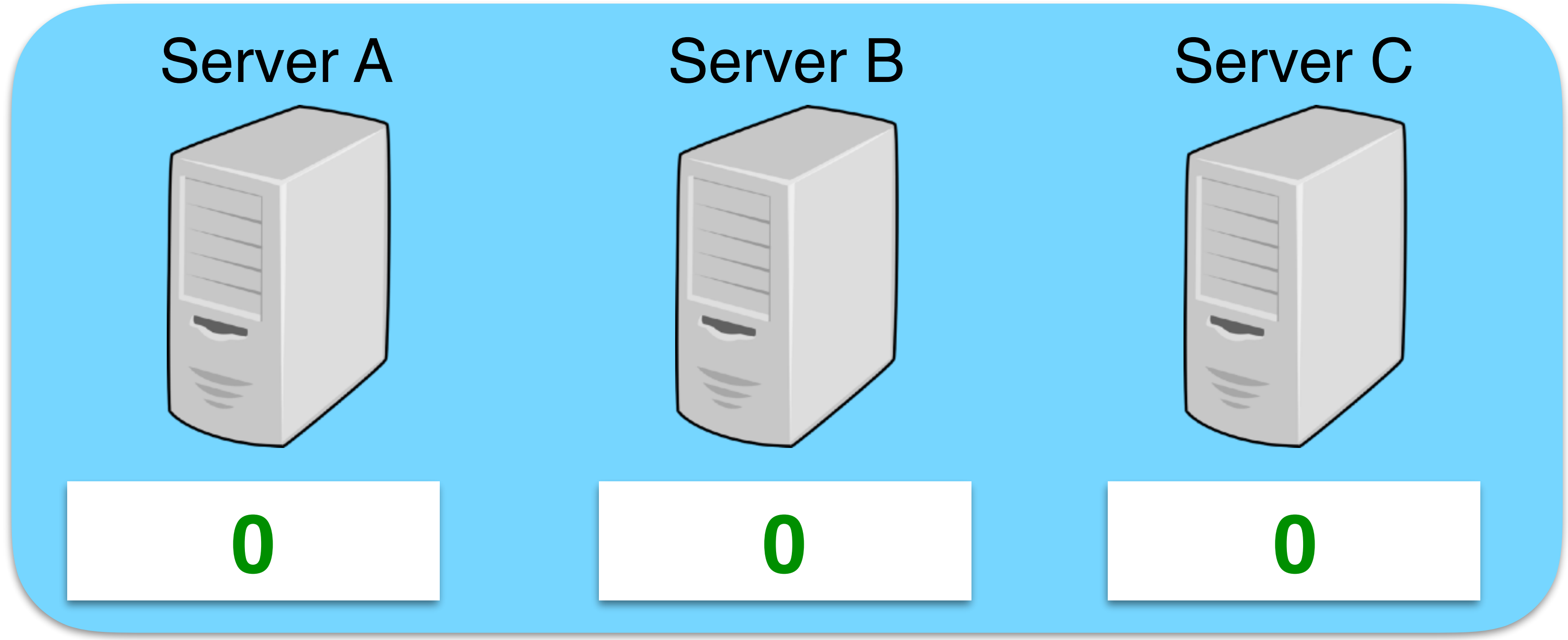


$x = 1$

More generally, servers

- hold shares of the client's private value x
- hold an <u>arbitrary</u> public predicate Valid(·)
  – expressed as an arithmetic circuit
- want to test if "Valid(x)" holds, without leaking x

Contribution 1
Secret-shared non-interactive proofs (SNIPs)



x_a x_b x_c

$x = 1$

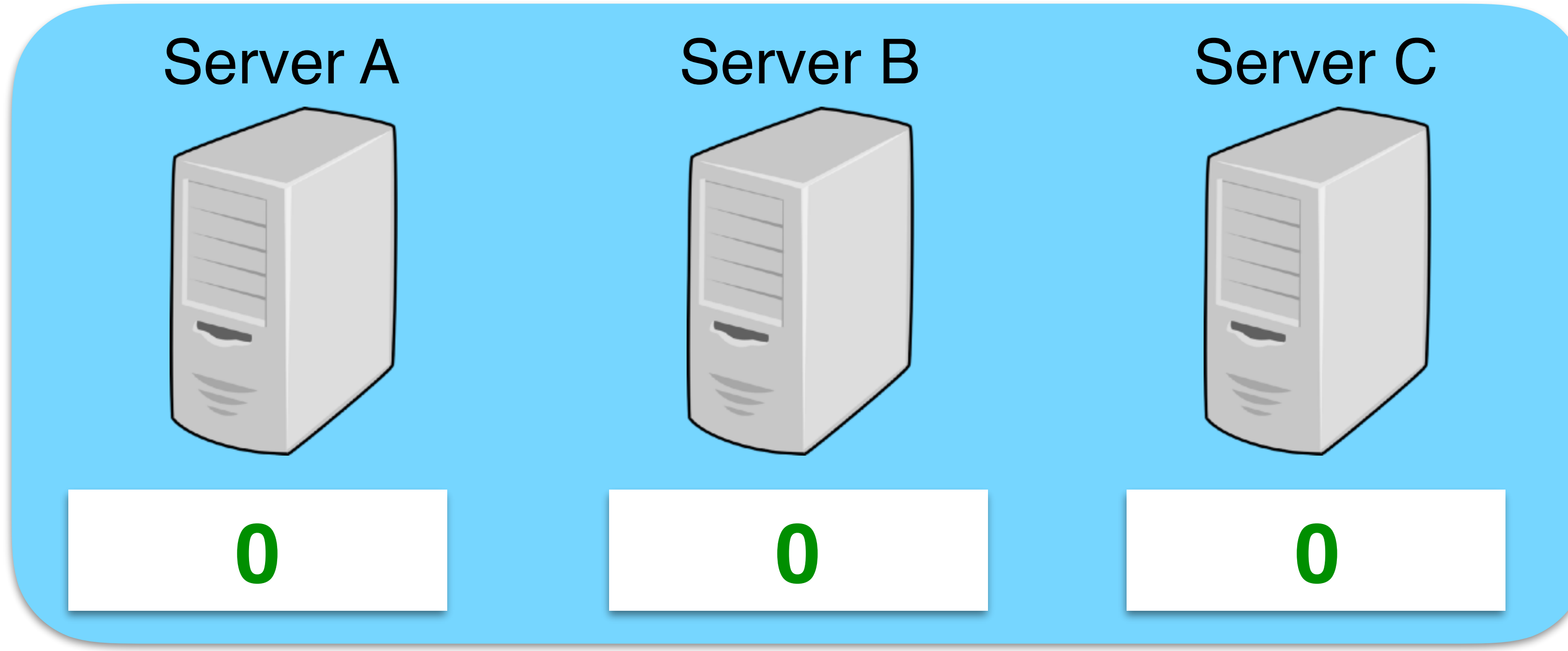
More generally, servers

- hold shares of the client’s private value x
- hold an <u>arbitrary</u> public predicate Valid(·)
  – expressed as an arithmetic circuit
- want to test if “Valid(x

For our running example:
Valid(x) = “$x \in \{0,1\}$”

Contribution 1

Secret-shared non-interactive proofs (SNIPs)



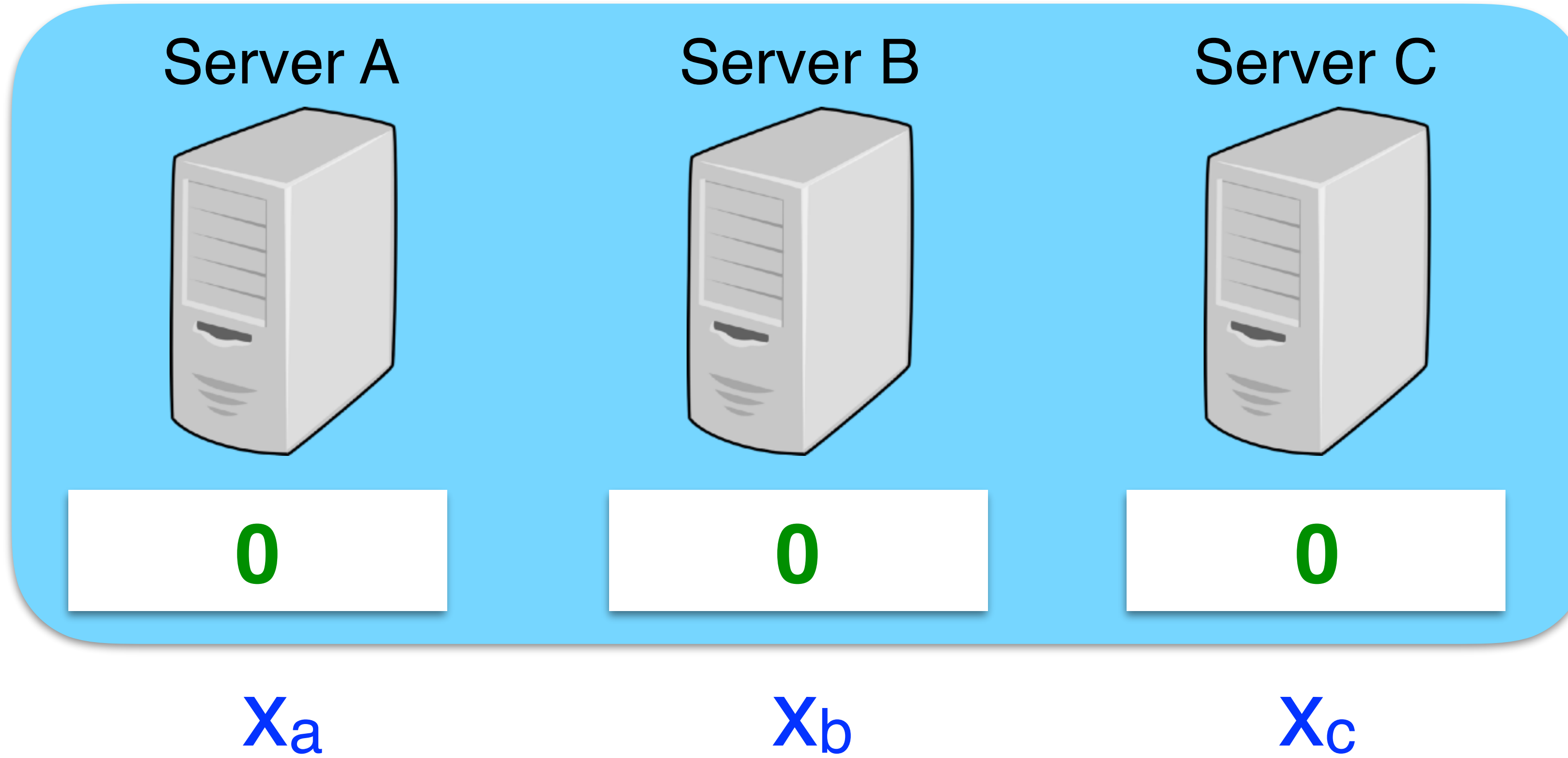
x_a x_b x_c

$x = 1$

More generally, servers

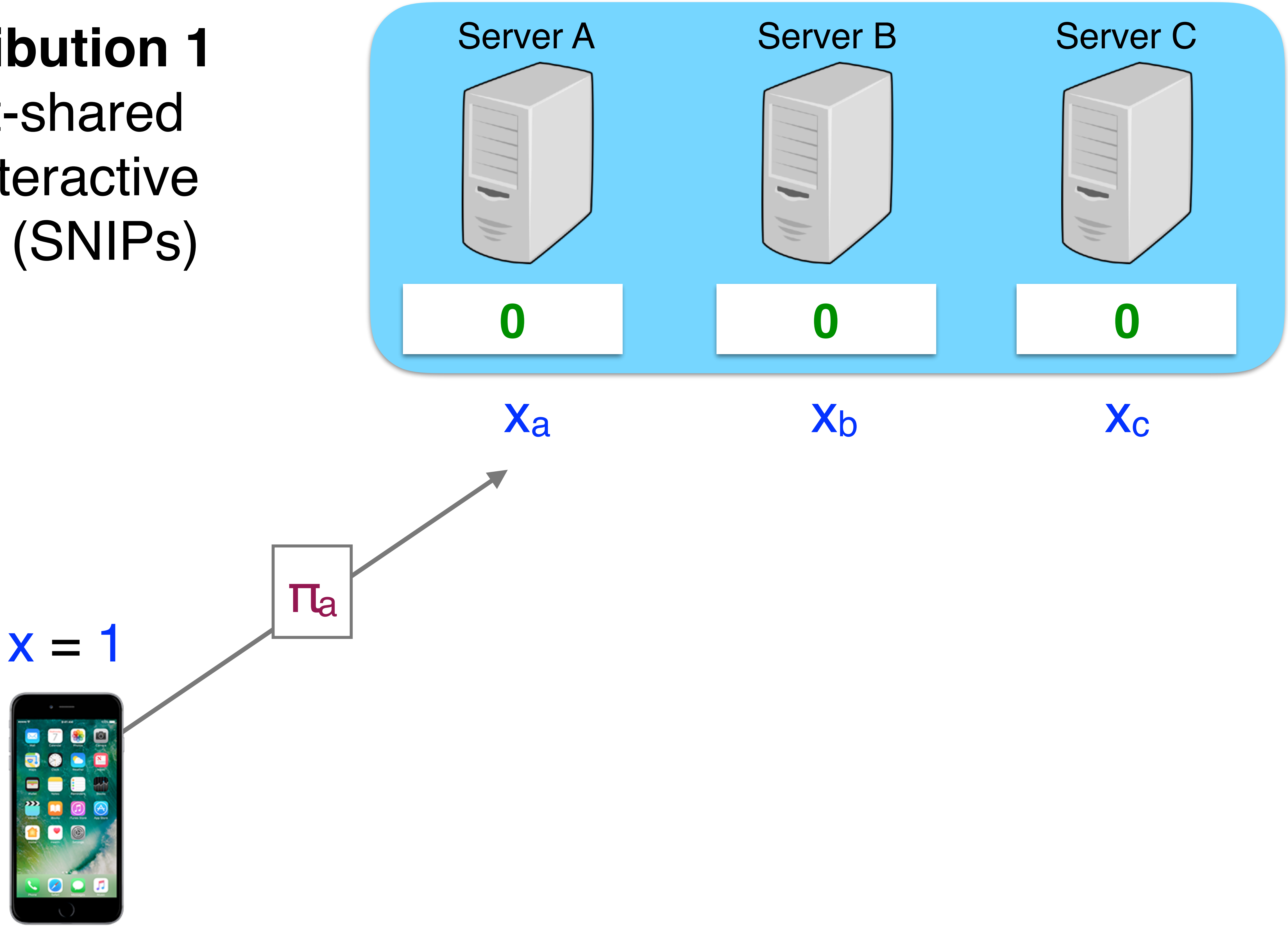
- hold shares of the client's private value x
- hold an <u>arbitrary</u> public predicate Valid(·)
  – expressed as an arithmetic circuit
- want to test if "Valid(x)" holds, without leaking x

Contribution 1
Secret-shared non-interactive proofs (SNIPs)

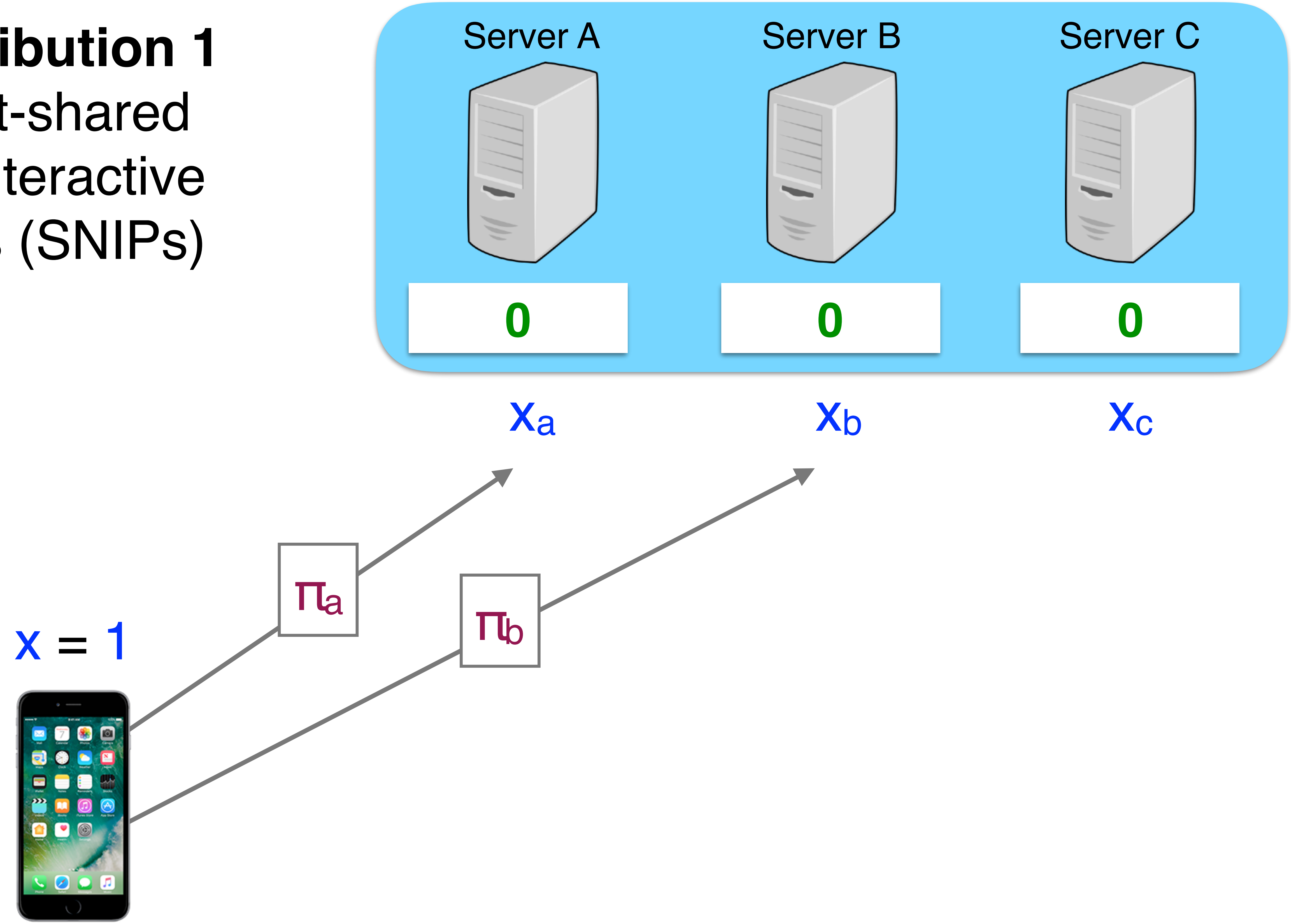


$x = 1$

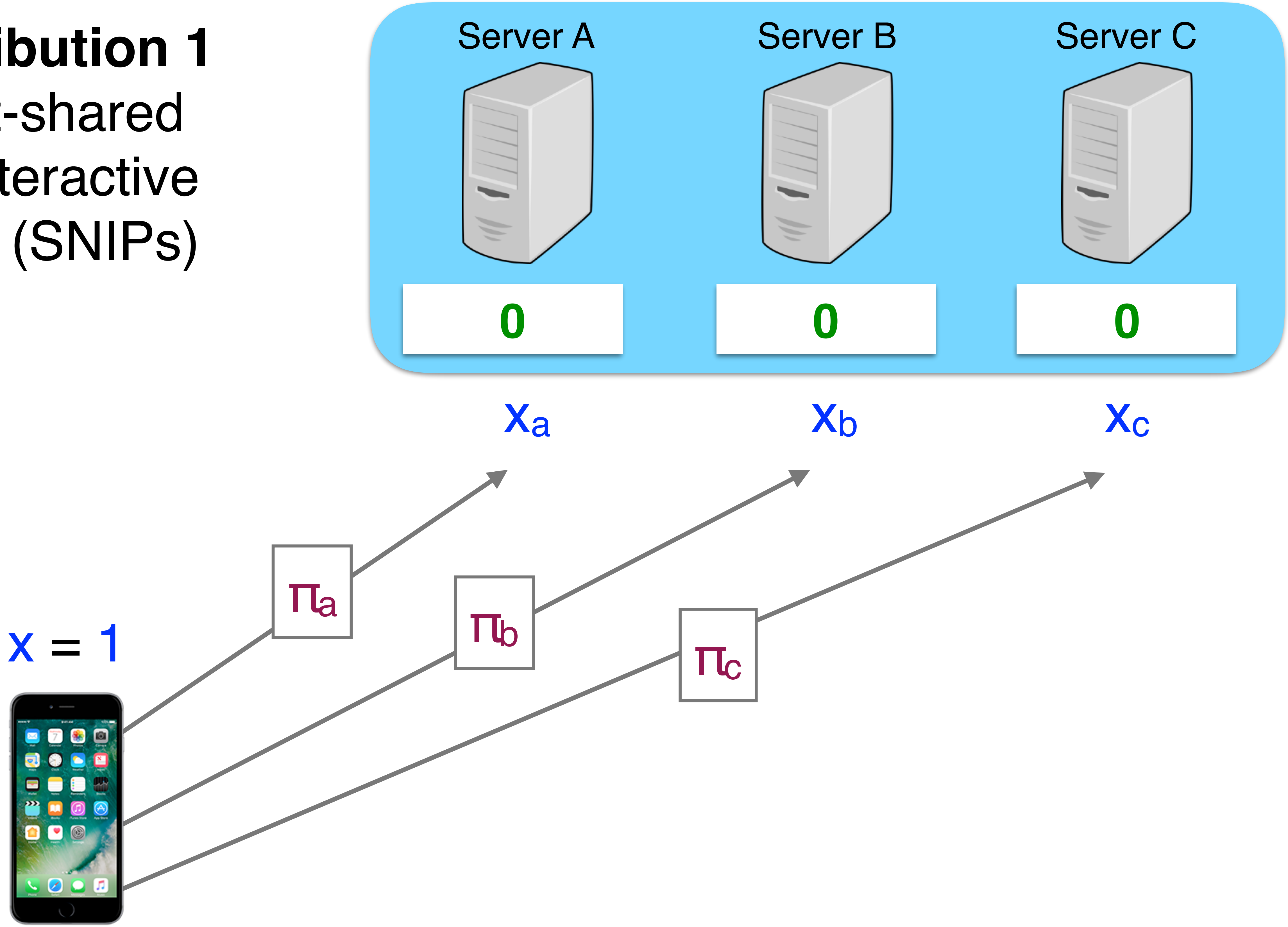
Contribution 1
Secret-shared non-interactive proofs (SNIPs)



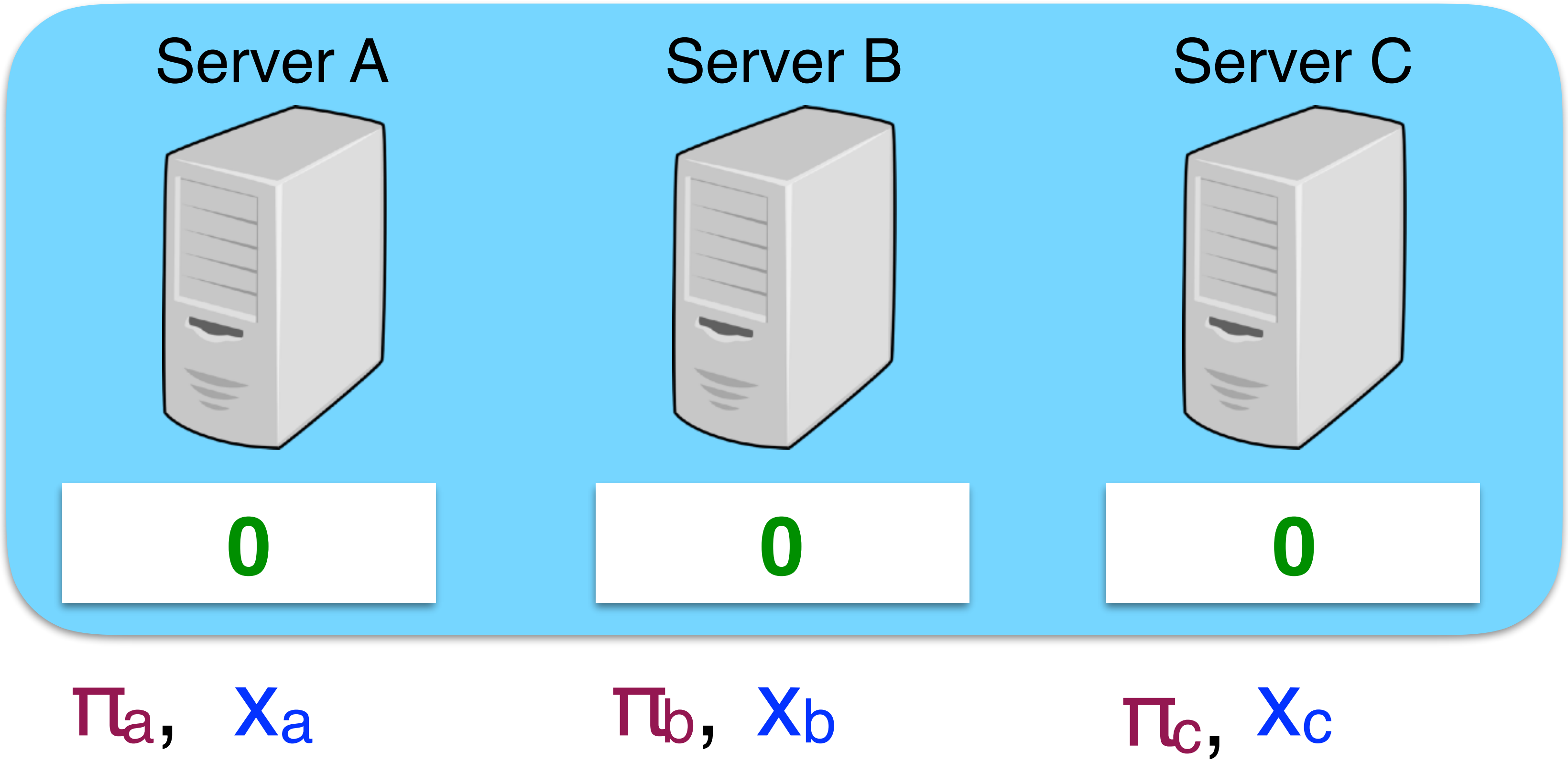
Contribution 1
Secret-shared
non-interactive
proofs (SNIPs)



Contribution 1
Secret-shared non-interactive proofs (SNIPs)

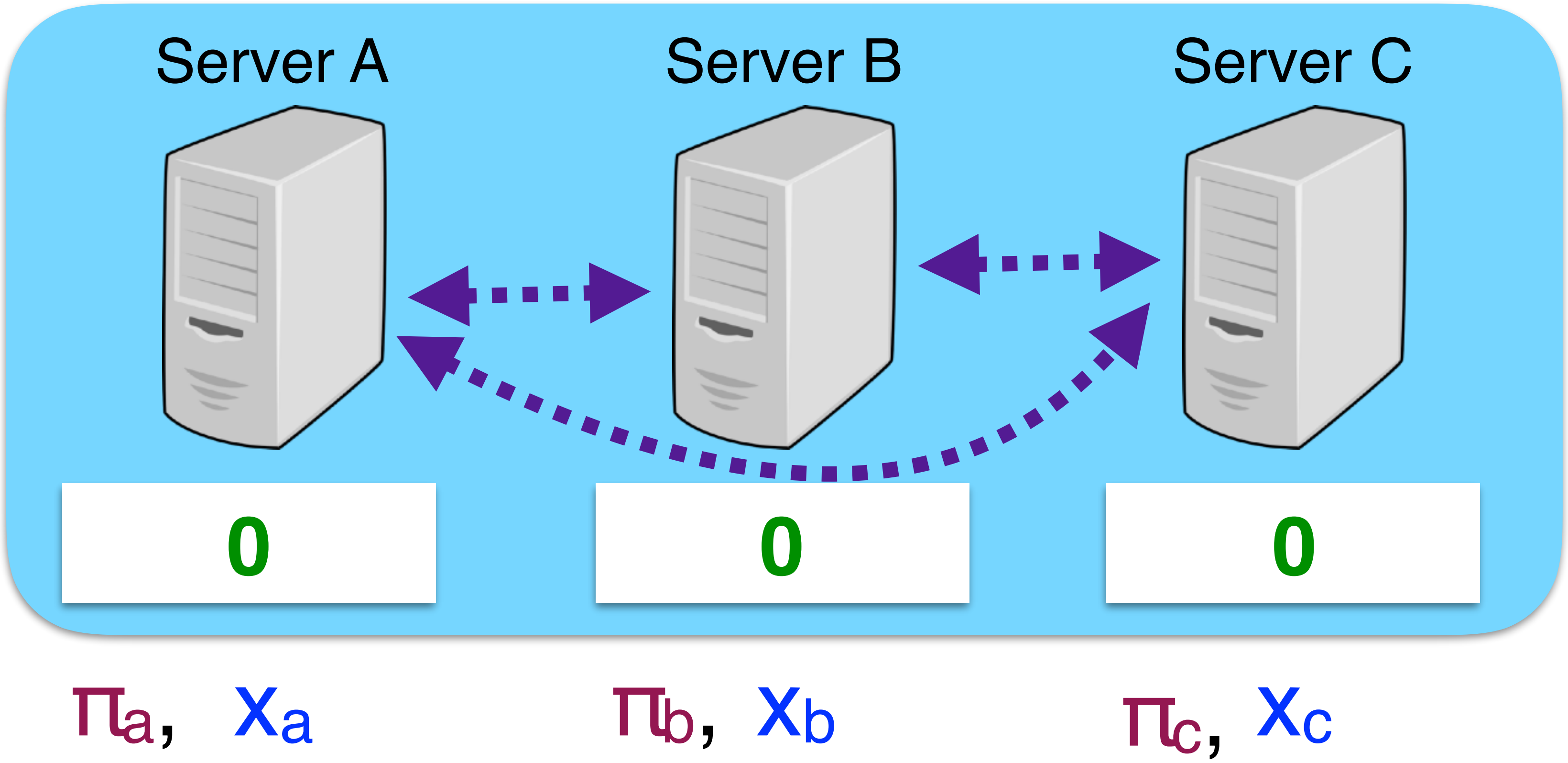


Contribution 1
Secret-shared non-interactive proofs (SNIPs)



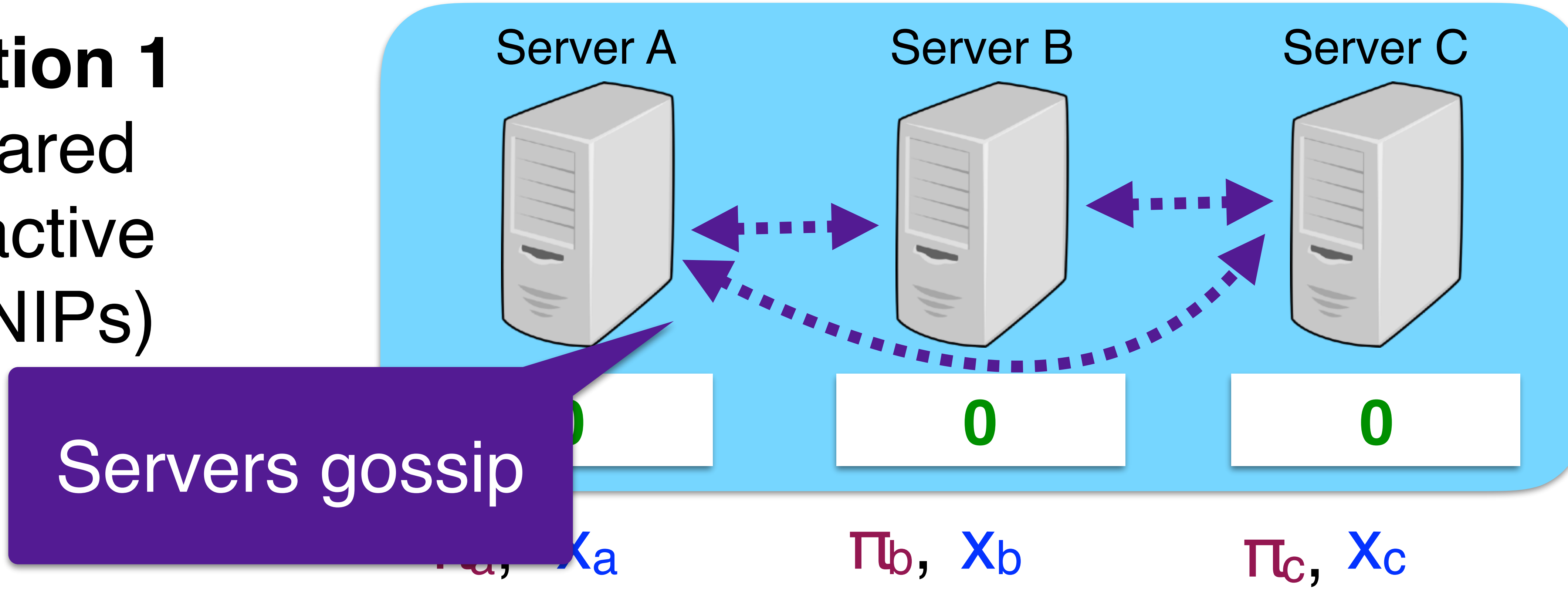
x = 1

Contribution 1
Secret-shared non-interactive proofs (SNIPs)



$x = 1$

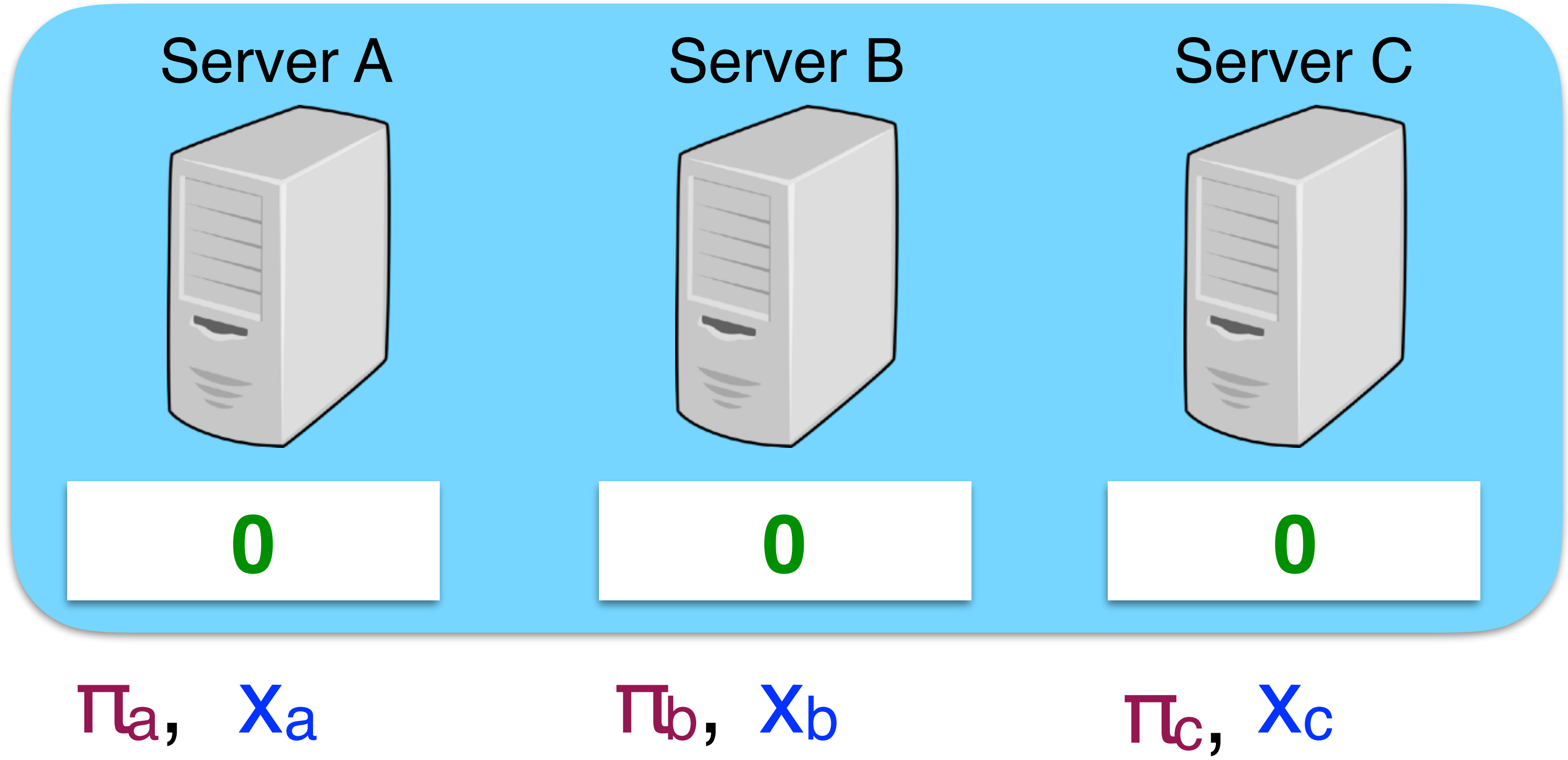
Contribution 1
Secret-shared non-interactive proofs (SNIPs)



$x = 1$

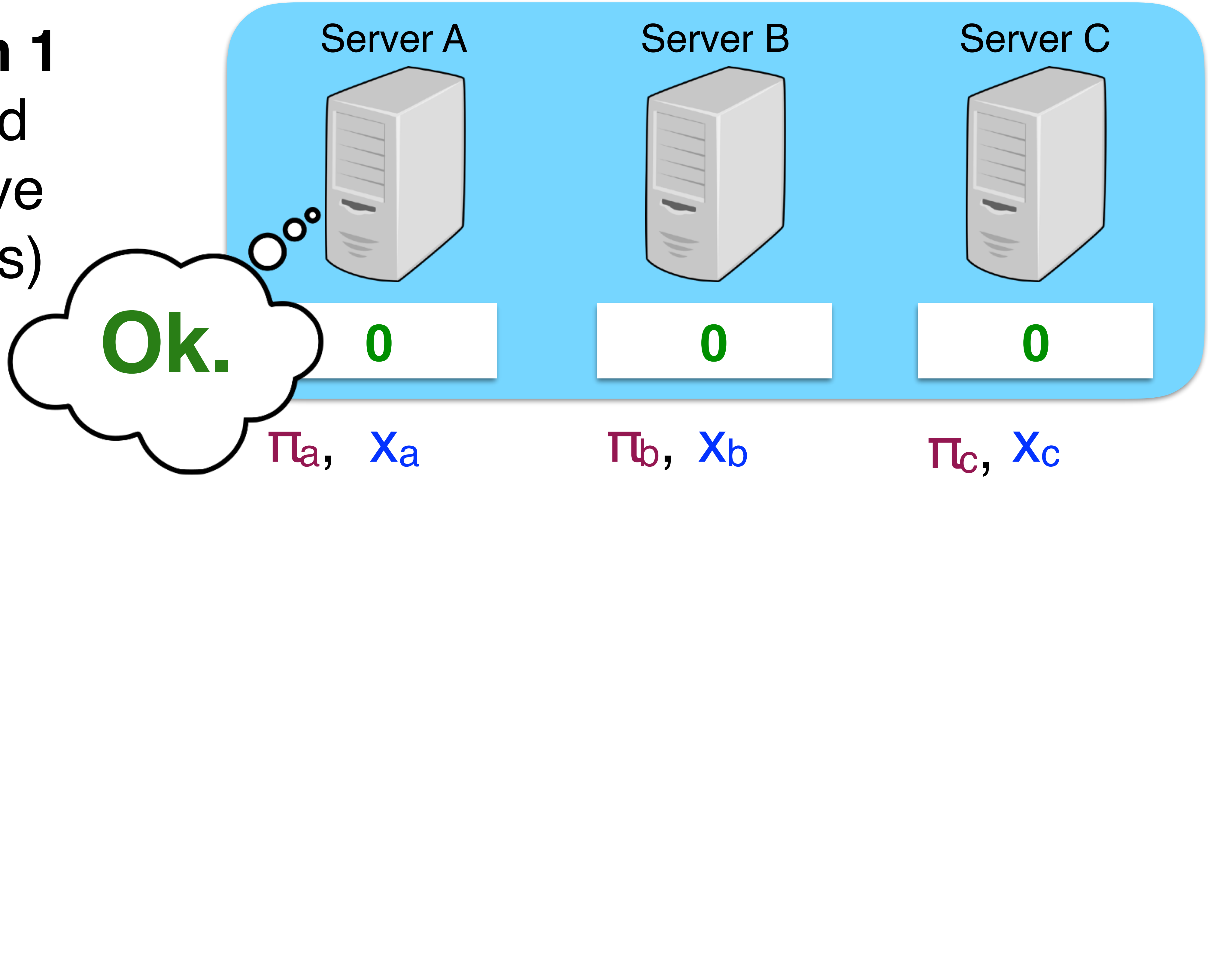
Contribution 1

Secret-shared non-interactive proofs (SNIPs)



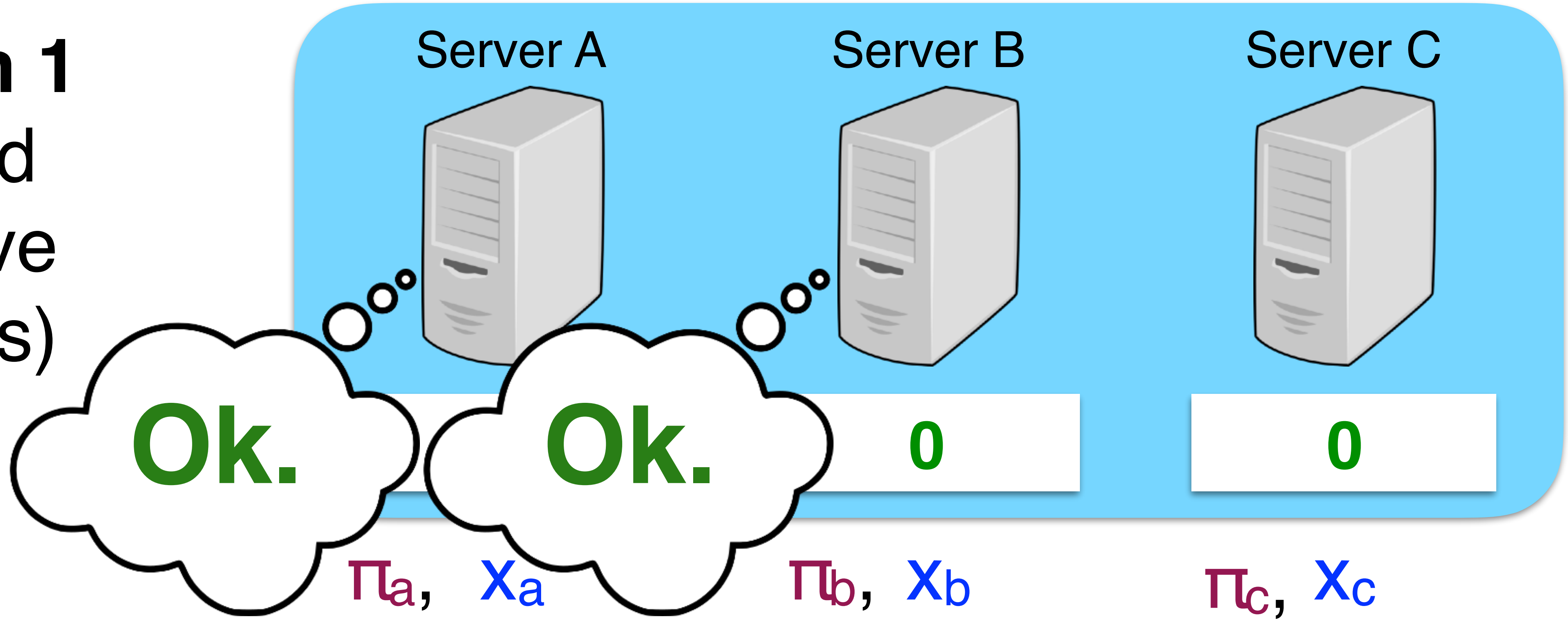
$x = 1$

Contribution 1
Secret-shared non-interactive proofs (SNIPs)



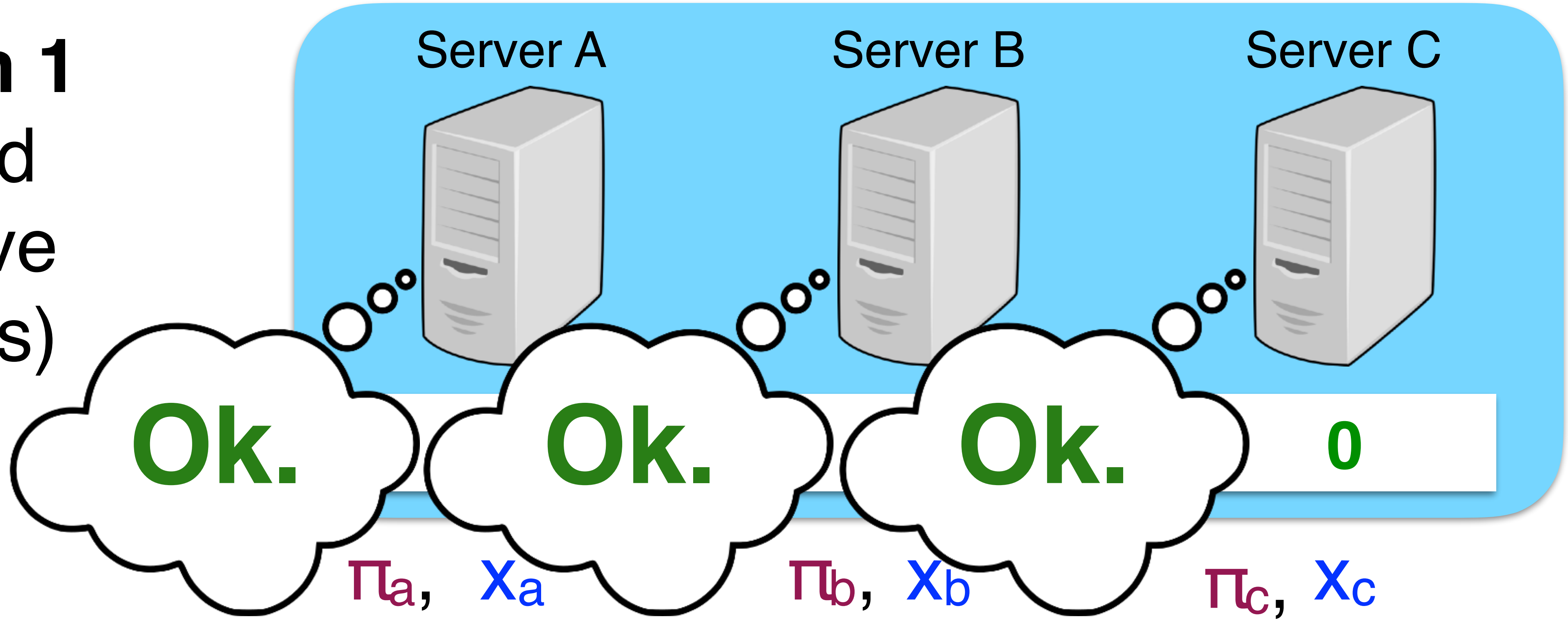
Contribution 1

Secret-shared non-interactive proofs (SNIPs)



$x = 1$

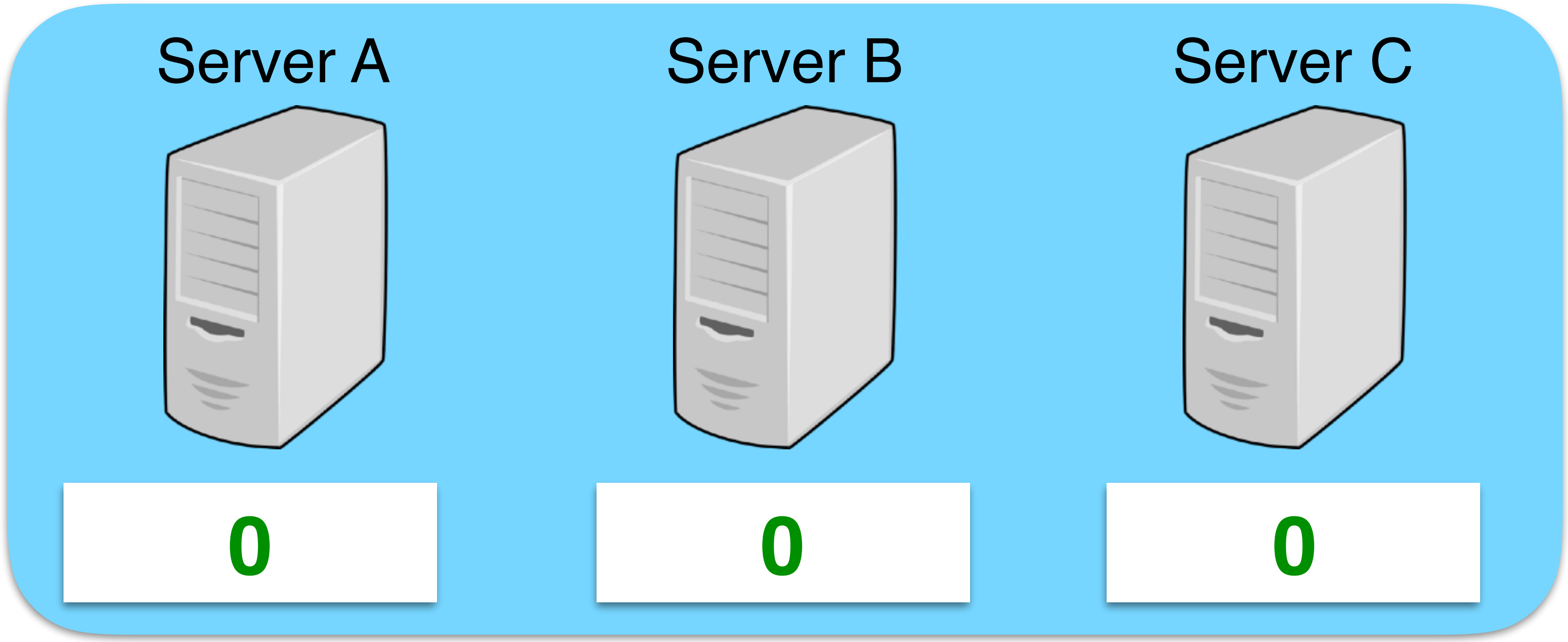
Contribution 1
Secret-shared non-interactive proofs (SNIPs)



$x = 1$

Contribution 1

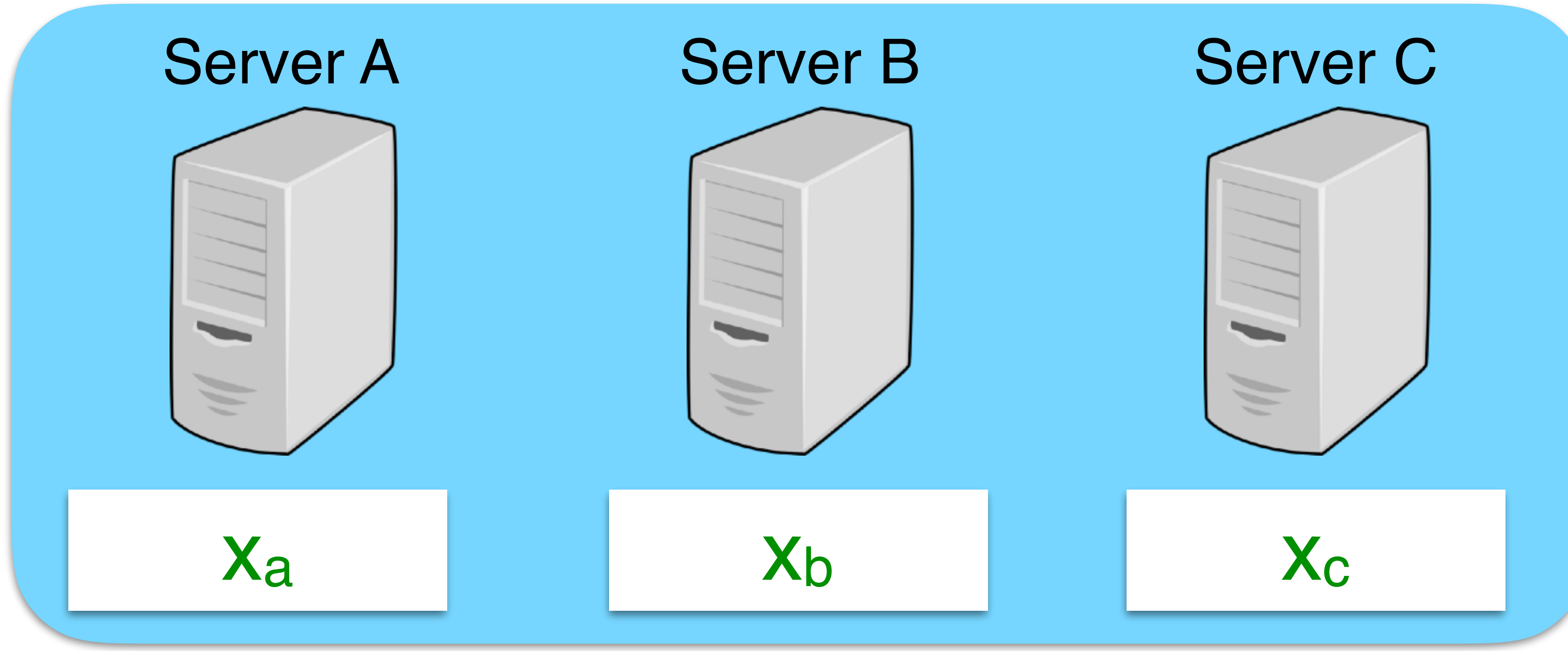
Secret-shared non-interactive proofs (SNIPs)



π_a, x_a π_b, x_b π_c, x_c

$x = 1$

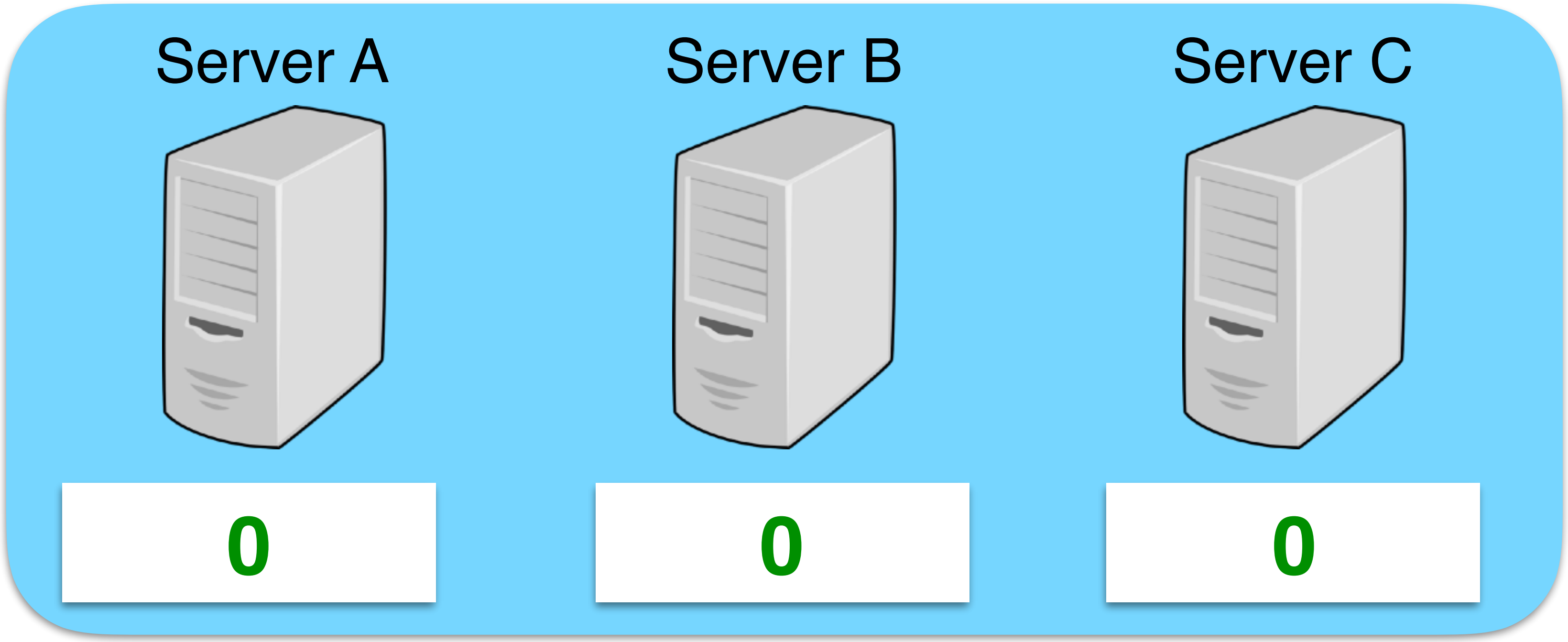
Contribution 1
Secret-shared non-interactive proofs (SNIPs)



$x = 1$

Contribution 1

Secret-shared non-interactive proofs (SNIPs)

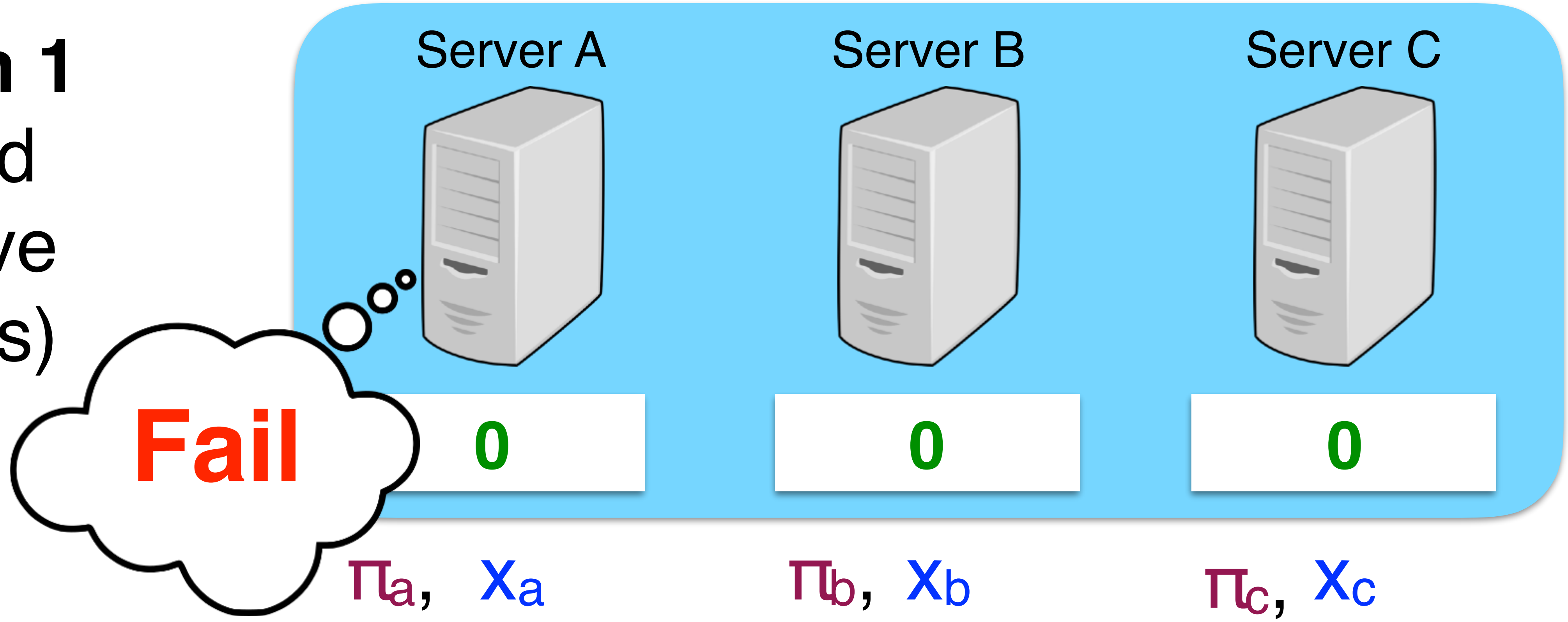


π_a, x_a π_b, x_b π_c, x_c

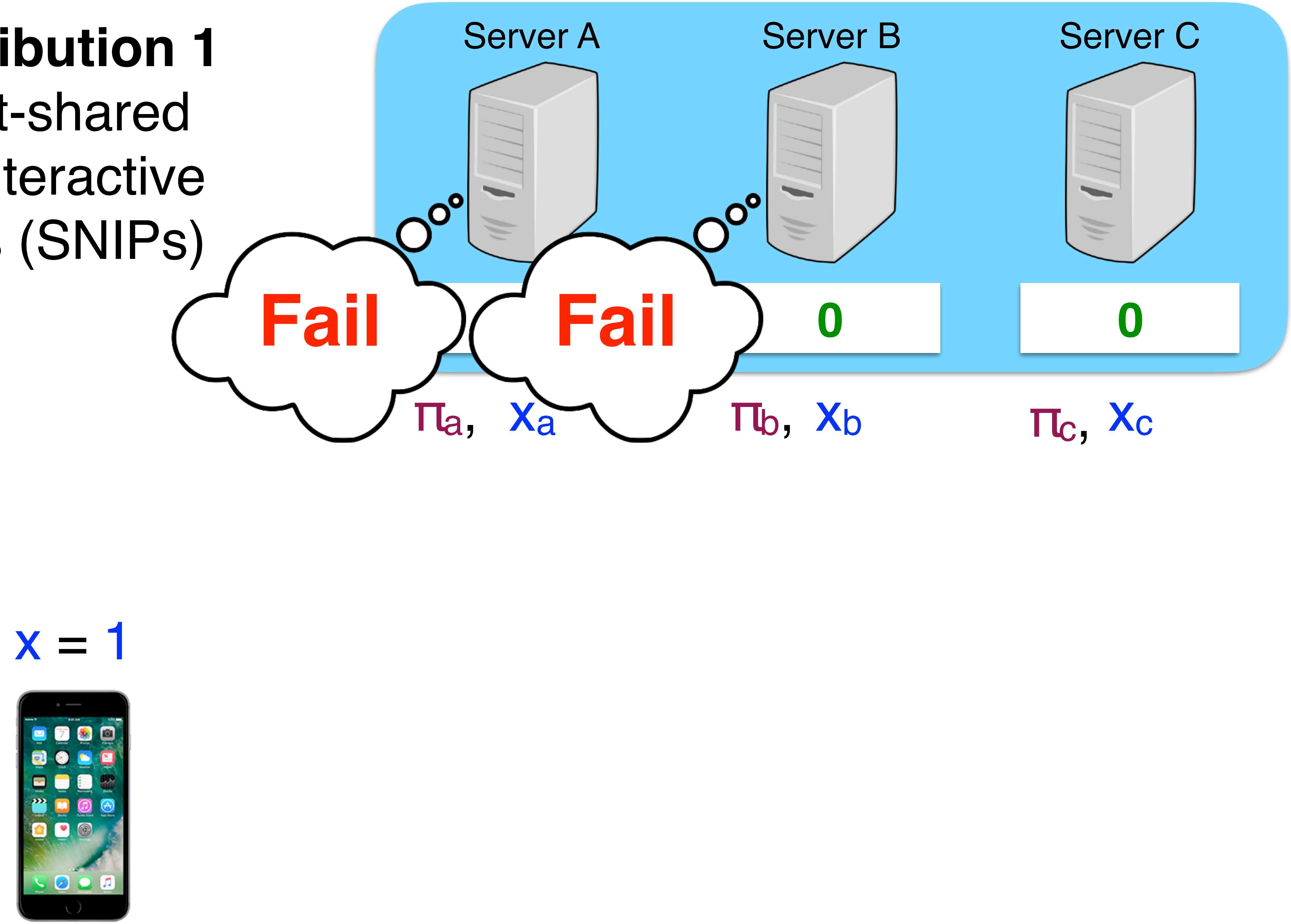
$x = 1$

Contribution 1

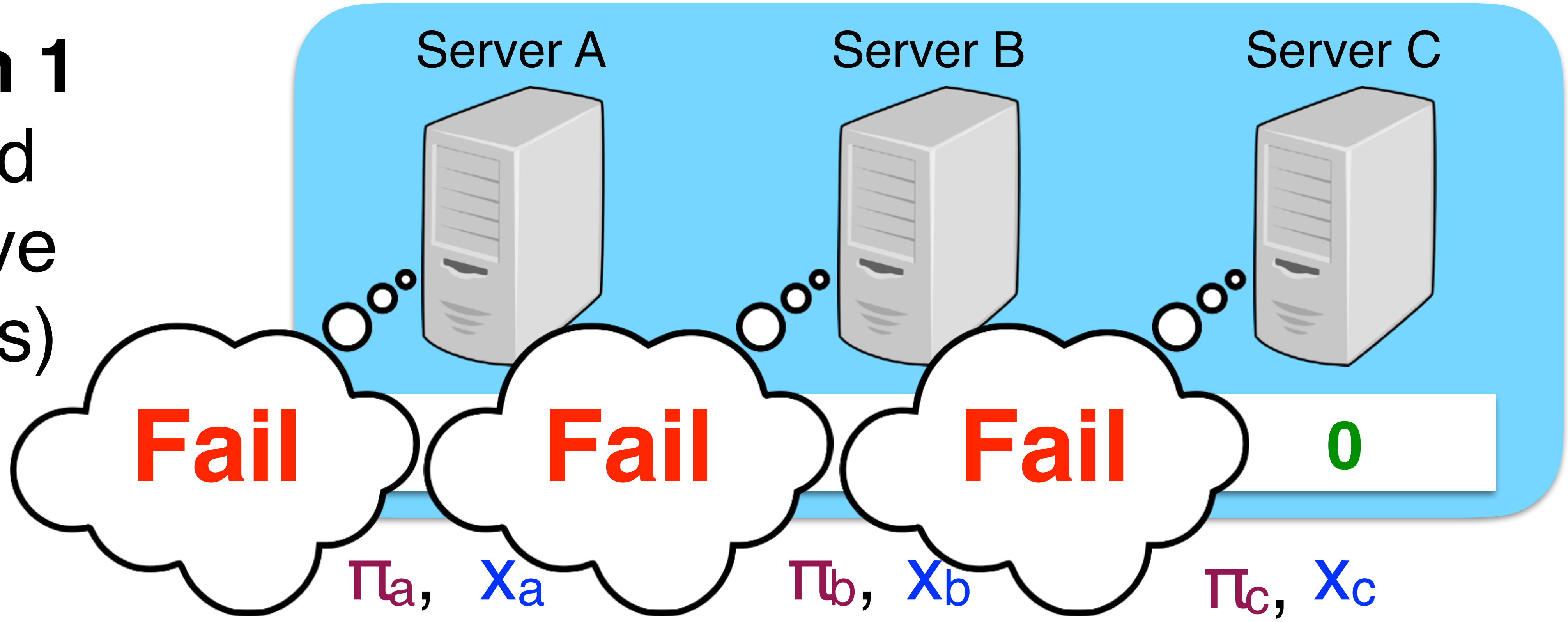
Secret-shared non-interactive proofs (SNIPs)



Contribution 1
Secret-shared non-interactive proofs (SNIPs)

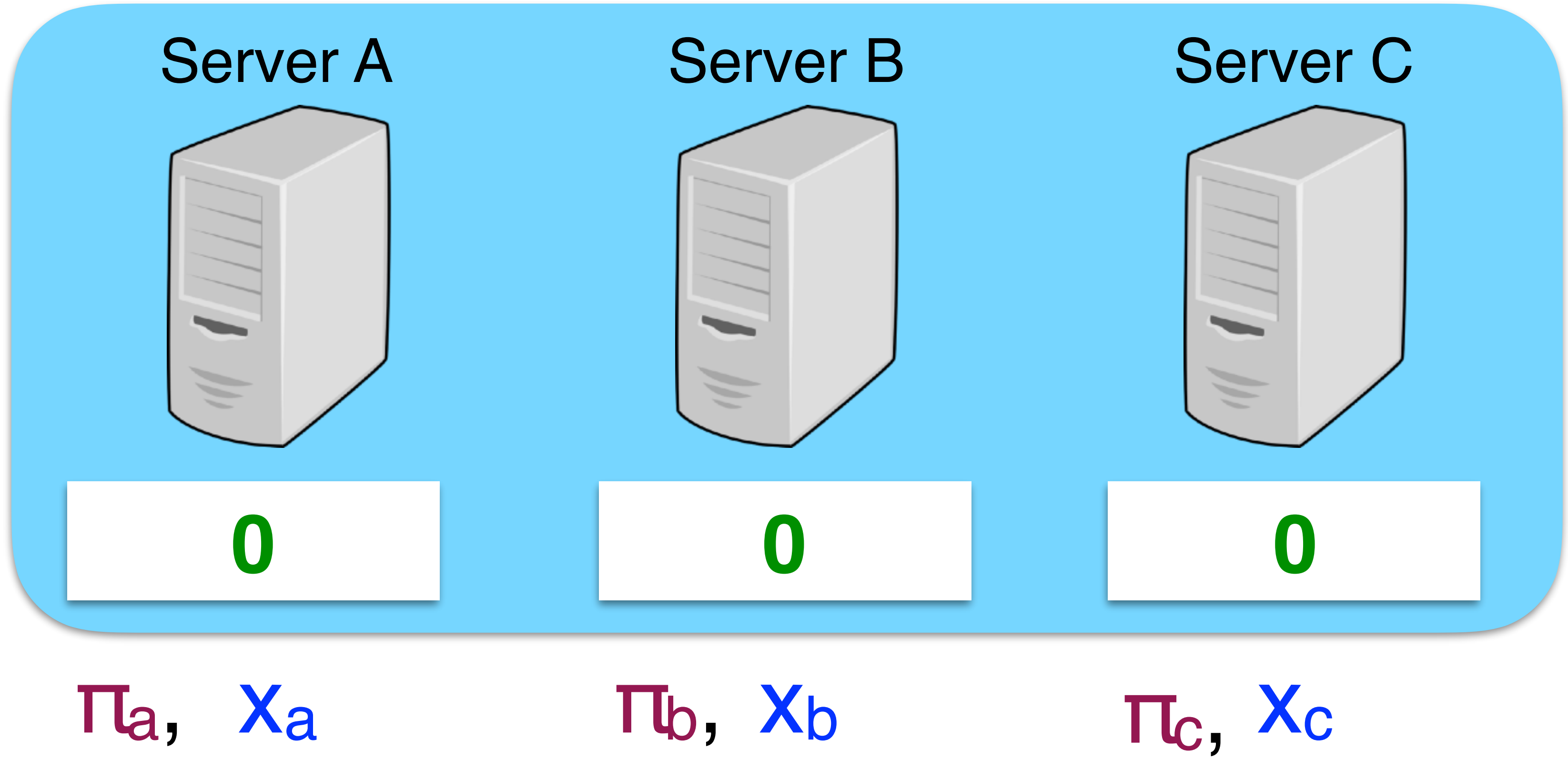


Contribution 1
Secret-shared non-interactive proofs (SNIPs)



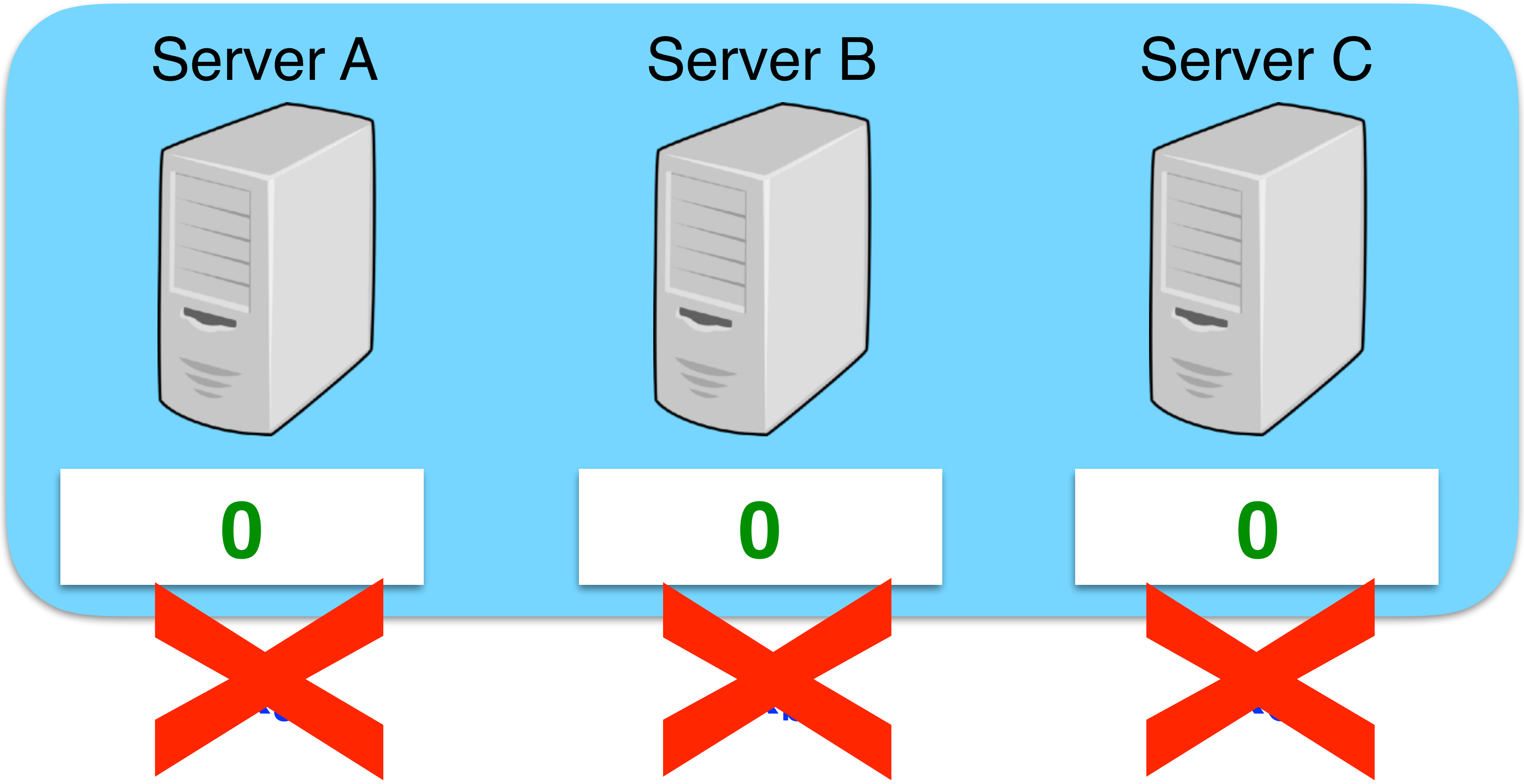
$x = 1$

Contribution 1
Secret-shared non-interactive proofs (SNIPs)



$x = 1$

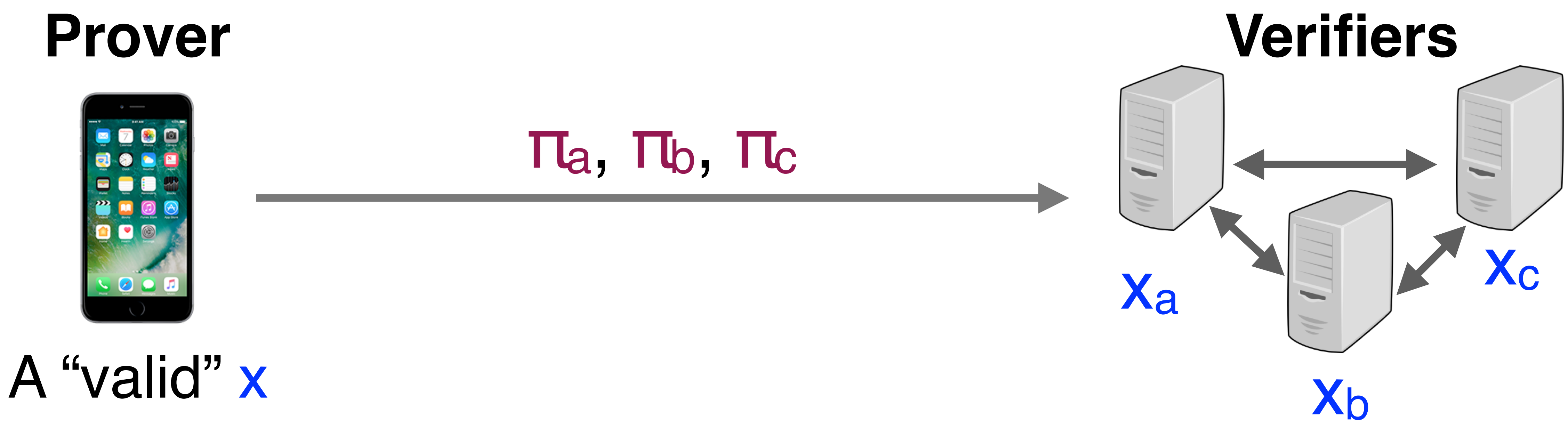
Contribution 1
Secret-shared non-interactive proofs (SNIPs)



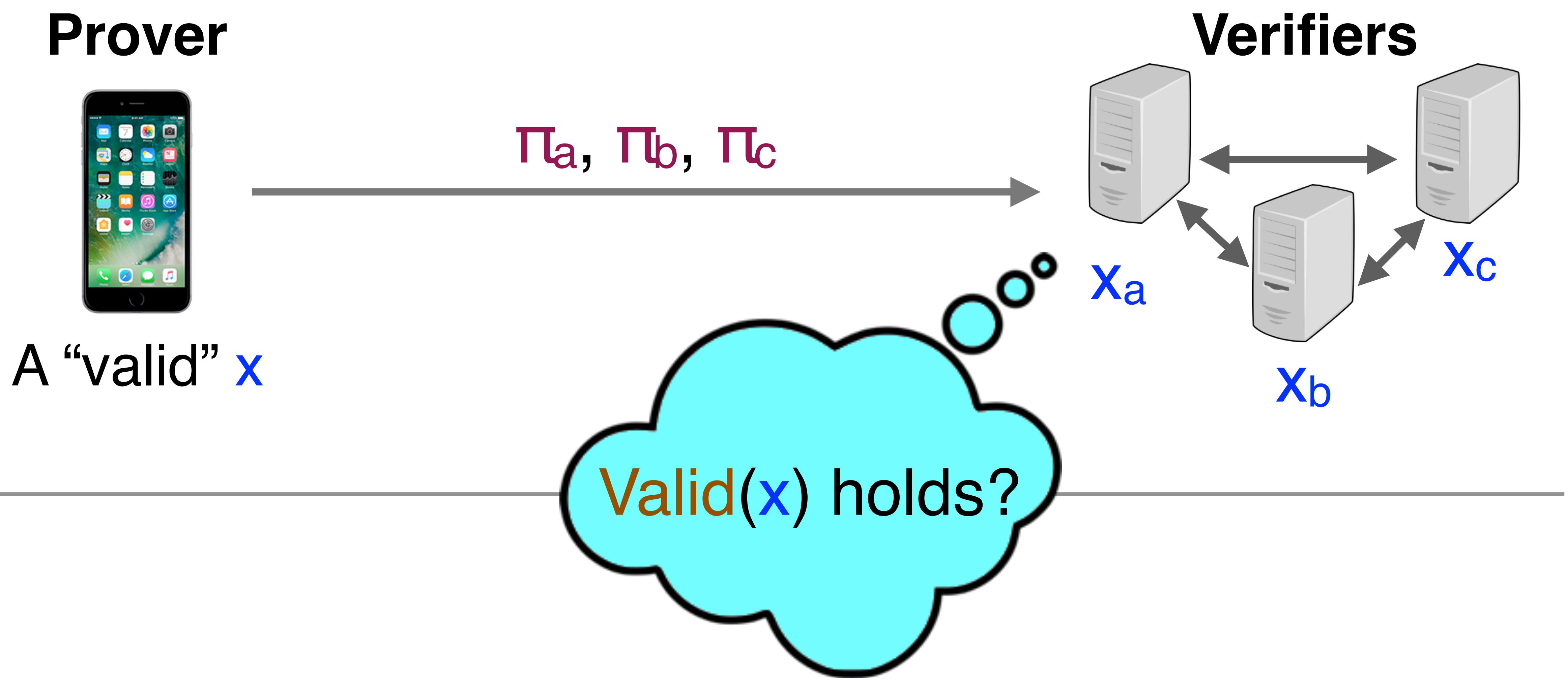
x = 1

- Prio servers *detect and reject malformed client submissions*
- In this example, each client can influence the aggregate statistic by +/- 1, at most

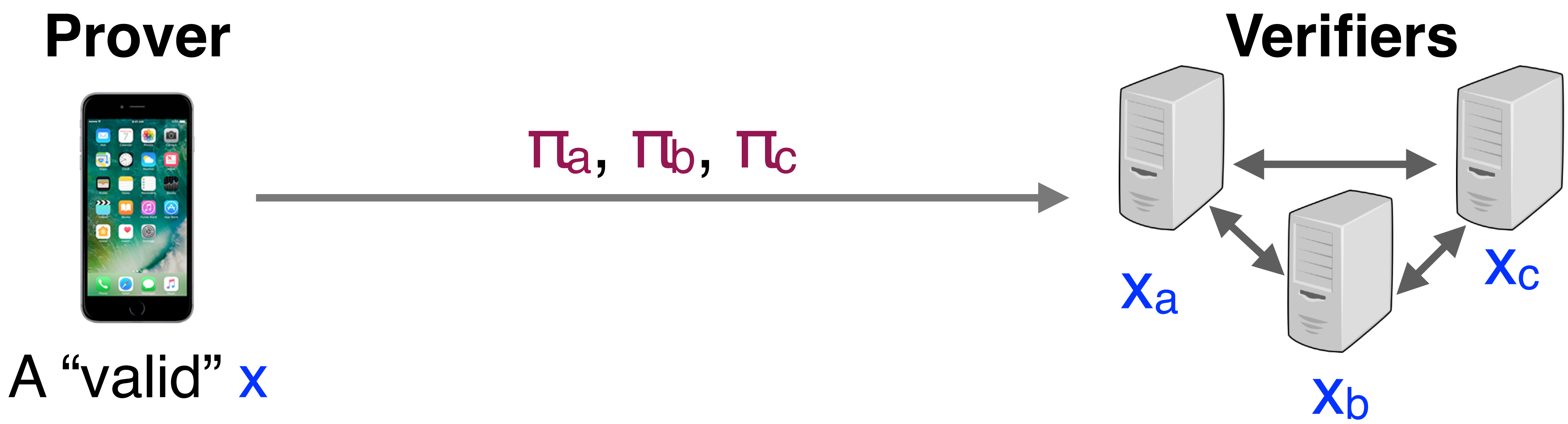
We need a proof system



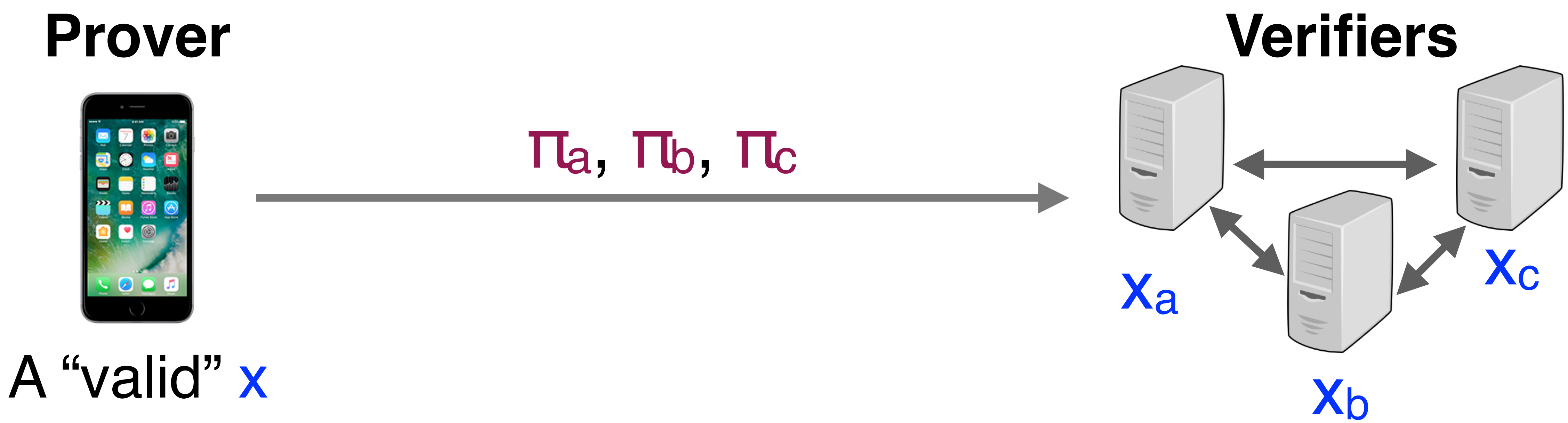
We need a proof system



We need a proof system



We need a proof system



Completeness.	Honest prover convinces honest verifiers.
Soundness.	Dishonest prover rarely convinces honest verifiers.
Zero knowledge.	Any proper subset of the verifiers learns nothing about x, except that x is valid.

Traditional techniques

- Non-interactive proofs in ROM
  [FS86], [BFM88], [BDMP91], [CP92], [CS97], [M00], …
- zkSNARKs and KOE-based proofs
  [G10], [L12], [GGPR13], [BCGTV13], [PGHR13], …
- Multi-party computation
  [Y82], [GMW87], [BGW88], [CCD88], [CLOS02], [DPSZ12], [DKLPSS13], …

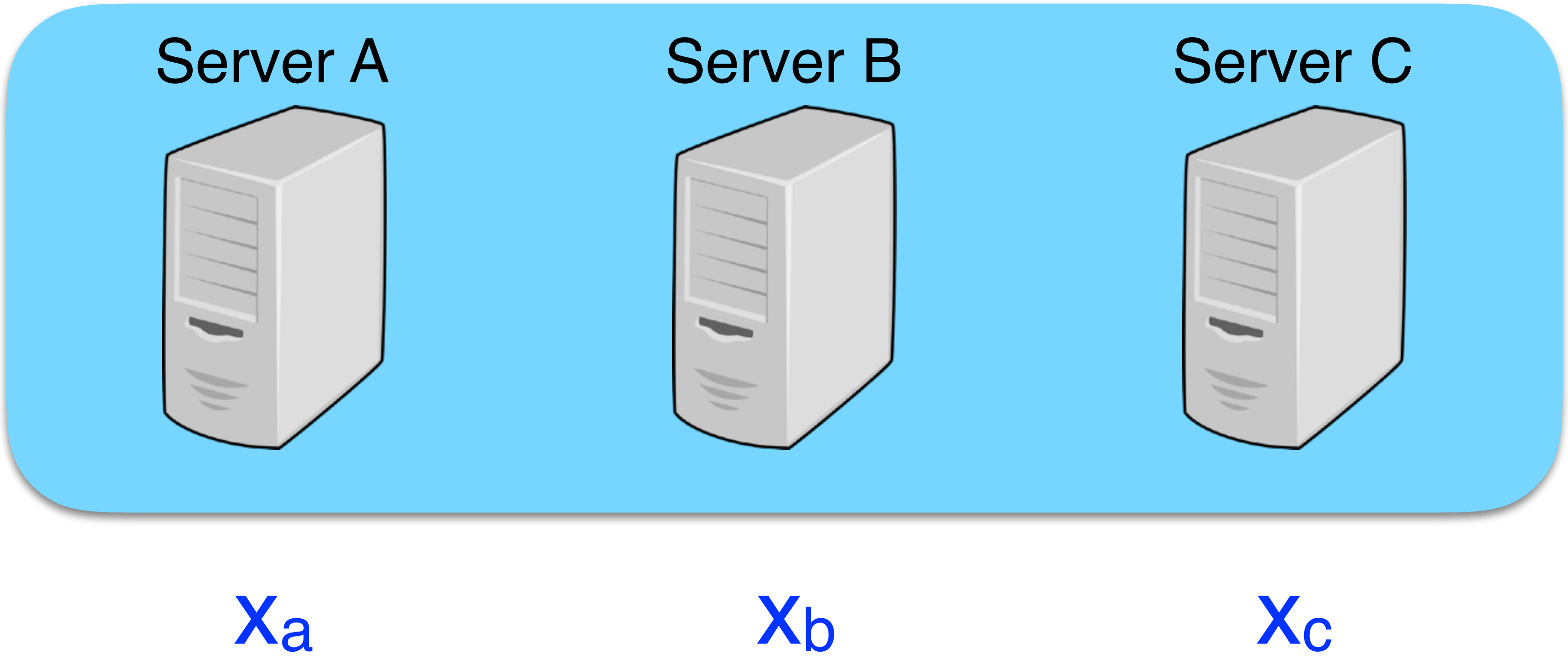
In our setting, SNIPs are a more efficient solution.

Traditional techniques

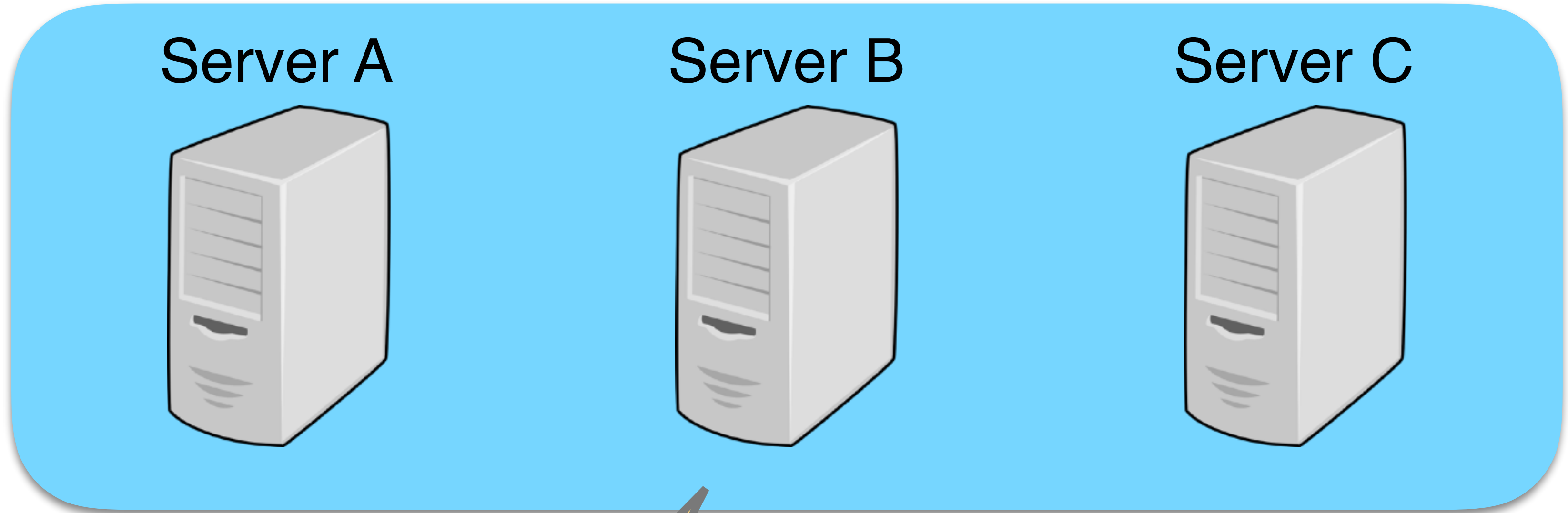
- Non-interactive proofs in ROM
  [FS86], [BFM88], [BDMP91], [CP92], [CS97], [M00], …
- zkSNARKs and KOE-based proofs
  [G10], [L12], [GGPR13], [BCGTV13], [PGHR13], …
- Multi-party computation
  [Y82], [GMW87], [BGW88], [CCD88], [CLOS02], [DPSZ12], [DKLPSS13], …

In our setting, SNIPs are a more efficient solution.

How SNIPs work



How SNIPs work

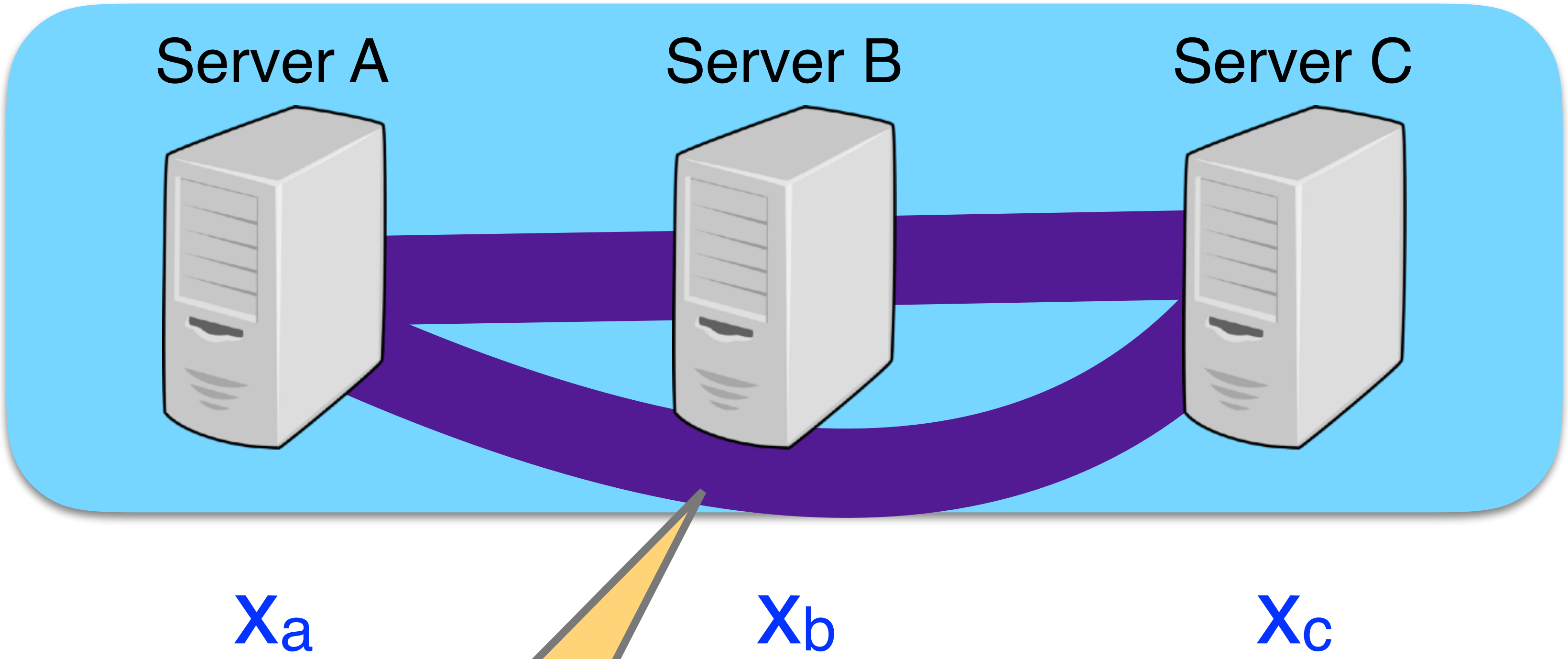


x_a x_b x_c

Could run **secure multiparty computation** to check that Valid(x) = 1.

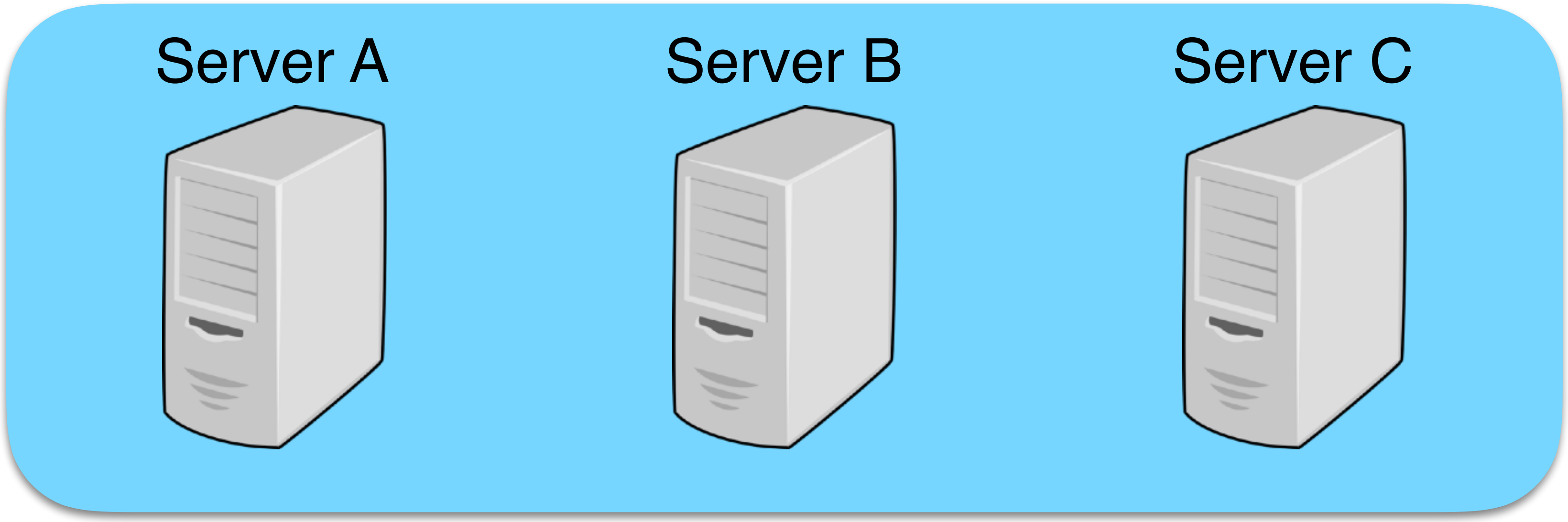
[GMW87], [BGW88]

How SNIPs work



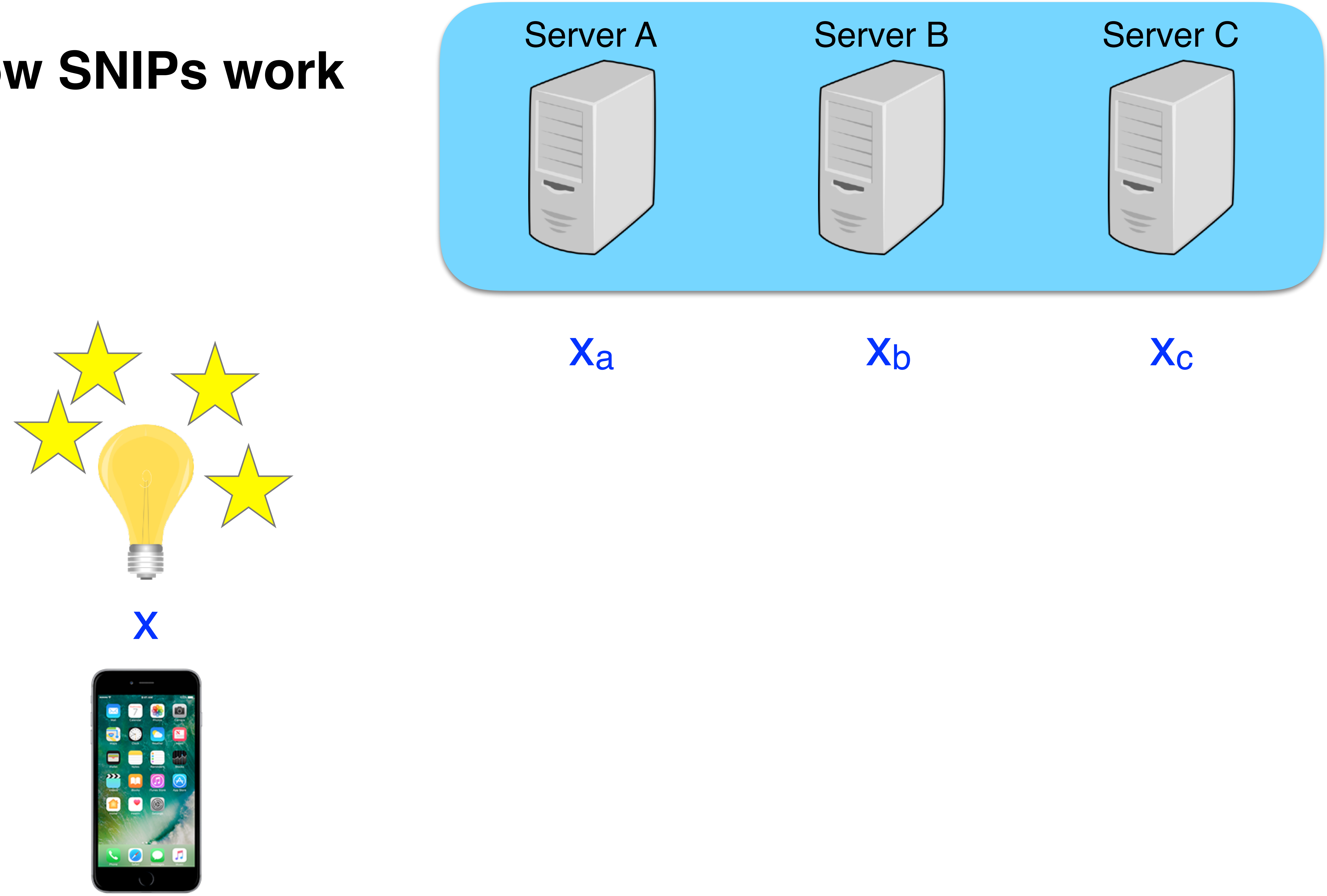
Could run **secure multiparty computation** to check that Valid(x) = 1.
[GMW87], [BGW88]

How SNIPs work

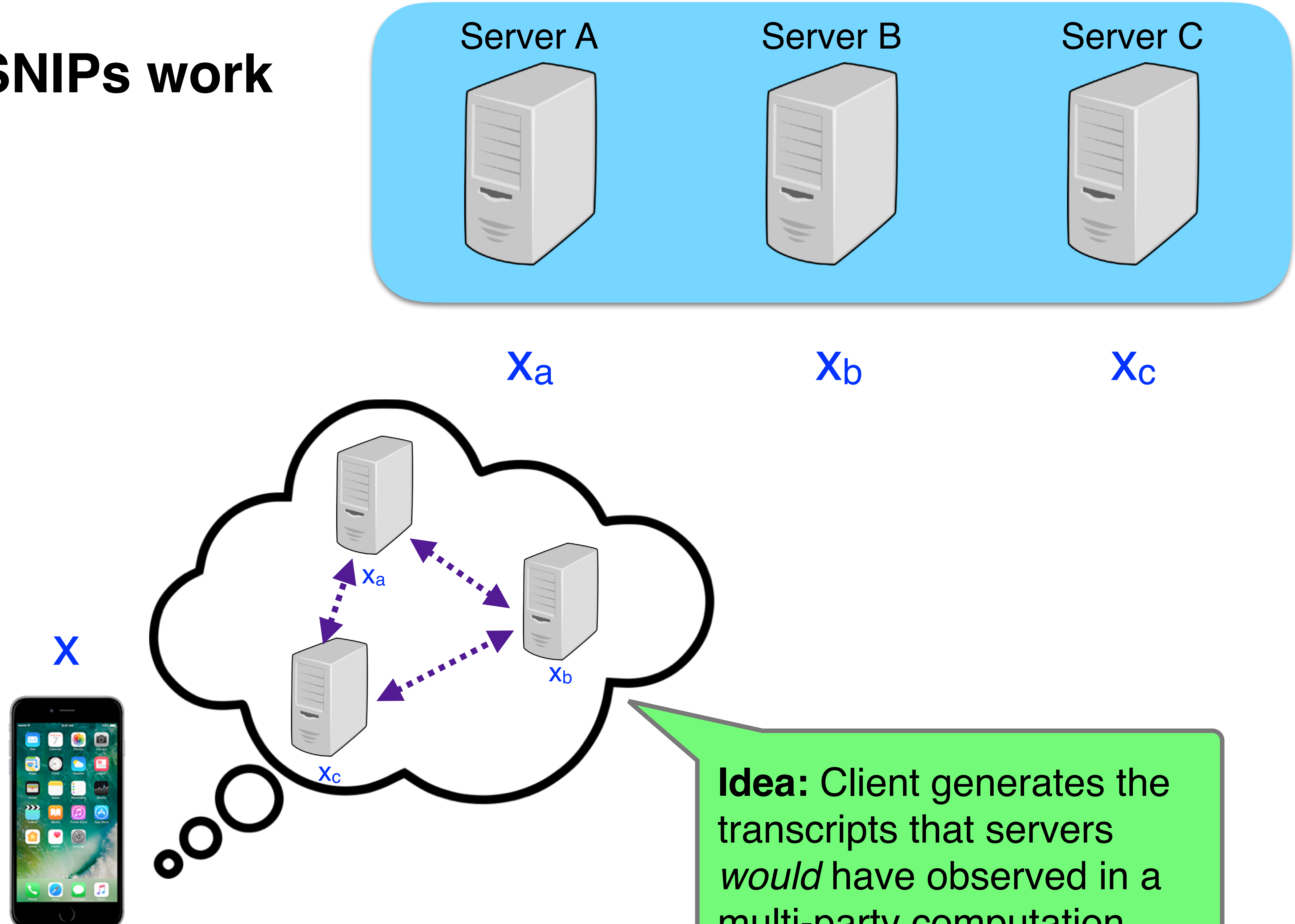


x_a x_b x_c

How SNIPs work



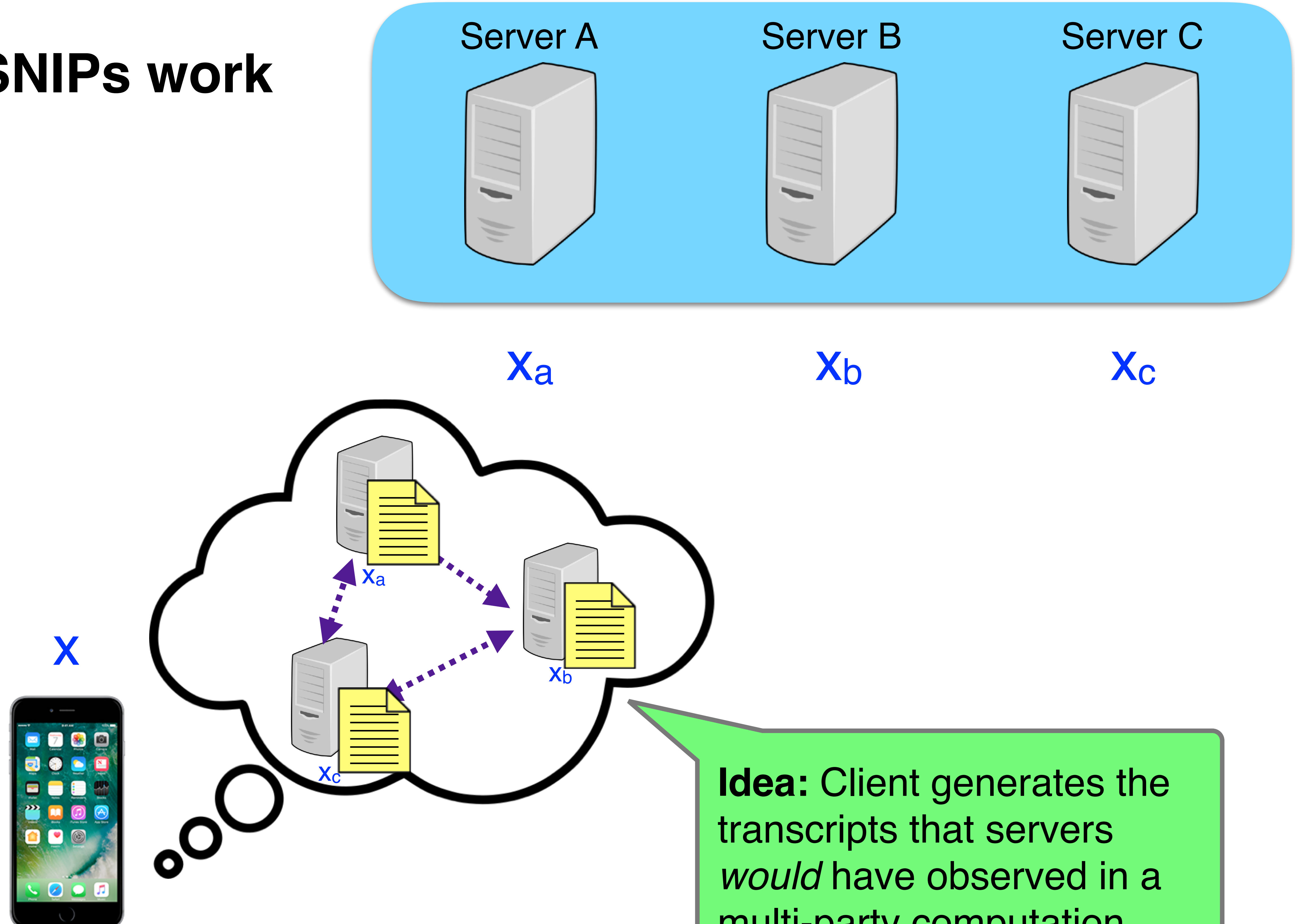
How SNIPs work



Idea: Client generates the transcripts that servers *would* have observed in a multi-party computation

See also [IKOS07]

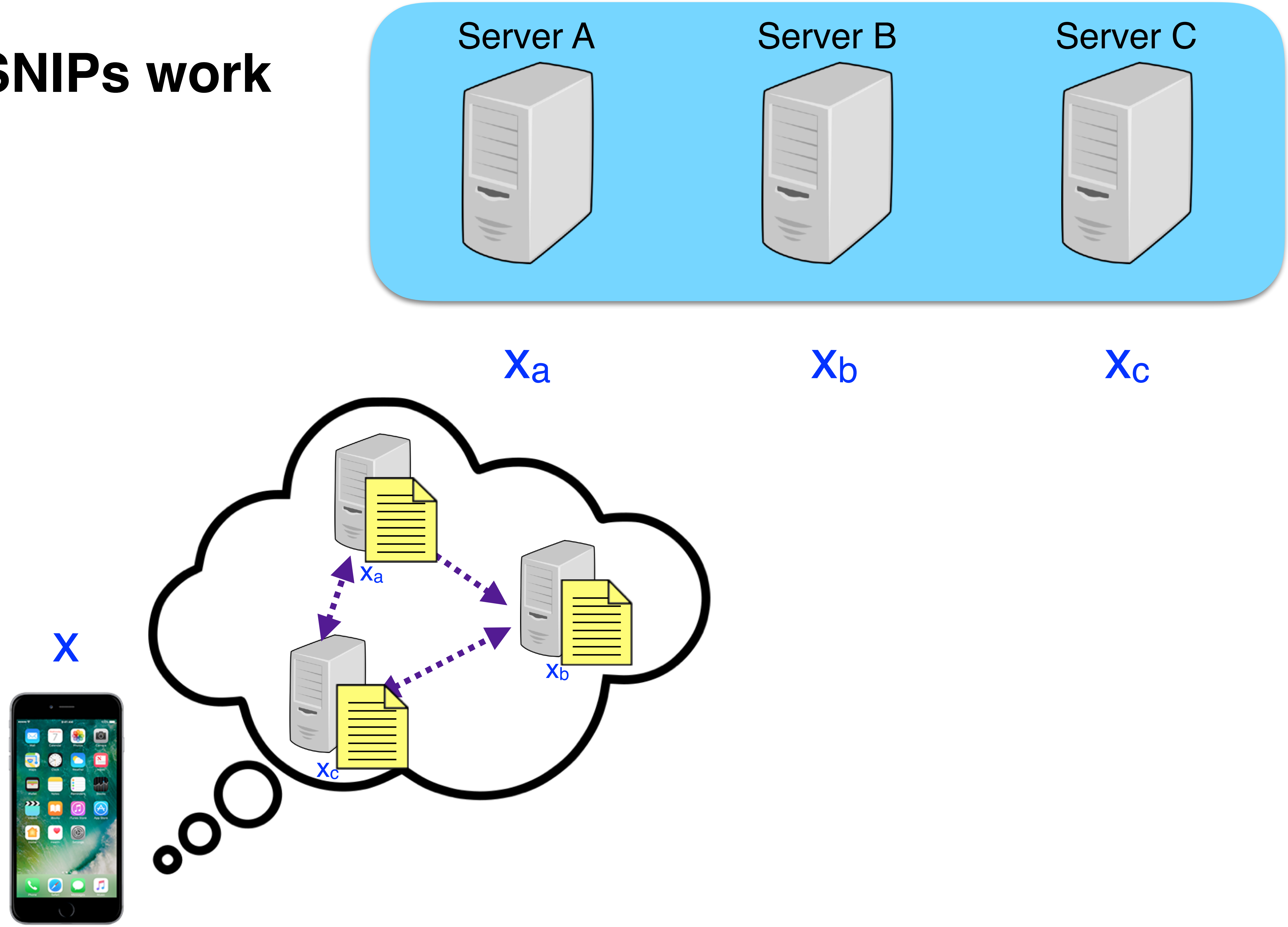
How SNIPs work



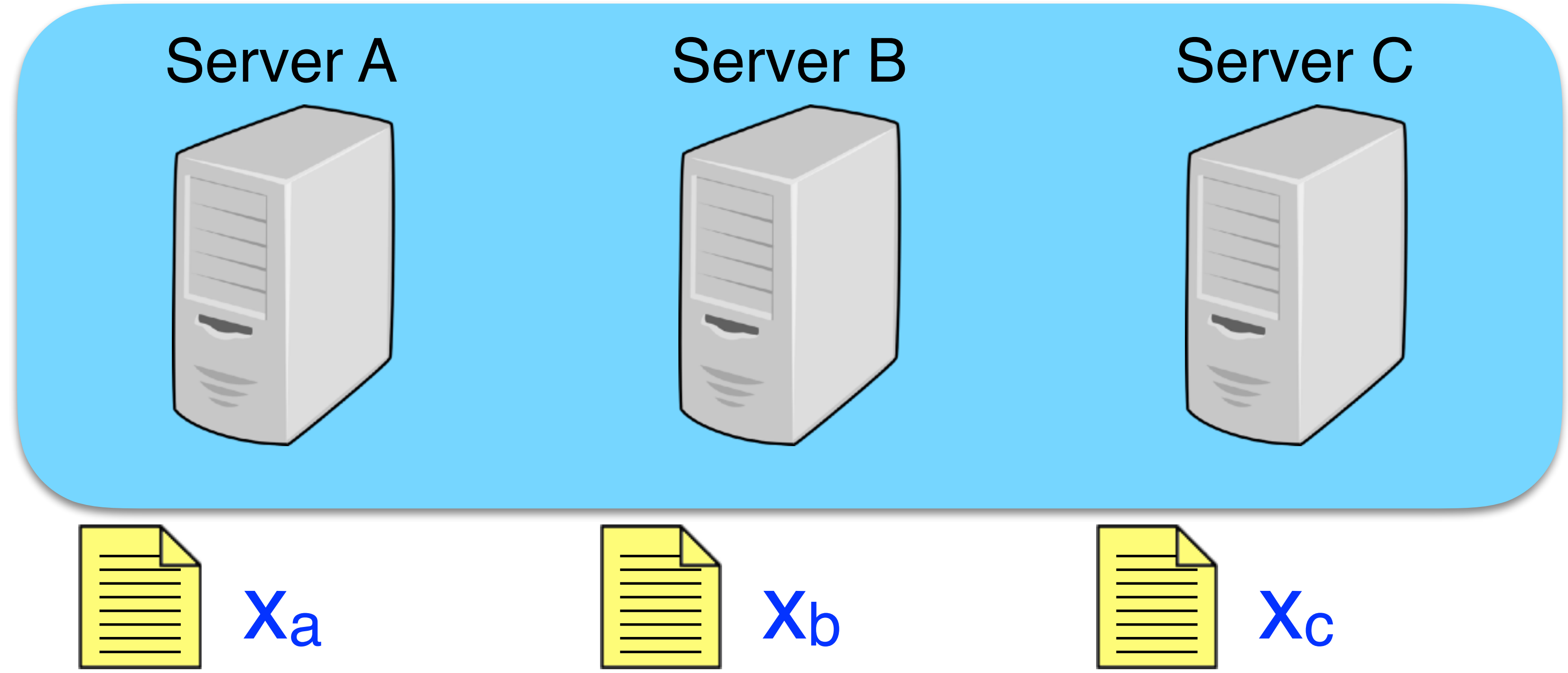
Idea: Client generates the transcripts that servers *would* have observed in a multi-party computation

See also [IKOS07]

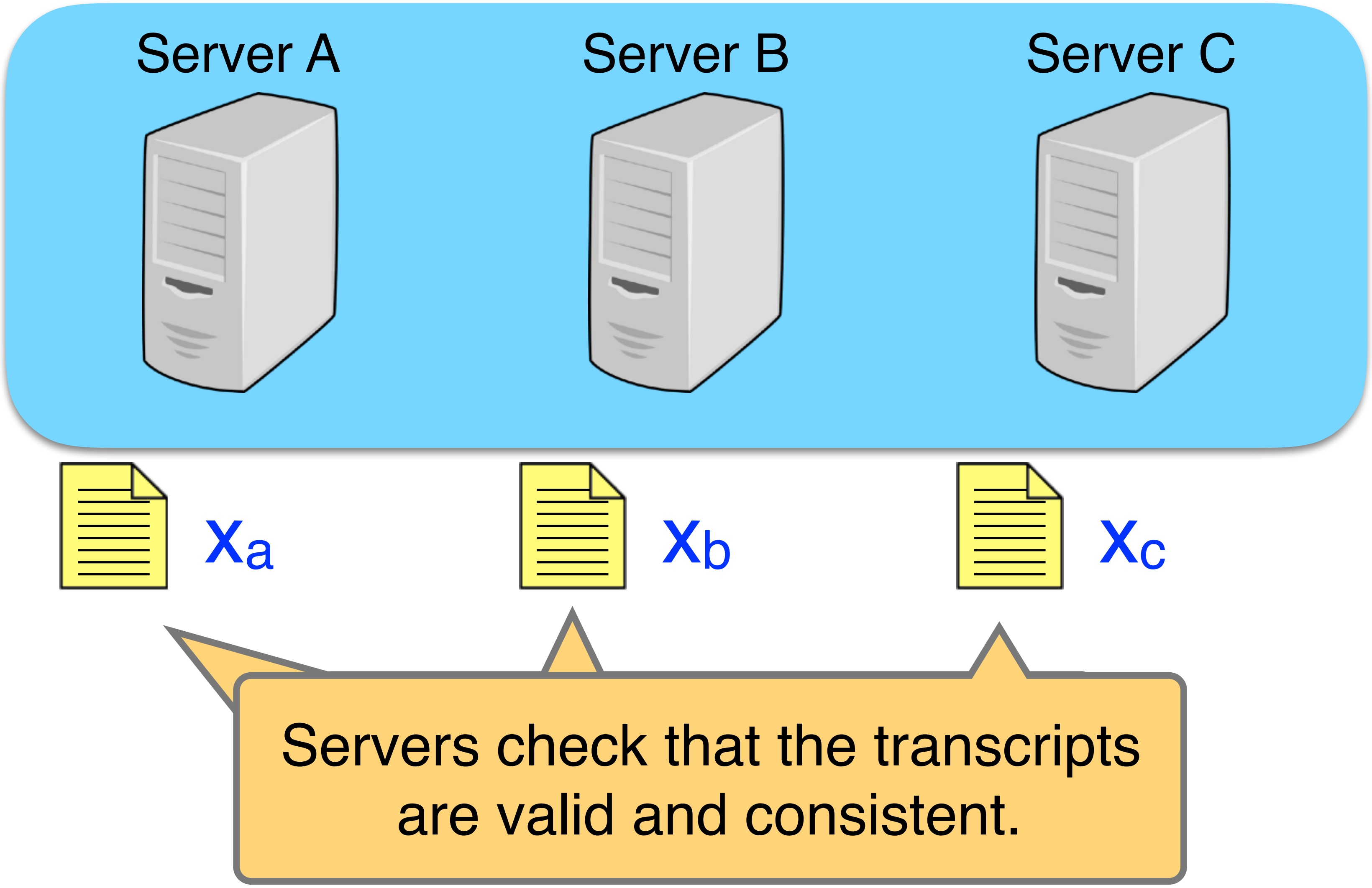
How SNIPs work



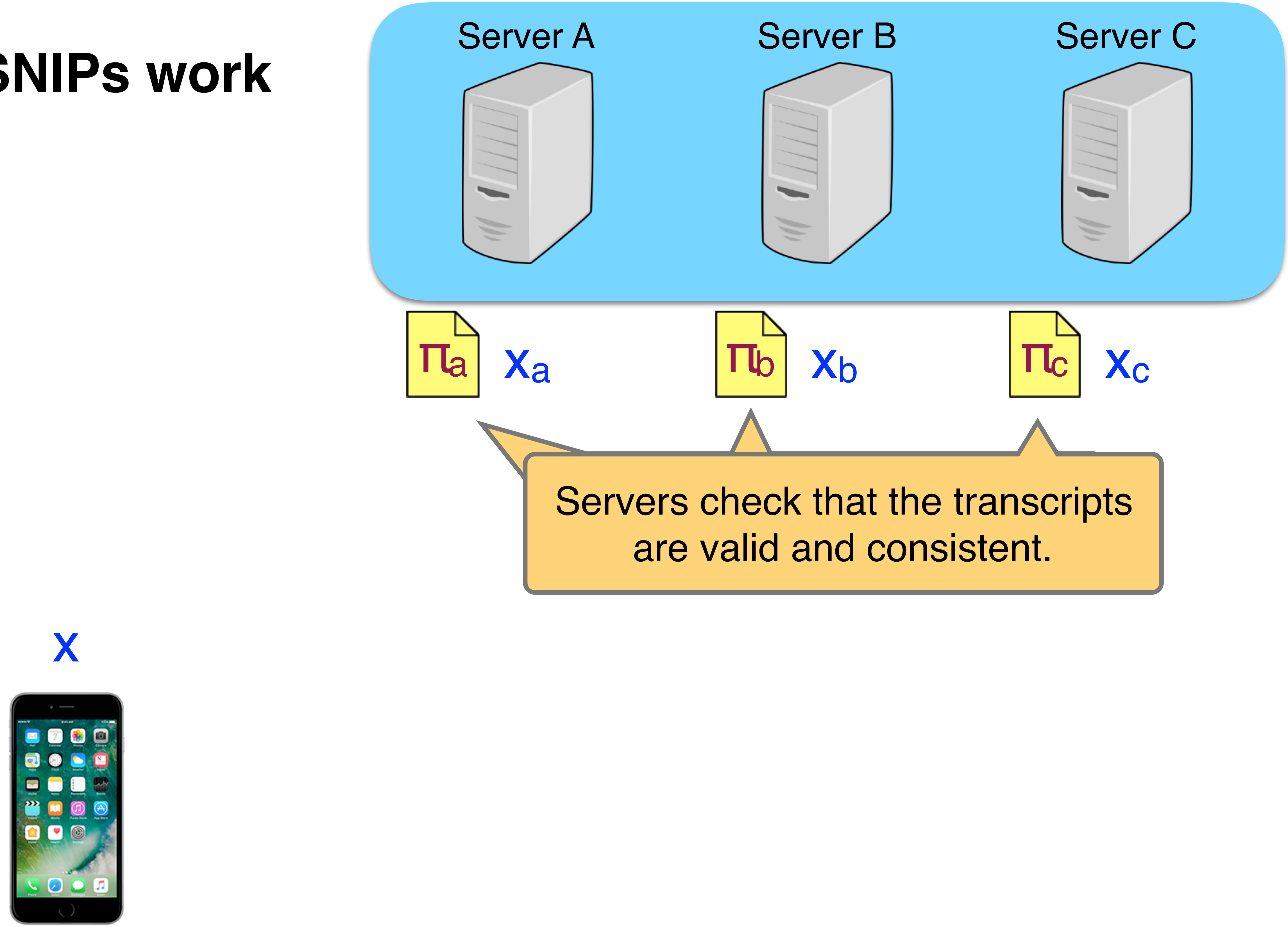
How SNIPs work



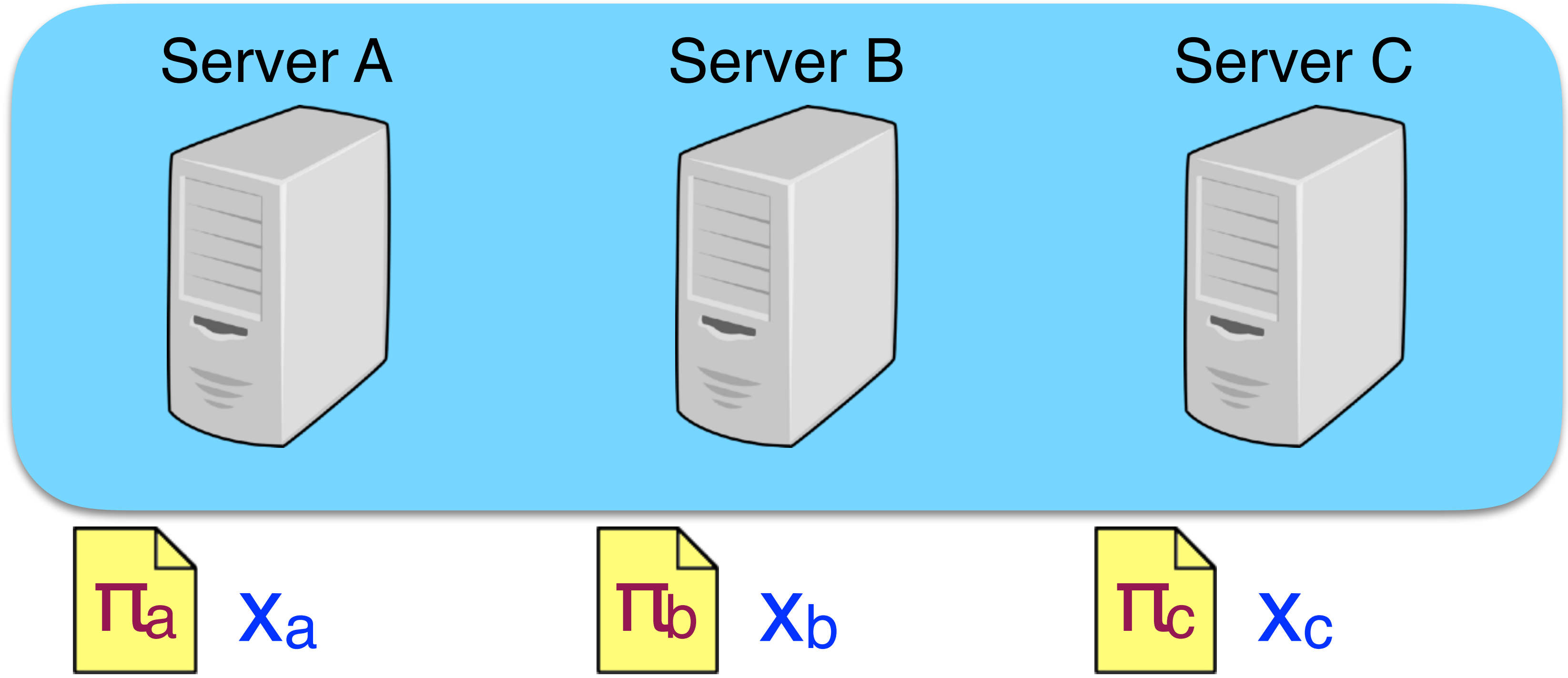
How SNIPs work



How SNIPs work



How SNIPs work

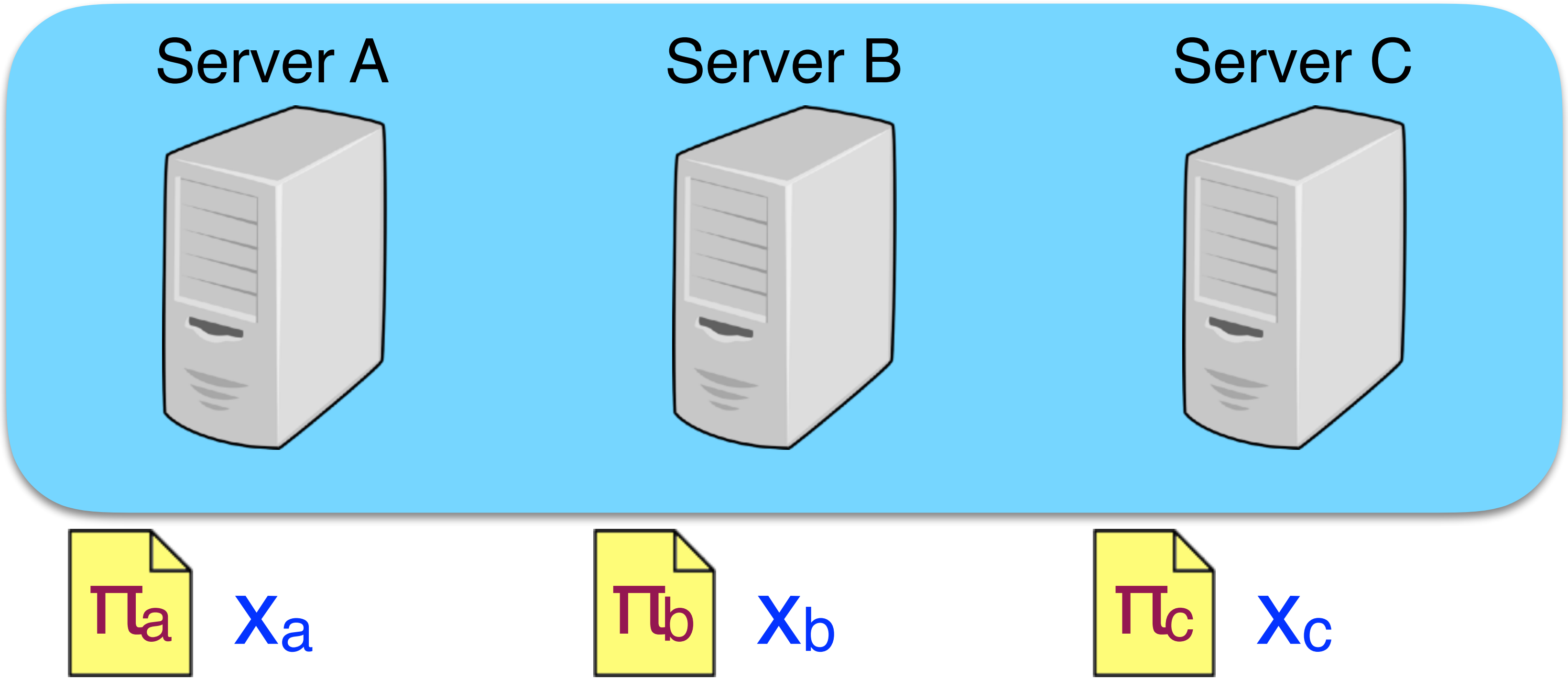


Servers check that the transcripts are valid and consistent.

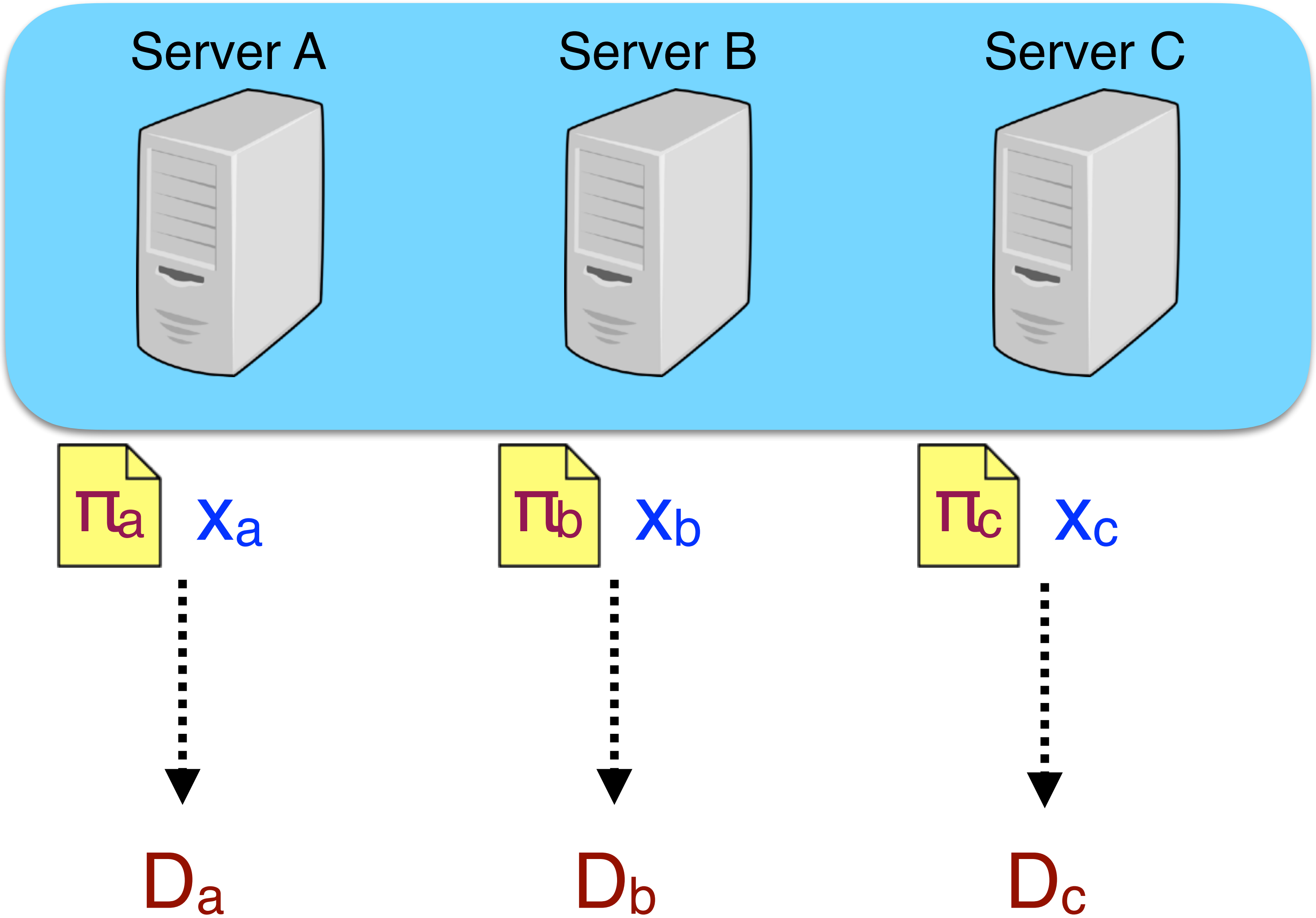
Checking a transcript is **much easier** than generating it!

x

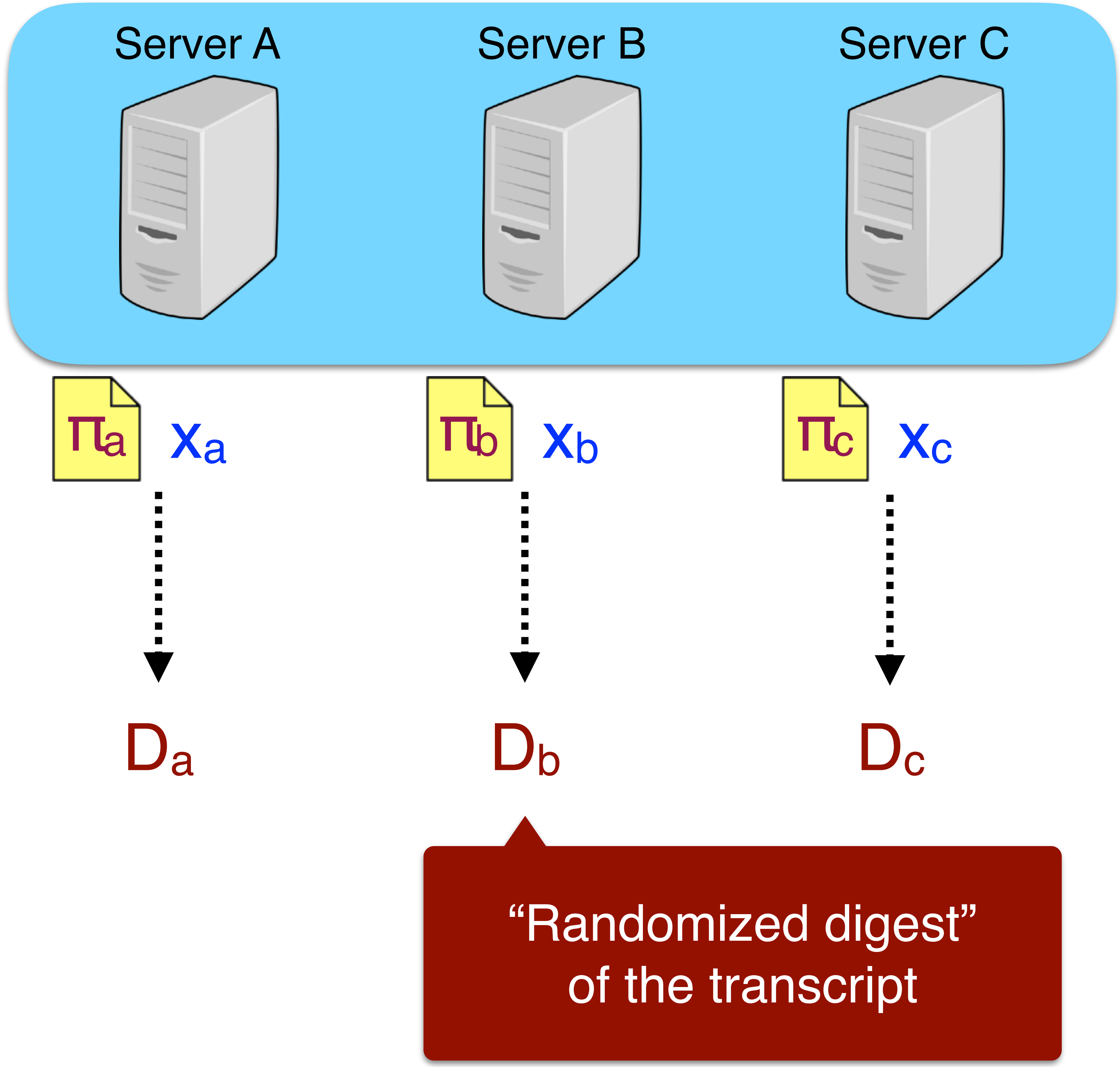
How SNIPs work



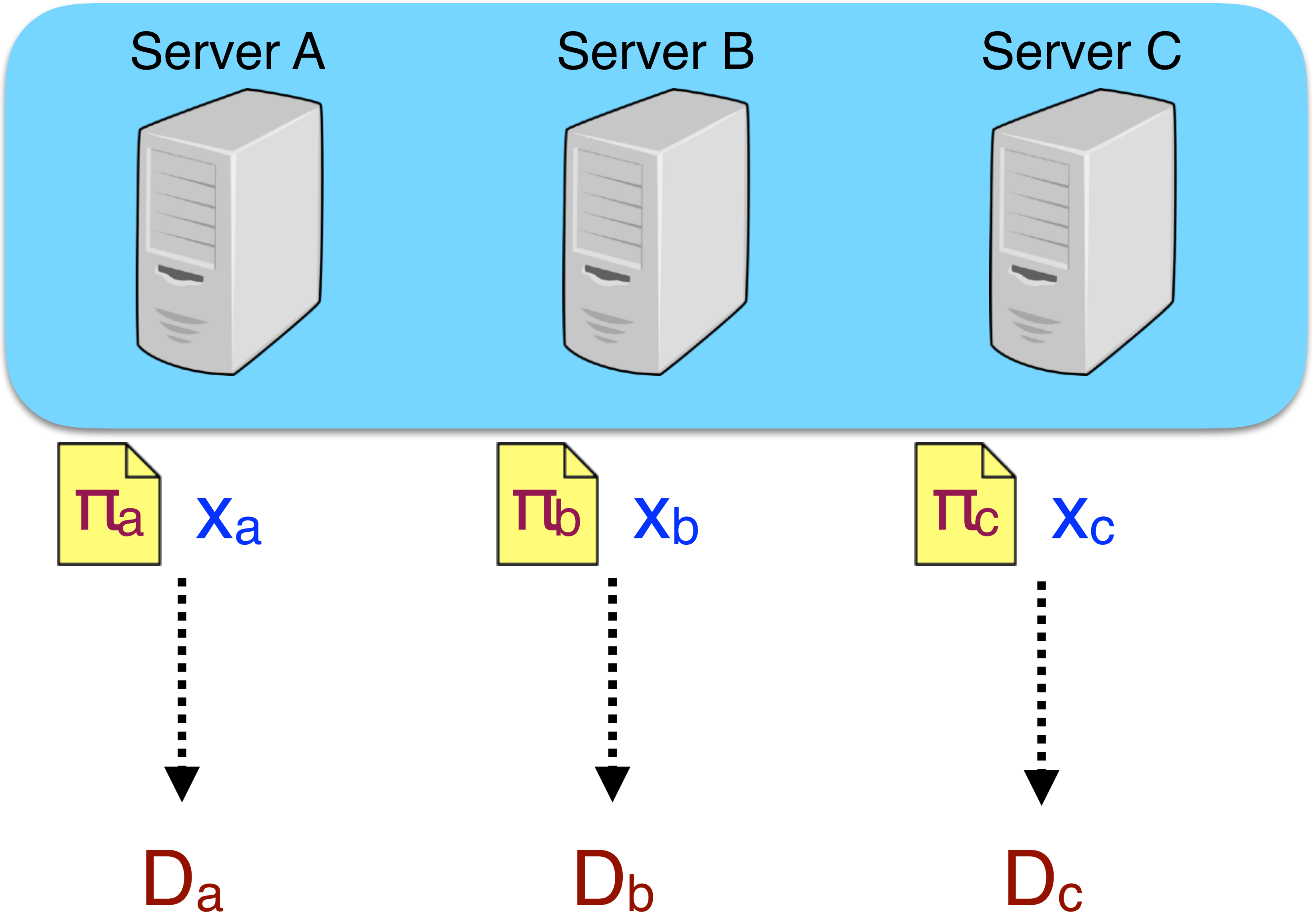
How SNIPs work



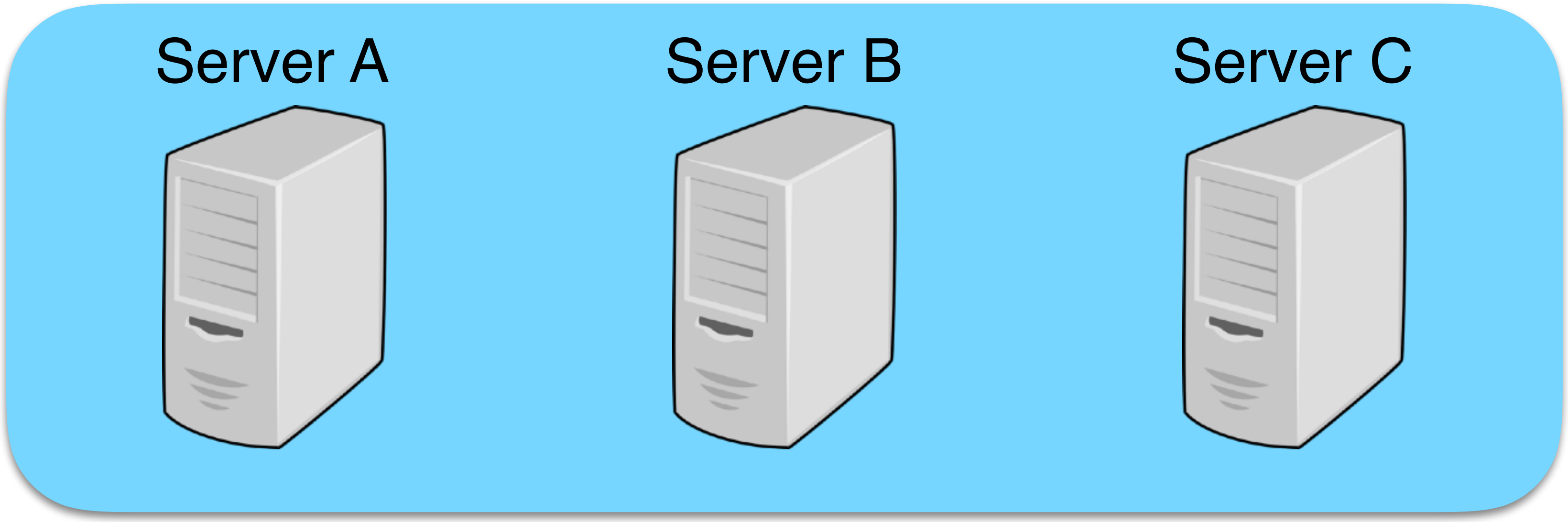
How SNIPs work



How SNIPs work



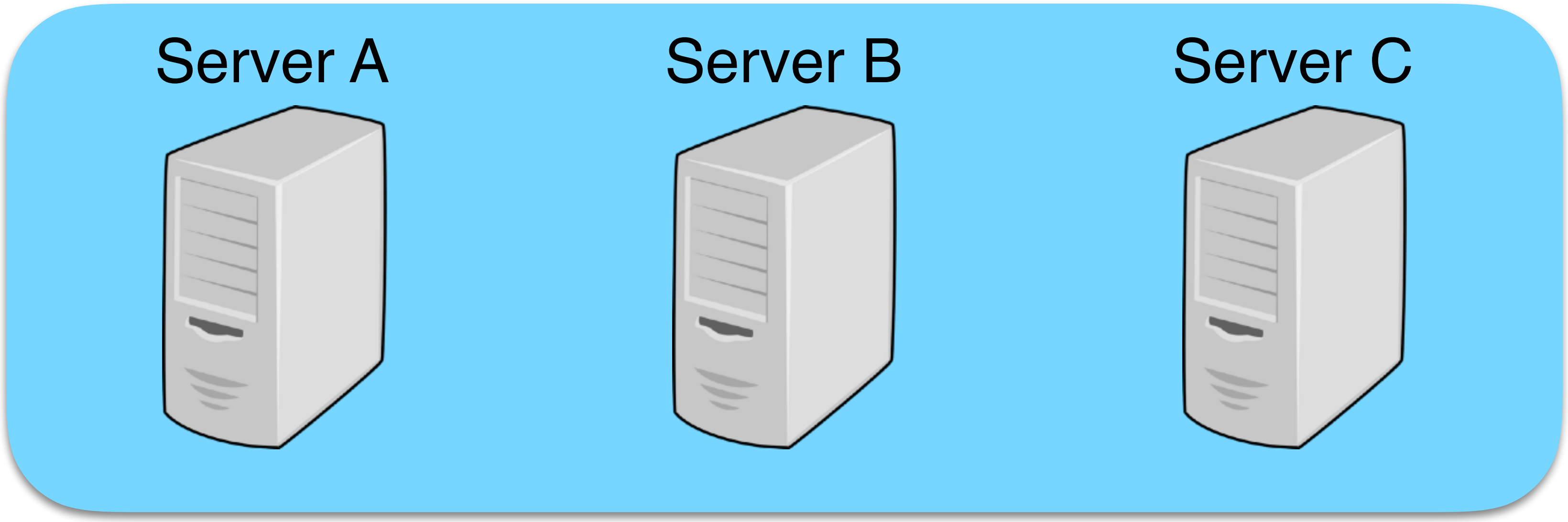
How SNIPs work



D_a D_b D_c

[BFO12]

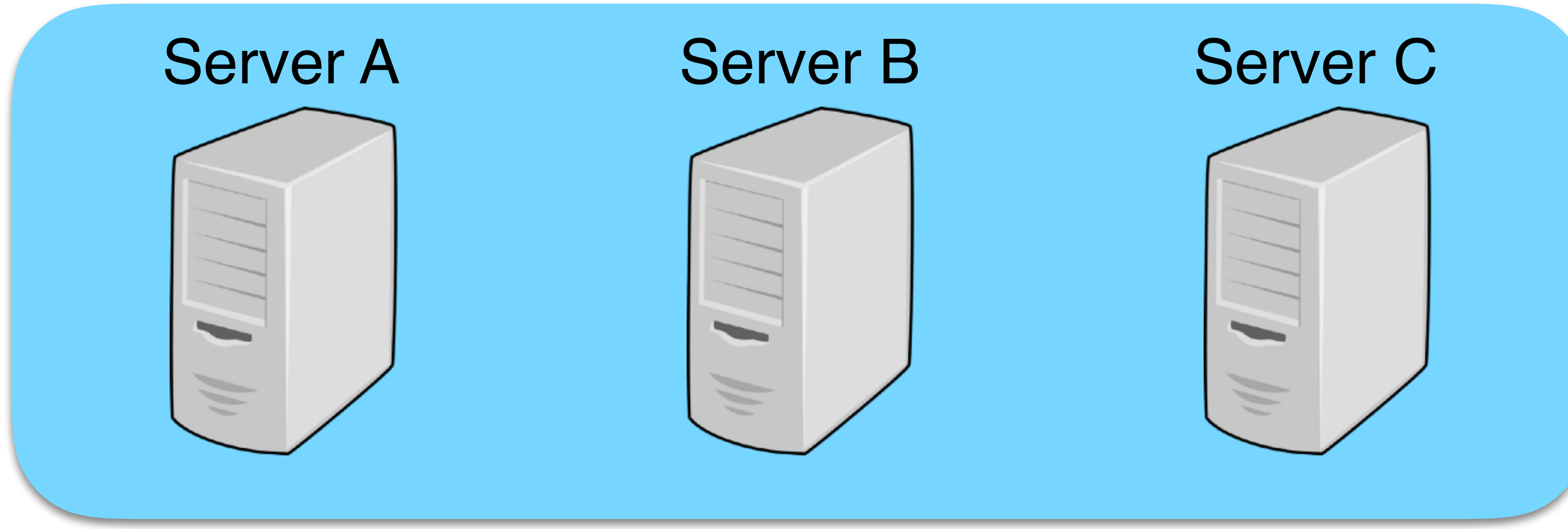
How SNIPs work



D_a D_b D_c

[BFO12]

How SNIPs work



D_a D_b D_c

- If x is valid, $D_a + D_b + D_c = 0$
- If x is invalid, $D_a + D_b + D_c \neq 0$ with high probability

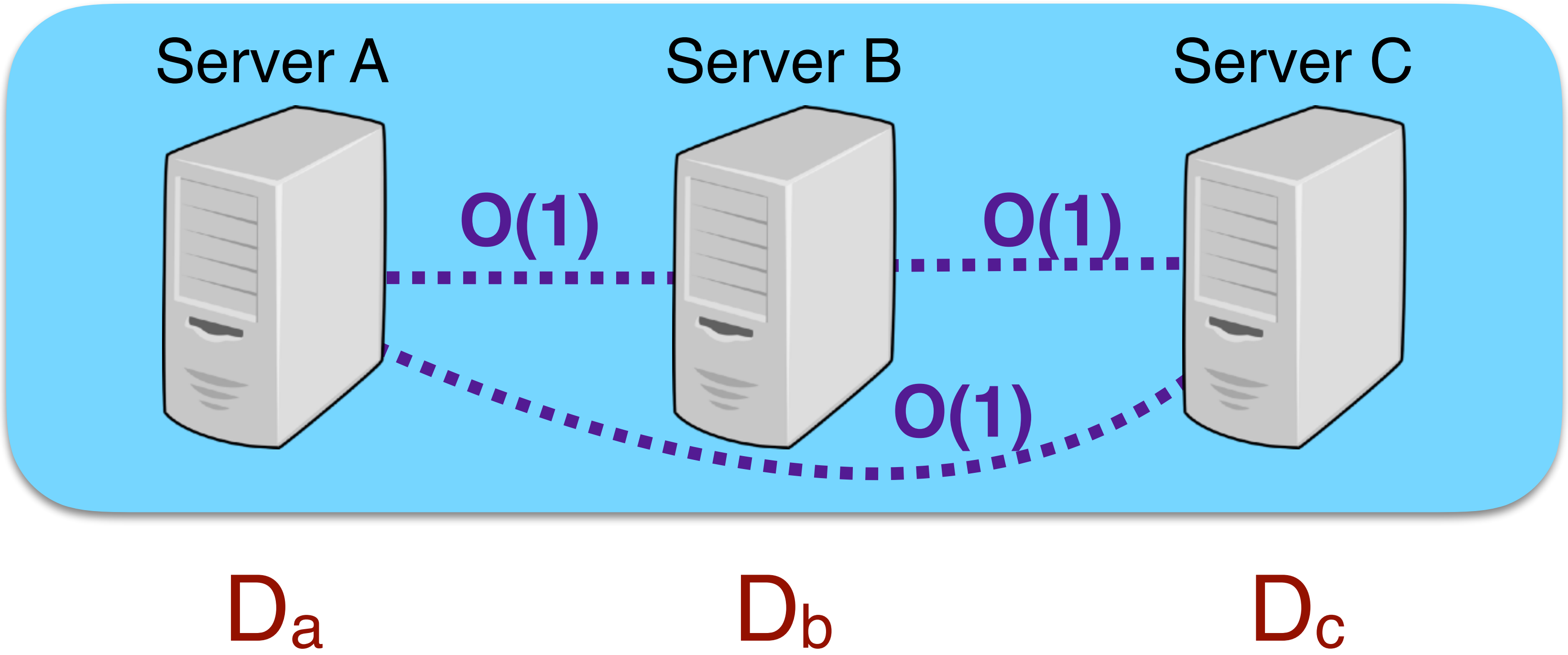
Servers run lightweight multi-party computation to check that

$$D_a + D_b + D_c = 0$$

If so, servers accept x is valid.

[BFO12]

How SNIPs work



- If x is valid, $D_a + D_b + D_c = 0$
- If x is invalid, $D_a + D_b + D_c \neq 0$ with high probability

Servers run lightweight multi-party computation to check that

$$D_a + D_b + D_c = 0$$

If so, servers accept x is valid.

[BFO12]

M = # of multiplication gates in Valid(·) circuit	**Public-key ops.**		**Communication**		**Slow-down**
	Client	Server	C-to-S	S-to-S	

M = # of multiplication gates in Valid(·) circuit	**Public-key ops.** Client	Server	**Communication** C-to-S	S-to-S	**Slow-down**
Dishonest-maj. MPC [CLOS02], [DPSZ12], …	0	Θ(M)	0	Θ(M)	5,000x at server

M = # of multiplication gates in Valid(·) circuit	**Public-key ops.** Client	Server	**Communication** C-to-S	S-to-S	**Slow-down**
Dishonest-maj. MPC [CLOS02], [DPSZ12], …	0	Θ(M)	0	Θ(M)	5,000x at server
Commits + NIZKs [FS86], [CP92], [CS97], …	Θ(M)	Θ(M)	Θ(M)	Θ(M)	50x at server

M = # of multiplication gates in Valid(·) circuit	**Public-key ops.** Client	**Public-key ops.** Server	**Communication** C-to-S	**Communication** S-to-S	**Slow-down**
Dishonest-maj. MPC [CLOS02], [DPSZ12], …	0	Θ(M)	0	Θ(M)	5,000x at server
Commits + NIZKs [FS86], [CP92], [CS97], …	Θ(M)	Θ(M)	Θ(M)	Θ(M)	50x at server
Commits + SNARKs [GGPR13], [BCGTV13], …	Θ(M)	O(1)	O(1)	O(1)	500x at client

M = # of multiplication gates in Valid(·) circuit	**Public-key ops.** Client	Server	**Communication** C-to-S	S-to-S	**Slow-down**
Dishonest-maj. MPC [CLOS02], [DPSZ12], …	0	Θ(M)	0	Θ(M)	5,000x at server
Commits + NIZKs [FS86], [CP92], [CS97], …	Θ(M)	Θ(M)	Θ(M)	Θ(M)	50x at server
Commits + SNARKs [GGPR13], [BCGTV13], …	Θ(M)	O(1)	O(1)	O(1)	500x at client
This work: SNIPs	0	0	Θ(M)	O(1)	1x

M = # of multiplication gates in Valid(·) circuit	**Public-key ops.** Client	Server	**Communication** C-to-S	S-to-S	**Slow-down**
Dishonest-maj. MPC [CLOS02], [DPSZ12], …	0	Θ(M)	0	Θ(M)	5,000x at server
Commits + NIZKs [FS86], [CP92], [CS97], …	Θ(M)	Θ(M)	Θ(M)	Θ(M)	50x at server
Commits + SNARKs [GGPR13], [BCGTV13], …	Θ(M)	O(1)	O(1)	O(1)	500x at client
This work: SNIPs	0	0	Θ(M)	O(1)	1x

For specific Valid() circuits, it is possible to eliminate this cost [BGI16]

M = # of multiplication gates in Valid(·) circuit	**Public-key ops.** Client	Server	**Communication** C-to-S	S-to-S	**Slow-down**
Dishonest-maj. MPC [CLOS02], [DPSZ12], …	0	$\Theta(M)$	0	$\Theta(M)$	5,000x at server
Commits + NIZKs [FS86], [CP92], [CS97], …	$\Theta(M)$	$\Theta(M)$	$\Theta(M)$	$\Theta(M)$	50x at server
Commits + SNARKs [GGPR13], [BCGTV13], …	$\Theta(M)$	$O(1)$	$O(1)$	$O(1)$	500x at client
This work: SNIPs	0	0	$\Theta(M)$	$O(1)$	1x

From sums to more complex aggregates

If you can compute private sums, you can compute many other interesting aggregates

[PrivStats11], [KDK11], [DFKZ13], [PrivEx14], [MDD16], …

- Average
- Variance
- Standard deviation
- Most popular (approx)
- “Heavy hitters” (approx)
- Min and max (approx)
- Quality of arbitrary regression model (R^2)
- Least-squares regression
- Stochastic gradient descent [Bonawitz et al. 2016]

Outline

- Background: The private aggregation problem
- A straw-man solution for private sums
- **Providing robustness with SNIPs**
- Evaluation
- Discussion: Real-world considerations

Outline

- Background: The private aggregation problem
- A straw-man solution for private sums
- Providing robustness with SNIPs
- **Evaluation**
- Discussion: Real-world considerations

Evaluation

- Implemented Prio in Go
  (see optimizations described in paper)
- Five-server cluster in EC2
- System collects the sum of “N” 0/1 values

Four variants

1. No privacy
2. No robustness (“straw man”)
3. Prio (privacy + robustness)
4. NIZK (privacy + robustness)

Evaluation

- Implemented Prio in Go
  (see optimizations described in paper)
- Five-server cluster in EC2
- System collects the sum of “N” 0/1 values

E.g., for privately measuring telemetry data.

Four variants

1. No privacy
2. No robustness (“straw man”)
3. Prio (privacy + robustness)
4. NIZK (privacy + robustness)

Evaluation

- Implemented Prio in Go
  (see optimizations described in paper)
- Five-server cluster in EC2
- System collects the sum of “N” 0/1 values

Four variants

1. No privacy
2. No robustness (“straw man”)
3. Prio (privacy + robustness)
4. NIZK (privacy + robustness)

Evaluation

- Implemented Prio in Go
  (see optimizations described in paper)
- Five-server cluster in EC2
- System collects the sum of “N” 0/1 values

Four variants

1. No privacy ← **one** server
2. No robustness (“straw man”)
3. Prio (privacy + robustness)
4. NIZK (privacy + robustness)

Evaluation

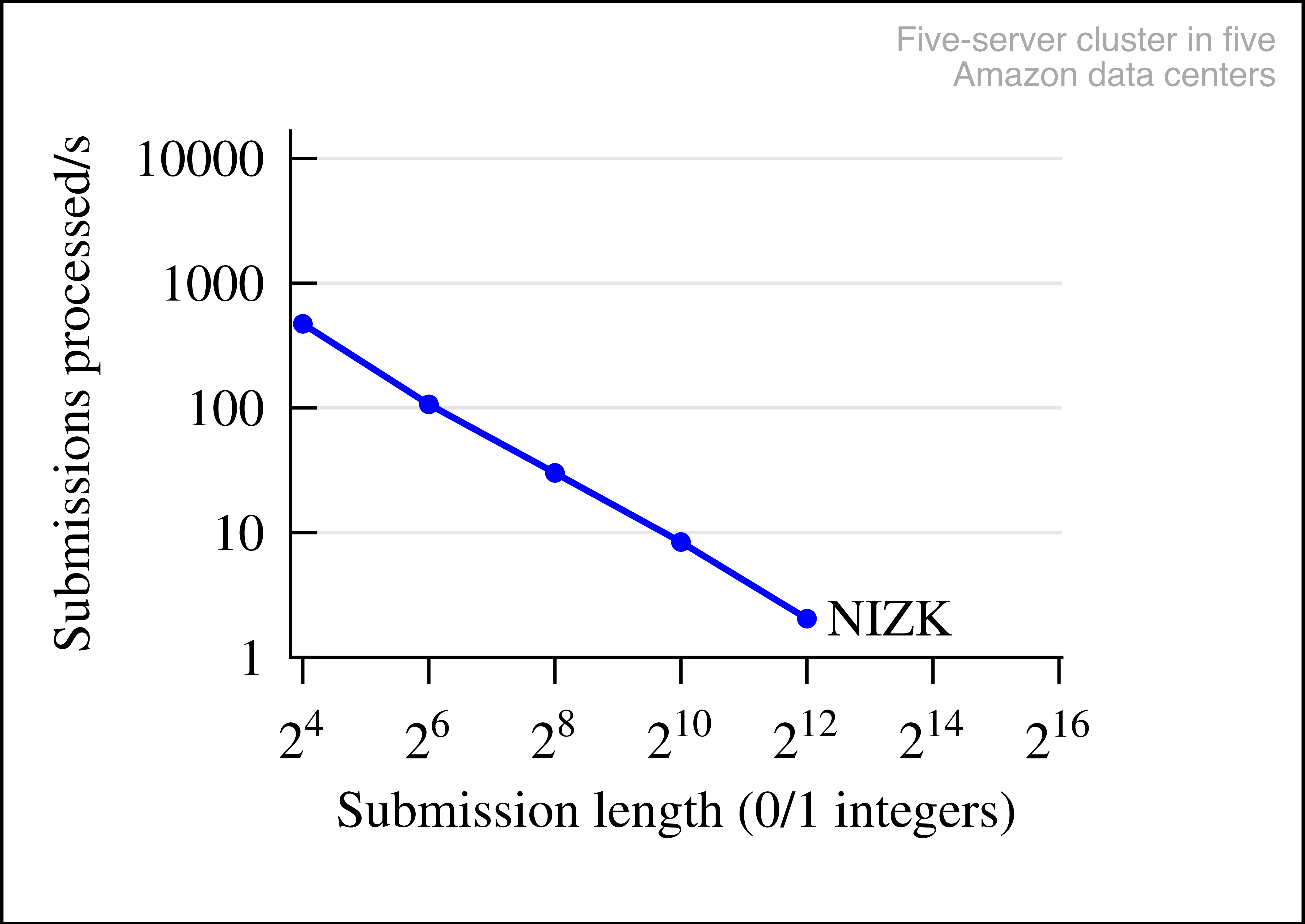
- Implemented Prio in Go
  (see optimizations described in paper)
- Five-server cluster in EC2
- System collects the sum of "N" 0/1 values

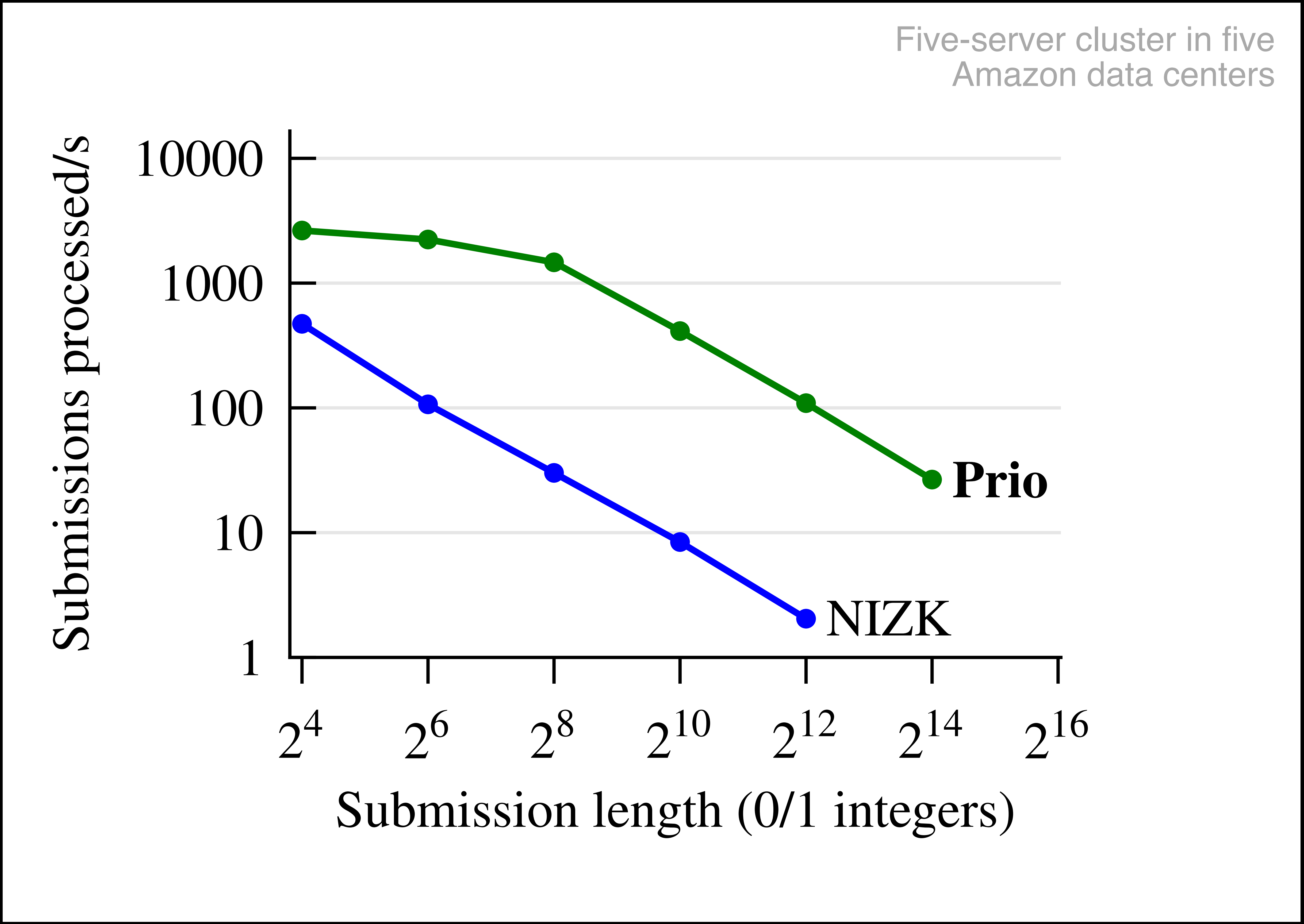
Four variants

1. No privacy
2. No robustness ("straw man")
3. Prio (privacy + robustness)
4. NIZK (privacy + robustness)

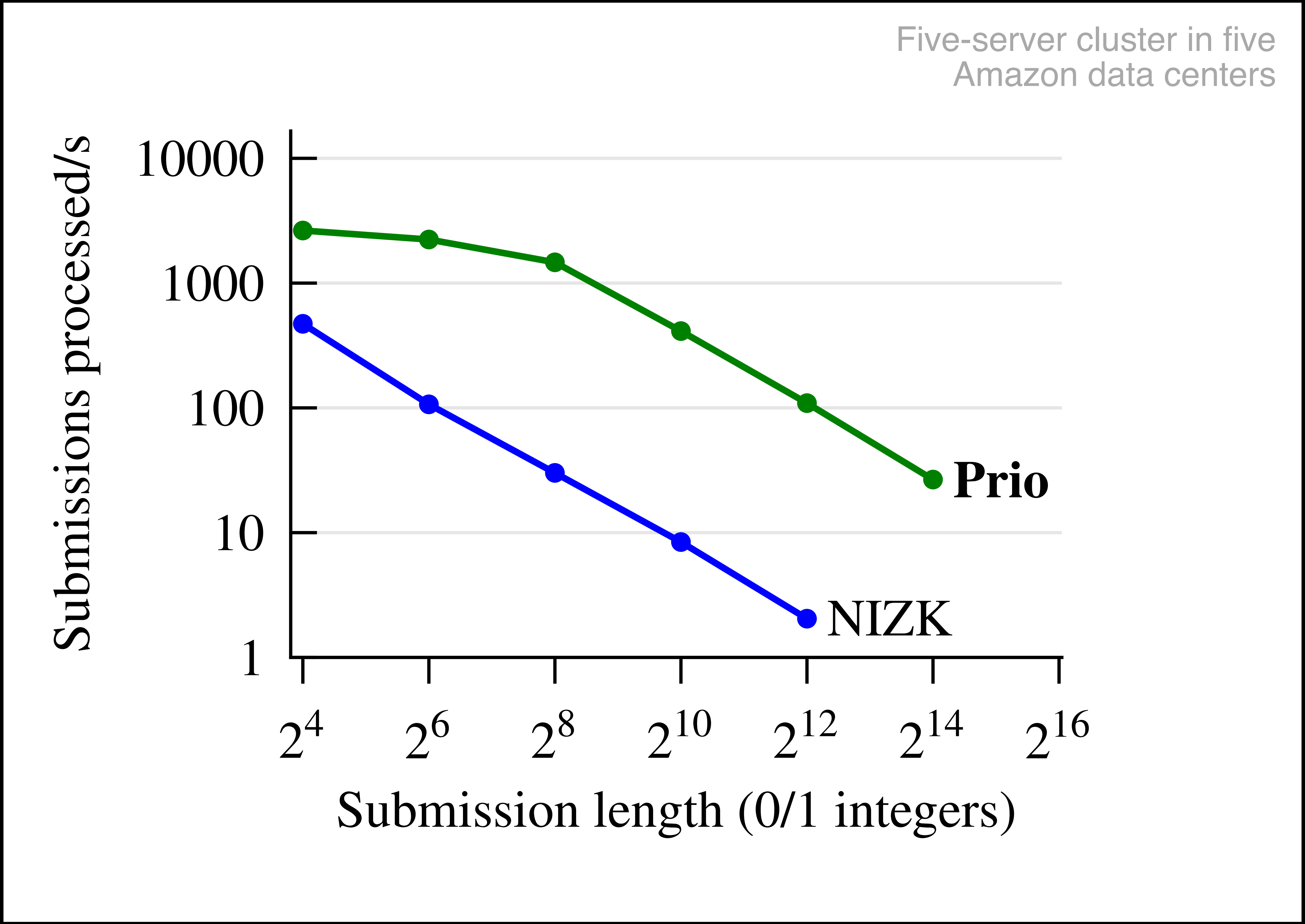
one server

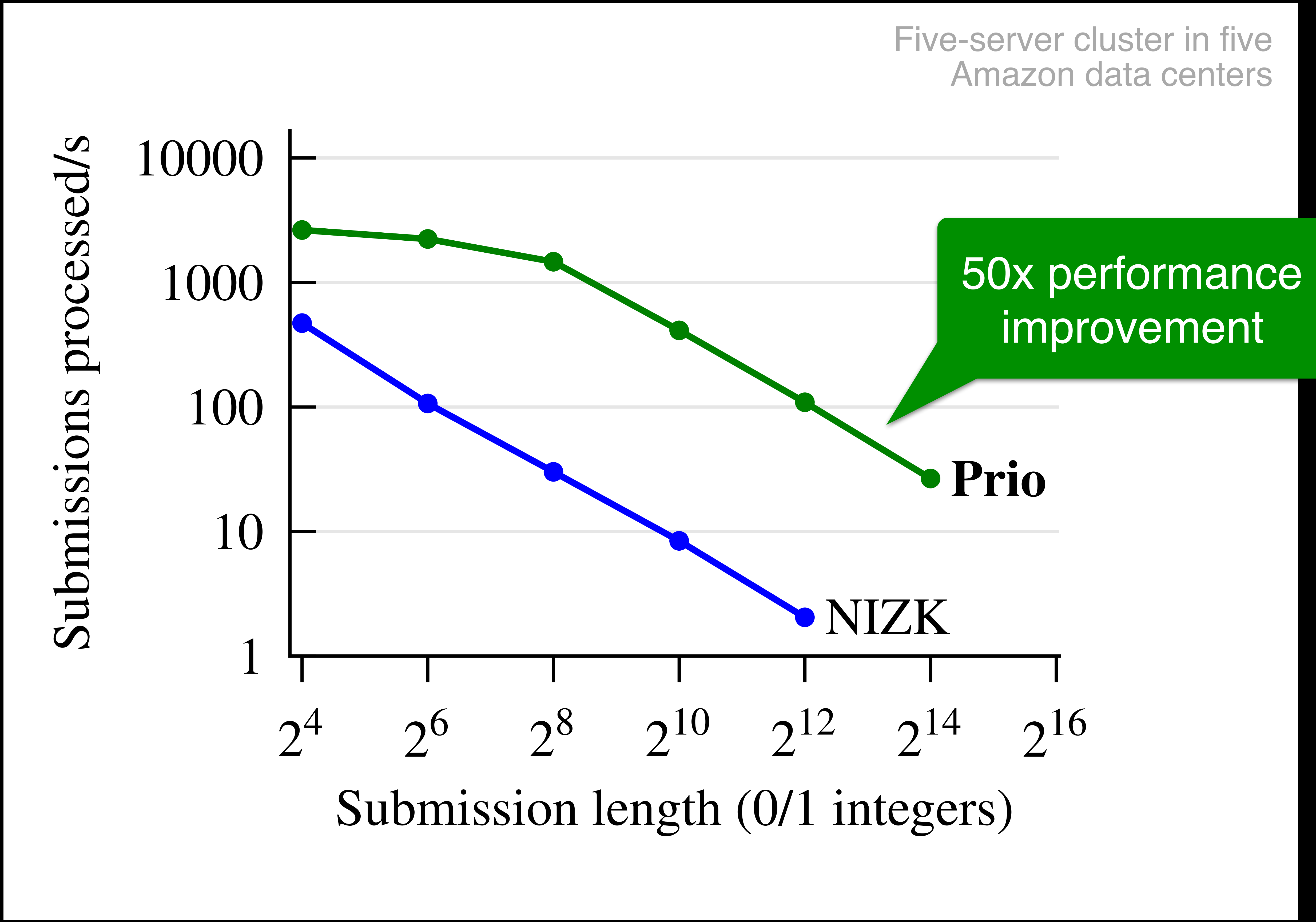
five servers

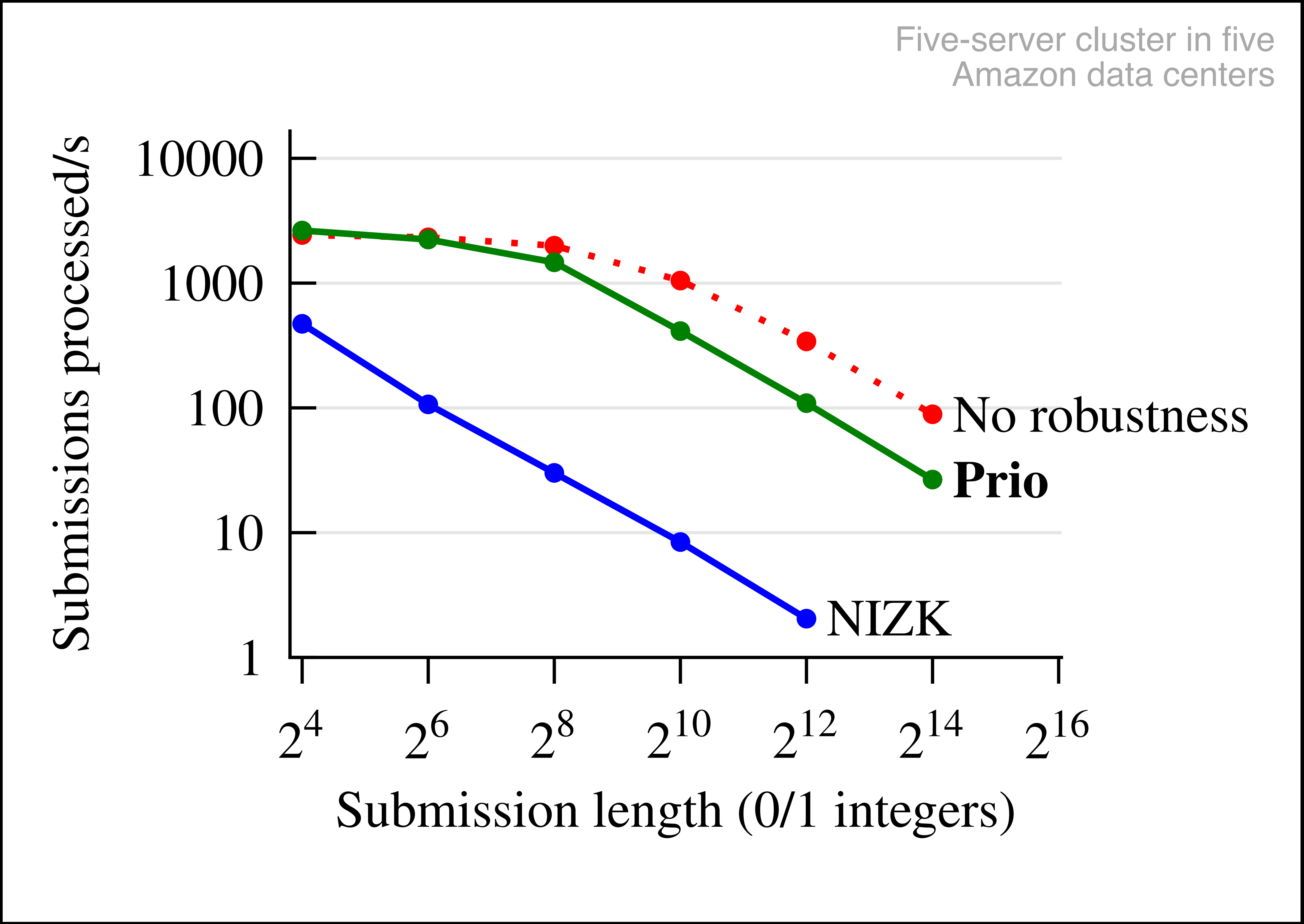

Five-server cluster in five
Amazon data centers
Submissions processed/s
10000
1000
100
10
1
NIZK
2^4
2^6
2^8
2^{10}
2^{12}
2^{14}
2^{16}
Submission length (0/1 integers)



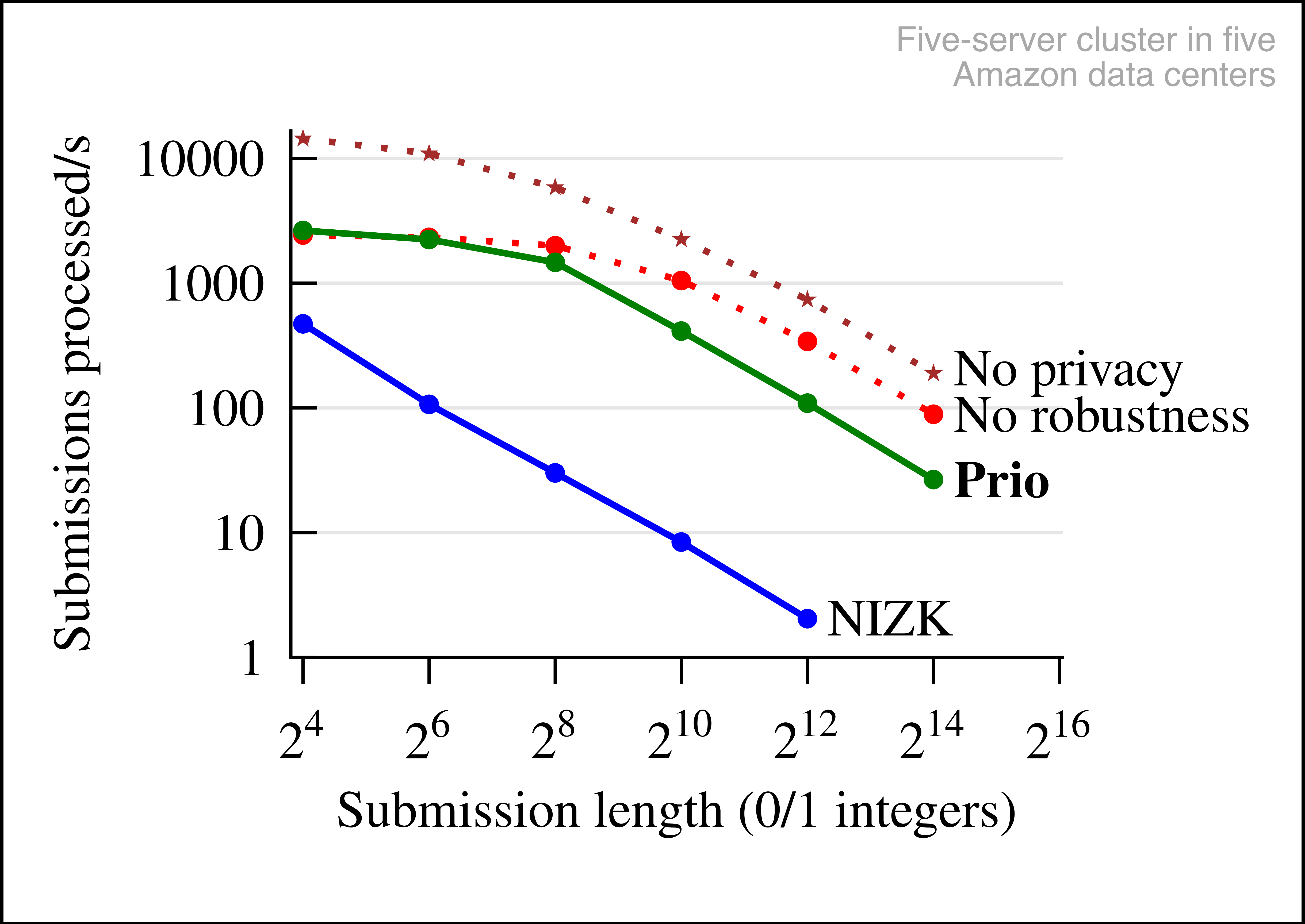
Five-server cluster in five
Amazon data centers
Submissions processed/s
10000
1000
100
10
1
Prio
NIZK
2^4
2^6
2^8
2^{10}
2^{12}
2^{14}
2^{16}
Submission length (0/1 integers)


Five-server cluster in five
Amazon data centers
Submissions processed/s
10000
1000
100
10
1
2^{4}
2^{6}
2^{8}
2^{10}
2^{12}
2^{14}
2^{16}
Submission length (0/1 integers)
Prio
NIZK

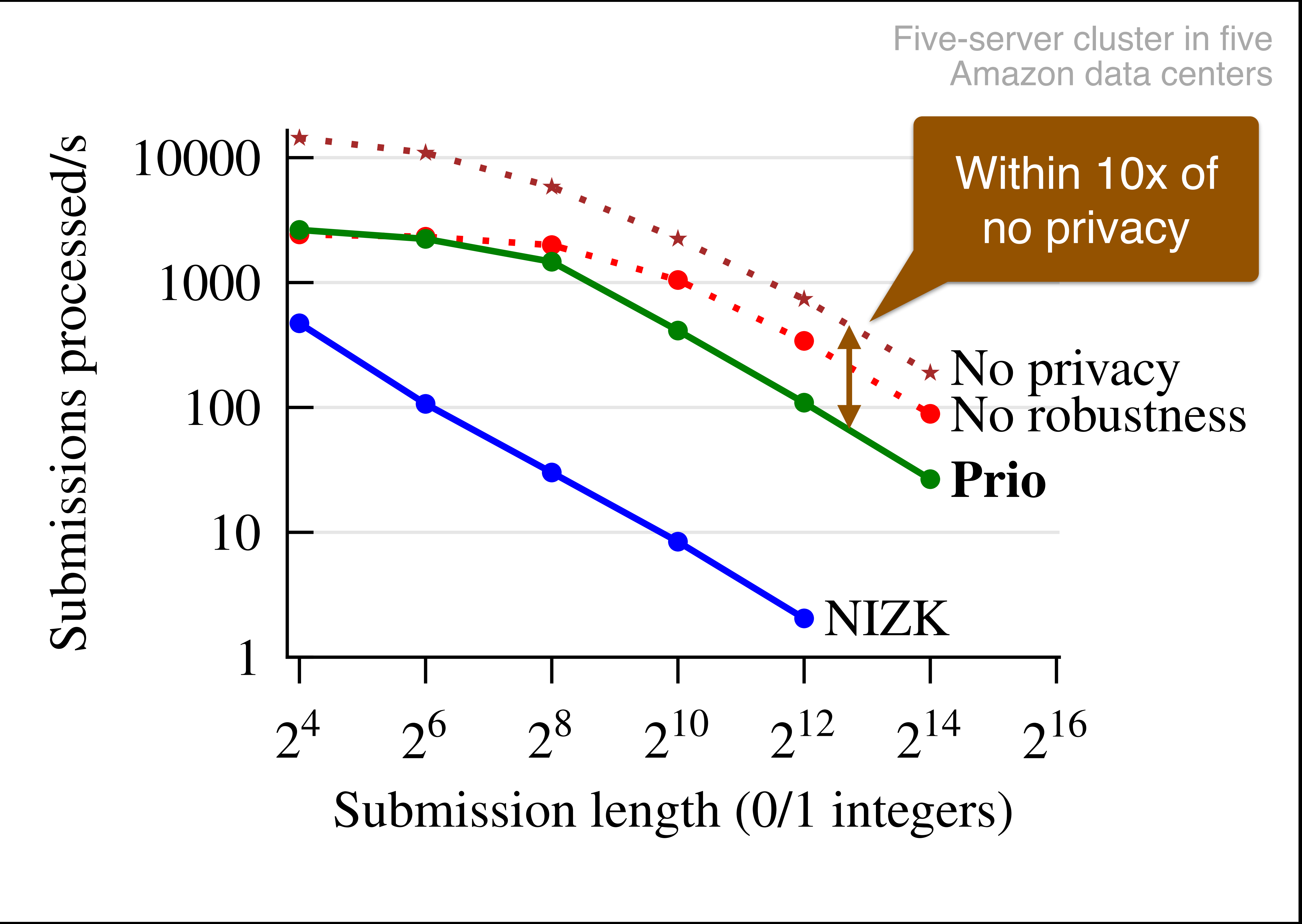

Five-server cluster in five
Amazon data centers
Submissions processed/s
10000
1000
100
10
1
50x performance
improvement
Prio
NIZK
2^4
2^6
2^8
2^{10}
2^{12}
2^{14}
2^{16}
Submission length (0/1 integers)



Five-server cluster in five
Amazon data centers
Submissions processed/s
10000
1000
100
10
1
No robustness
Prio
NIZK
2^4
2^6
2^8
2^{10}
2^{12}
2^{14}
2^{16}
Submission length (0/1 integers)



Five-server cluster in five
Amazon data centers
Submissions processed/s
10000
1000
100
10
1
2^4
2^6
2^8
2^{10}
2^{12}
2^{14}
2^{16}
Submission length (0/1 integers)
No privacy
No robustness
Prio
NIZK



Five-server cluster in five
Amazon data centers
Within 10x of
no privacy
Submissions processed/s
10000
1000
100
10
1
No privacy
No robustness
Prio
NIZK
2^4
2^6
2^8
2^{10}
2^{12}
2^{14}
2^{16}
Submission length (0/1 integers)

Outline

- Background: The private aggregation problem
- A straw-man solution for private sums
- Providing robustness with SNIPs
- **Evaluation**
- Discussion: Real-world considerations

Outline

- Background: The private aggregation problem
- A straw-man solution for private sums
- Providing robustness with SNIPs
- Evaluation
- **Discussion: Real-world considerations**

Real-world considerations: Technical

Every company we spoke with said:

- Server resources are cheap, client resources are not
- Client <u>bandwidth usage</u> is the important quantity to minimize
- Need some defense against faulty/disruptive clients
- Privately collecting popular URLs is the interesting application
  - Existing solutions are good, but not great

Areas of vehement disagreement between companies:

- Non-colluding servers—realistic?
- Does SGX obviate the need for these cryptographic protocols?

Real-world considerations: Technical

Every company we spoke with said:

- Server resources are cheap, client resources are not
- Client <u>bandwidth usage</u> is the important quantity to minimize
- Need some defense against faulty/disruptive clients
- Privately collecting popular URLs is the interesting application
  - Existing solutions are good, but not great

Areas of vehement disagreement between companies:

- Non-colluding servers—realistic?
- Does SGX obviate the need for these cryptographic protocols?

Real-world considerations: Non-technical

Real-world considerations: Non-technical

“If we’re already collecting __________ without privacy protections, why bother adding privacy protections?”

- Pitch: Collect new statistics that you couldn’t collect before

“We don’t yet know what aggregates we want to collect.”

- Pitch: It’s possible to retain some flexibility
  (e.g., can later break out statistics by geographic area)

Real-world considerations: Non-technical

“If we’re already collecting _________ without privacy protections, why bother adding privacy protections?”

- Pitch: Collect new statistics that you couldn’t collect before

“We don’t yet know what aggregates we want to collect.”

- Pitch: It’s possible to retain some flexibility
  (e.g., can later break out statistics by geographic area)

Real-world considerations: Non-technical

“If we’re already collecting _________ without privacy protections, why bother adding privacy protections?”

- Pitch: Collect new statistics that you couldn’t collect before

“We don’t yet know what aggregates we want to collect.”

- Pitch: It’s possible to retain some flexibility
  (e.g., can later break out statistics by geographic area)

“Please sign this stack of non-disclosure agreements…

Real-world considerations: Non-technical

“If we’re already collecting __________ without privacy protections, why bother adding privacy protections?”

- Pitch: Collect new statistics that you couldn’t collect before

“We don’t yet know what aggregates we want to collect.”

- Pitch: It’s possible to retain some flexibility
  (e.g., can later break out statistics by geographic area)

“Please sign this stack of non-disclosure agreements…
and before we show you the non-disclosure agreements…

Real-world considerations: Non-technical

“If we’re already collecting __________ without privacy protections, why bother adding privacy protections?”

– Pitch: Collect new statistics that you couldn’t collect before

“We don’t yet know what aggregates we want to collect.”

– Pitch: It’s possible to retain some flexibility
(e.g., can later break out statistics by geographic area)

*“Please sign this stack of non-disclosure agreements…
and before we show you the non-disclosure agreements…
you have to sign a preliminary non-disclosure agreement.”*

Real-world considerations: Non-technical

“If we’re already collecting __________ without privacy protections, why bother adding privacy protections?”

- Pitch: Collect new statistics that you couldn’t collect before

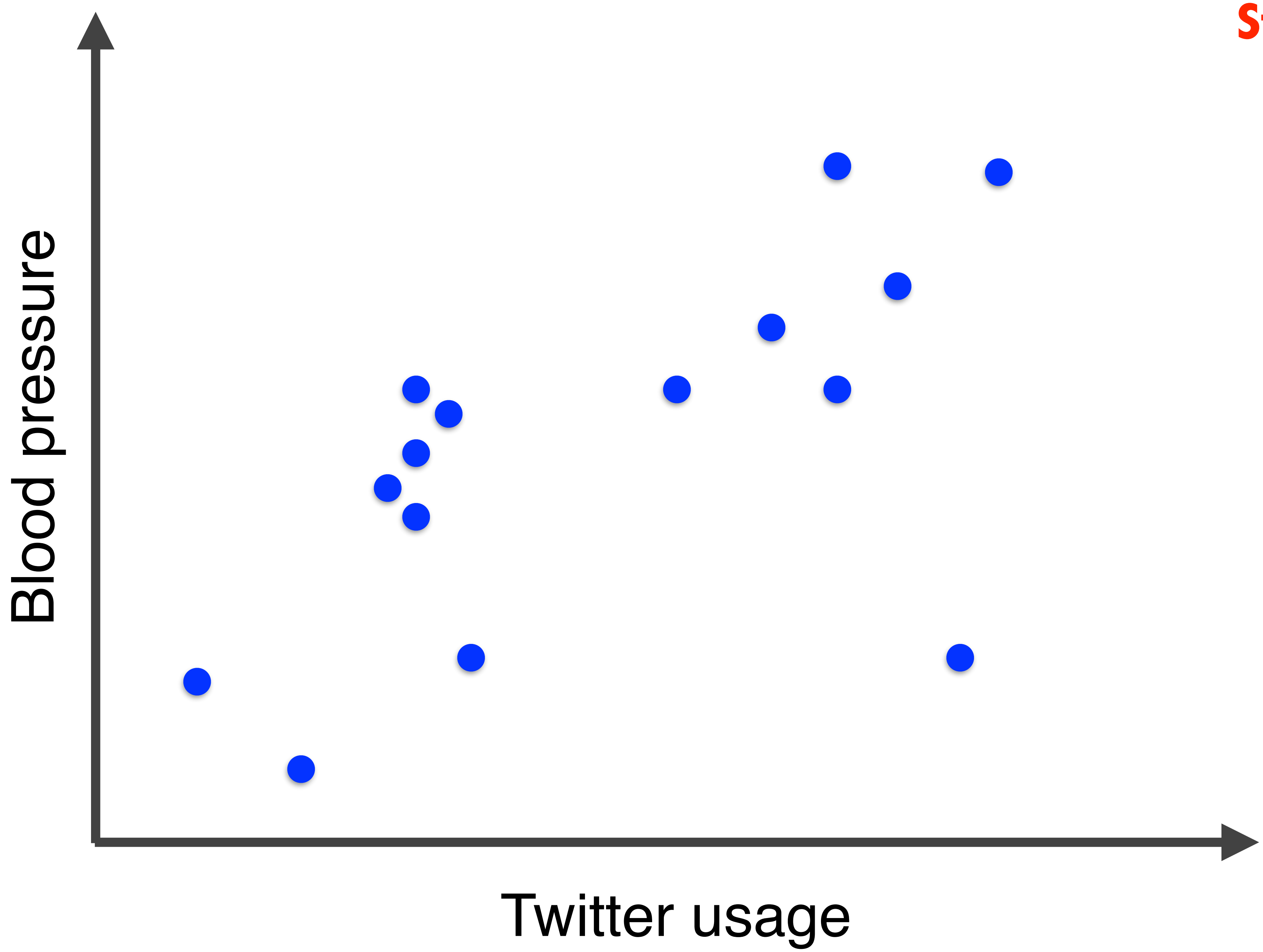
“We don’t yet know what aggregates we want to collect.”

- Pitch: It’s possible to retain some flexibility
  (e.g., can later break out statistics by geographic area)

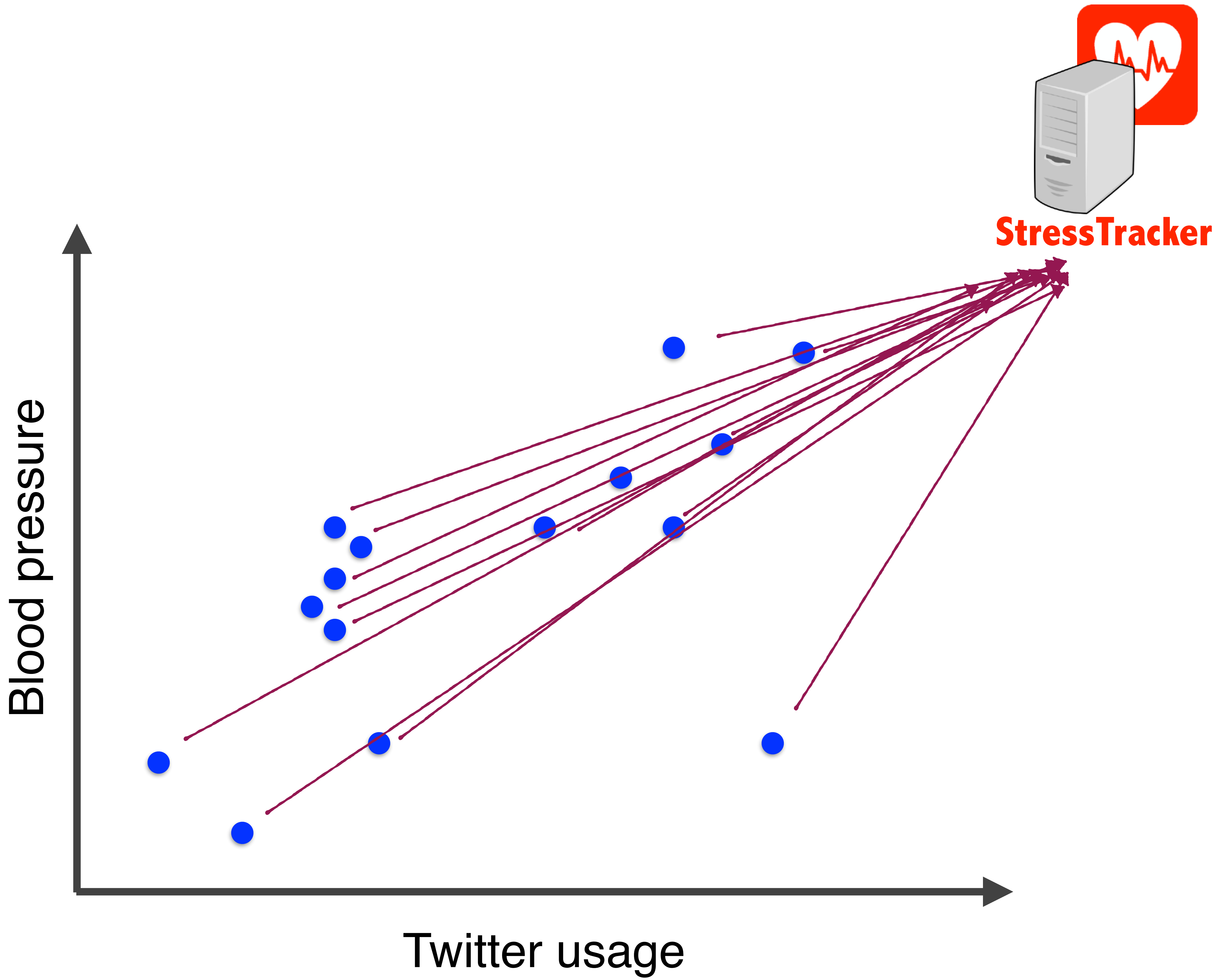
“Please sign this stack of non-disclosure agreements…
and before we show you the non-disclosure agreements…
you have to sign a preliminary non-disclosure agreement.”

- **???**

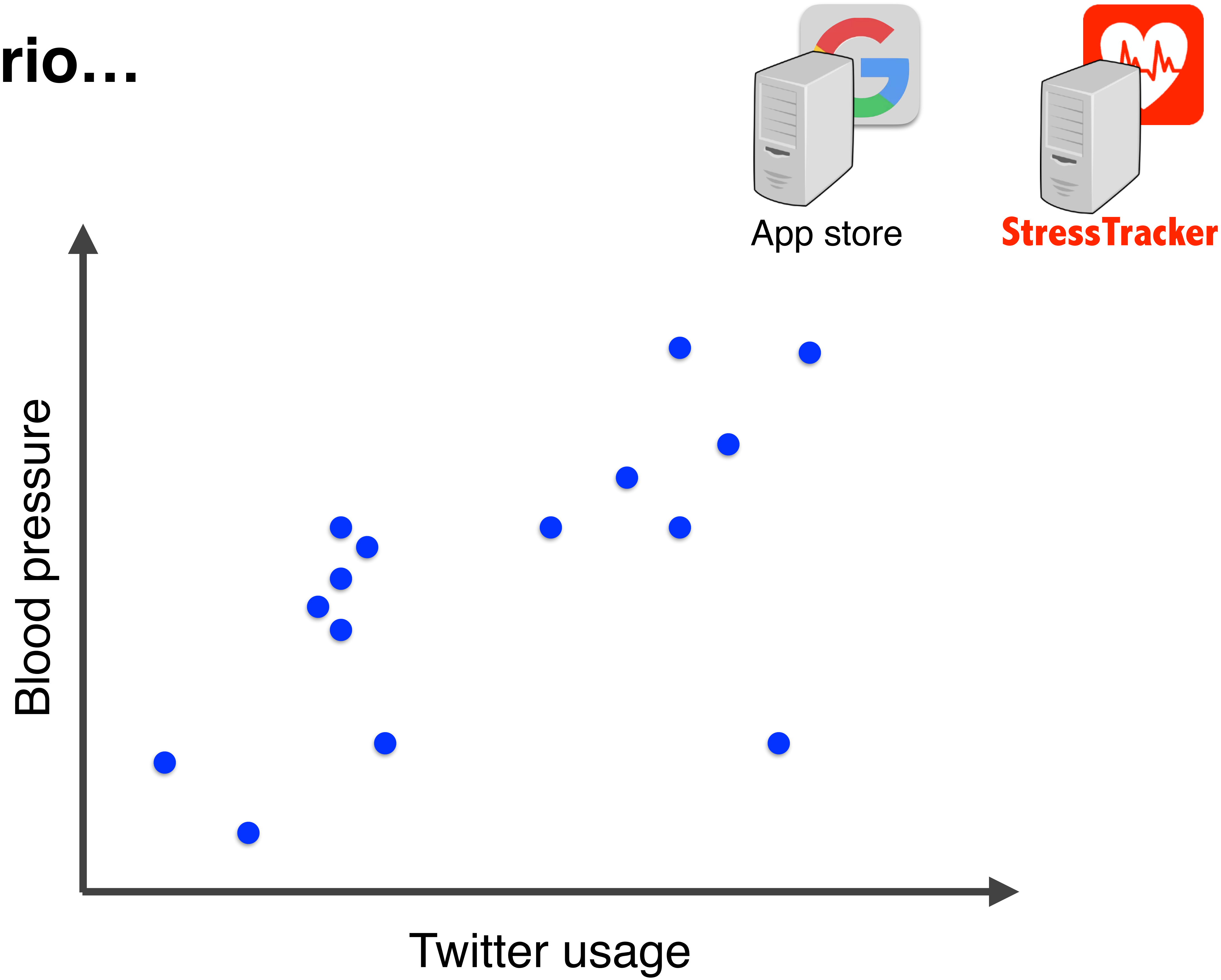
Today



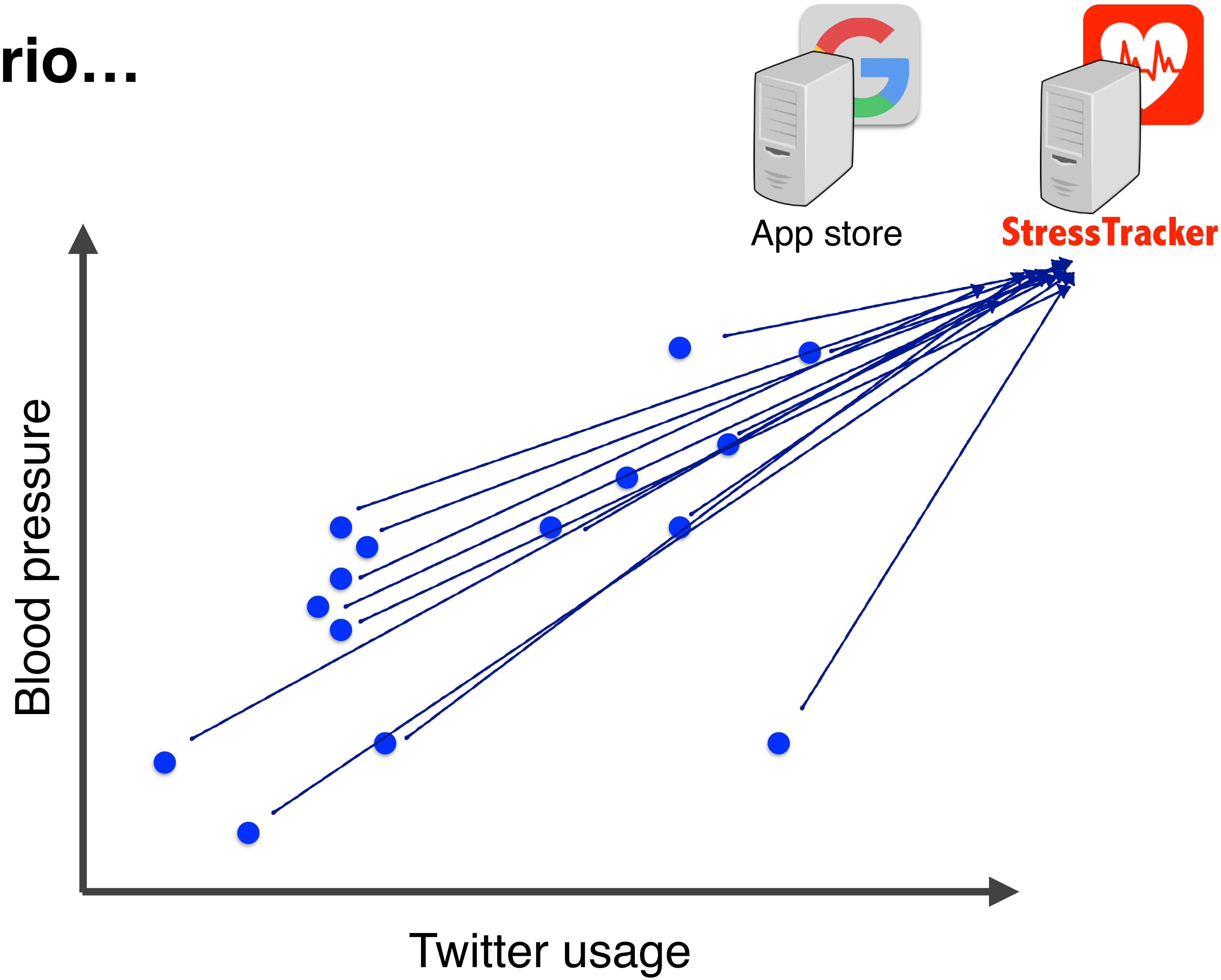
Today



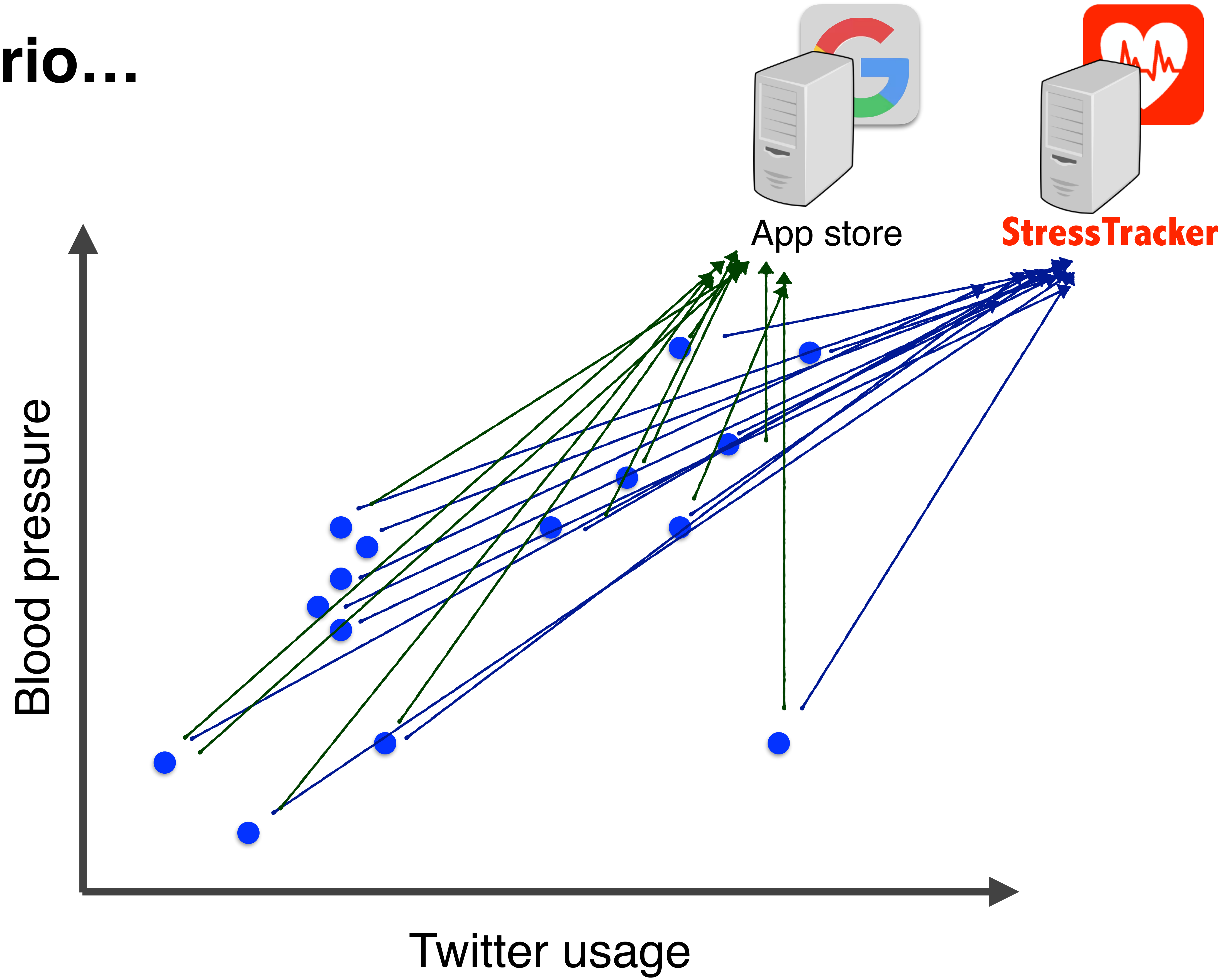
With Prio…



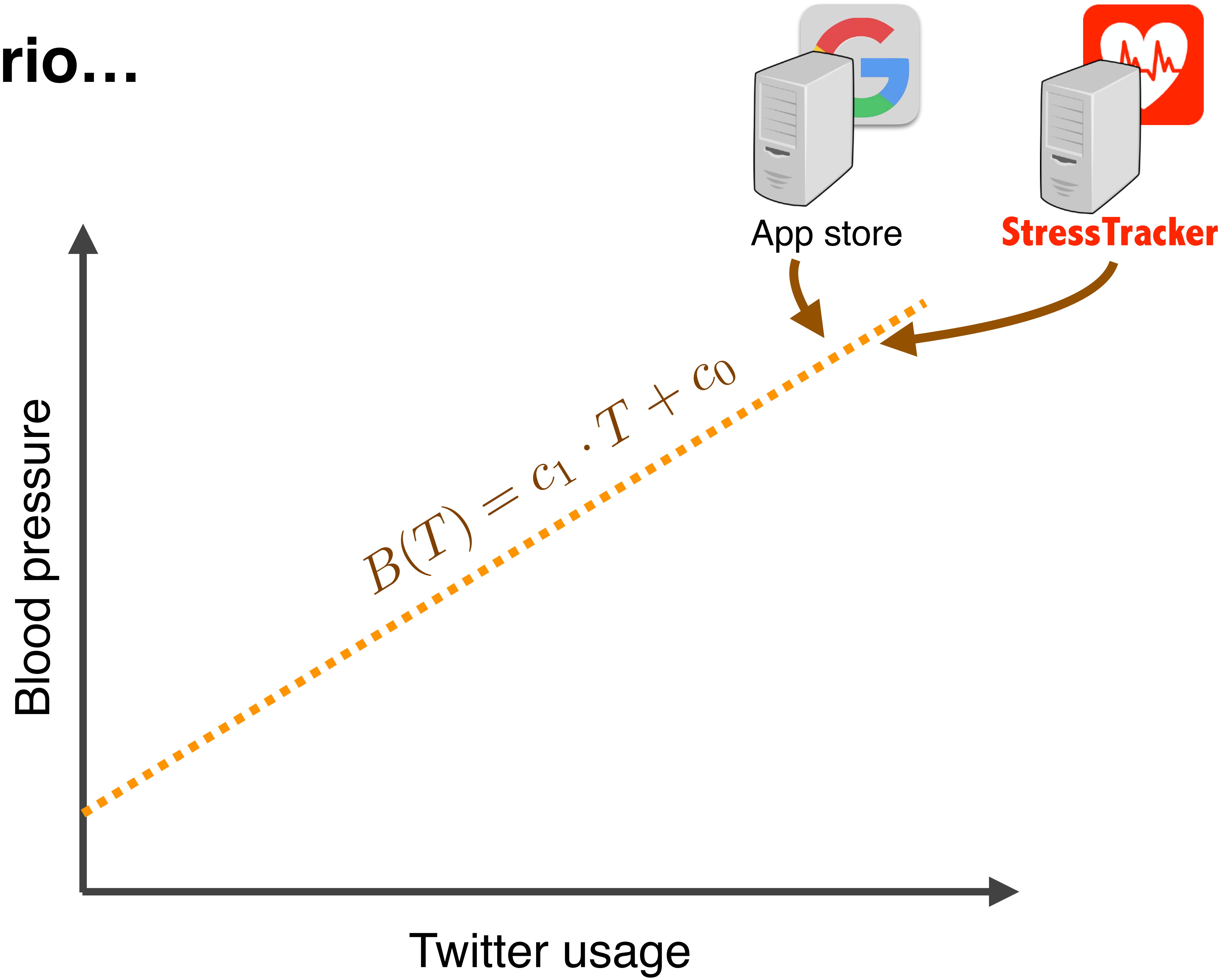
With Prio…



With Prio…



With Prio…



App store
StressTracker
$B(T) = c_1 \cdot T + c_0$
Blood pressure
Twitter usage

Conclusions

- Wholesale collection of sensitive user data puts our security at risk.
- Prio is the first system for aggregation that provides:
  - **exact correctness**,
  - **privacy**,
  - **robustness**, and
  - **efficiency**.
- To do so, Prio uses SNIPs and aggregatable encodings.
- These techniques together bring private aggregation closer to practical.

Thank you!

Henry Corrigan-Gibbs
henrycg@cs.stanford.edu

https://crypto.stanford.edu/prio/