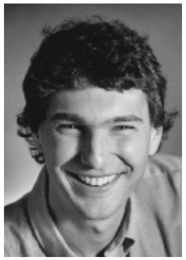
SiTP Dec. 2017

Cryptography for IoT

Dan Boneh
Stanford University

… but first: Computer Security at Stanford



Alex Aiken
software analysis

Dan Boneh
applied Crypto,
crypto currencies

Matei Zaharia
security and big data

Dawson Engler
automated
bug finding

David Mazières
Op. Systems

Phil Levis
IoT Security

John Mitchell
protocol design,
online ed.

Mendel Rosenblum
VM's in security

Courses

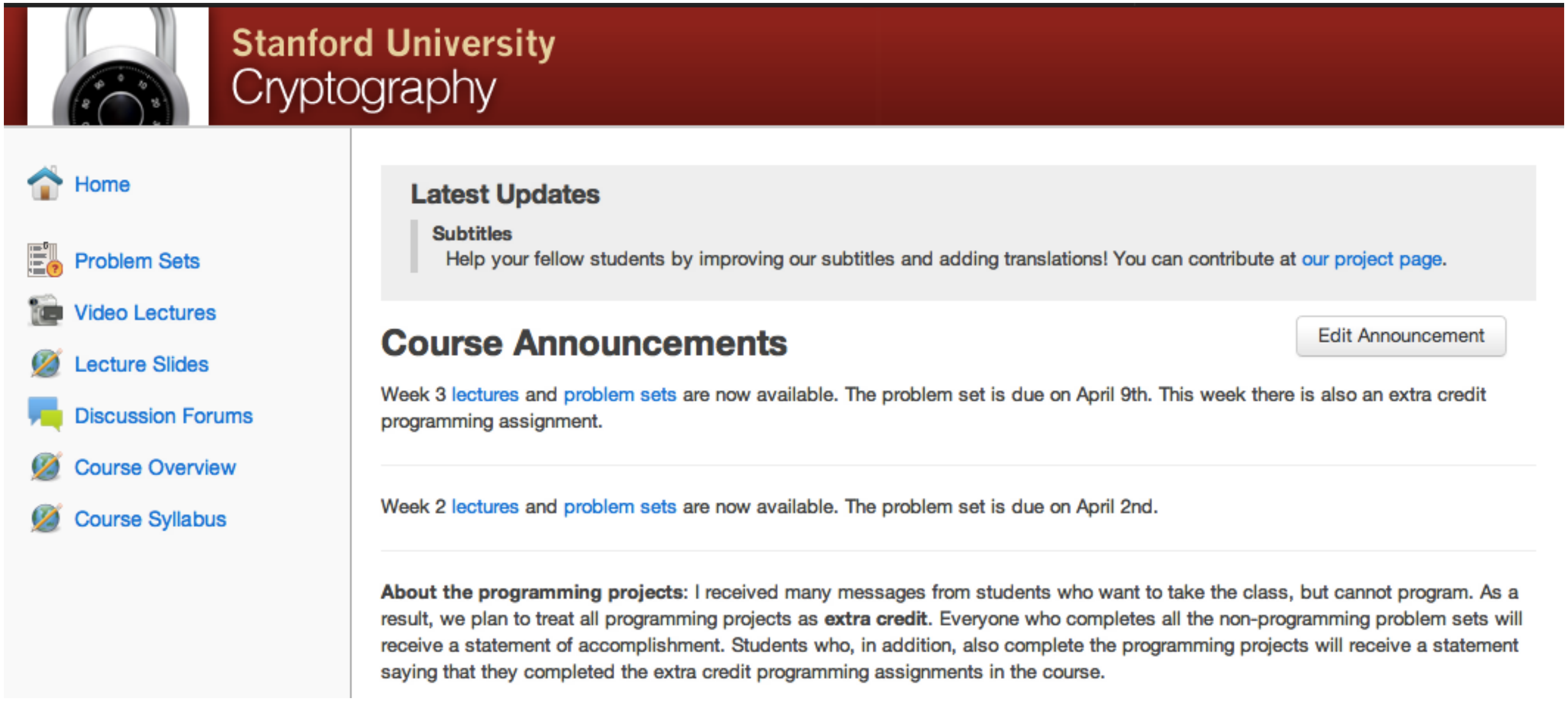
- **Courses**:
  - CS55N (freshmen seminar): ten ideas in computer security
  - **CS155: Computer Security**
  - **CS251: Crypto currencies and blockchain technologies**
  - **CS255: Intro to Crypto**
  - CS259: Security analysis of network protocols
  - CS355: Graduate course in cryptography

- Stanford Advanced Computer Security Certificate

  http://scpd.stanford.edu/computerSecurity/

Online Courses

//www.coursera.org/learn/crypto



Course open to the public

Free Book Draft

A Graduate Course in Applied Cryptography

Dan Boneh and Victor Shoup

Free at: //cryptobook.us

Please send us comments

Multiparty computation (MPC) and SGX

MPC for genomic data analysis

[Jagadeesh, Wu, Birgmeier, Boneh, Bejerano, *Science* 2017]

What genes causes a specific disorder?

$$\begin{array}{r c}
v_1: & \multirow{4}{*}{} \\
 & \\
v_3: & \\
 &
\end{array}
\begin{bmatrix}
0 & \mathbf{1} & 0 & \mathbf{2} & 0 & \mathbf{1} \\
\mathbf{1} & 0 & \mathbf{1} & \mathbf{2} & 0 & \mathbf{1} \\
\mathbf{2} & 0 & 0 & \mathbf{2} & \mathbf{1} & \mathbf{1} \\
0 & 0 & \mathbf{1} & \mathbf{2} & 0 & \mathbf{1}
\end{bmatrix}$$

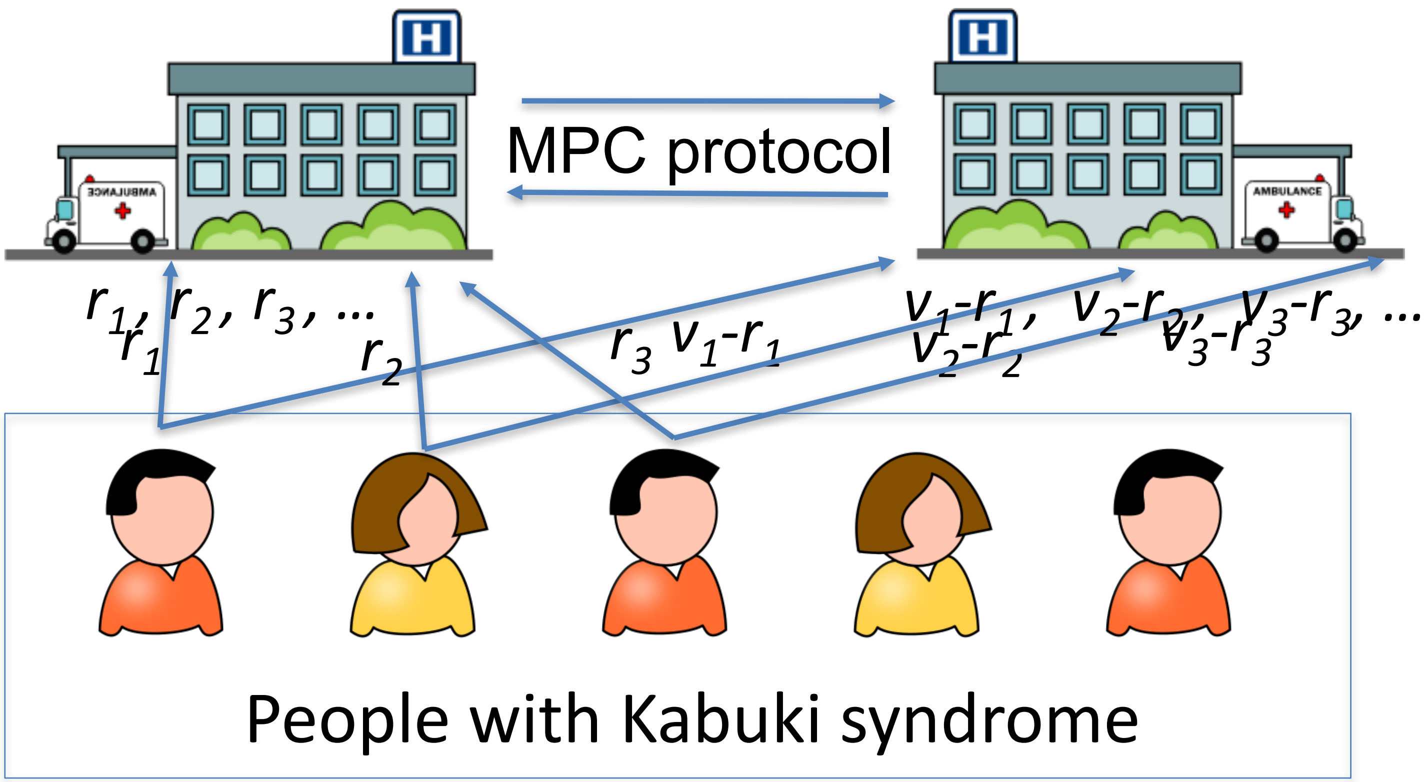
People with Kabuki syndrome

Each has 211 to 374 rare genes out of ≈20,000 genes

Patient i: vector v_i of dim 20,000 that is 0 for normal genes

MPC for genomic data analysis

[Jagadeesh, Wu, Birgmeier, Boneh, Bejerano, 2017]

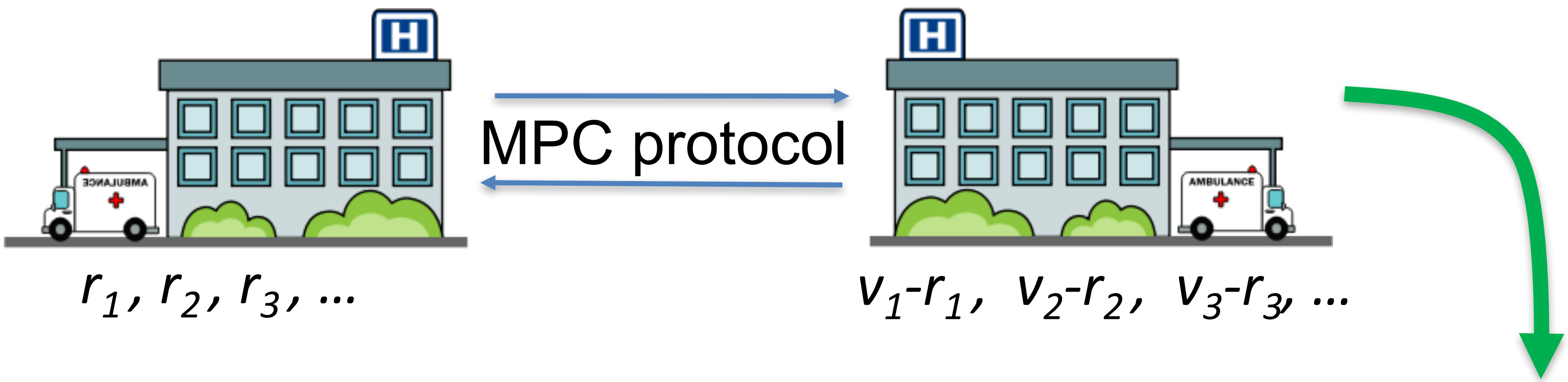


Each has 211 to 374 rare genes out of ≈20,000 genes

Patient i: vector v_i of dim 20,000 that is 0 for normal genes

MPC for genomic data analysis

[Jagadeesh, Wu, Birgmeier, Boneh, Bejerano, 2017]



People with Kabuki syndrome

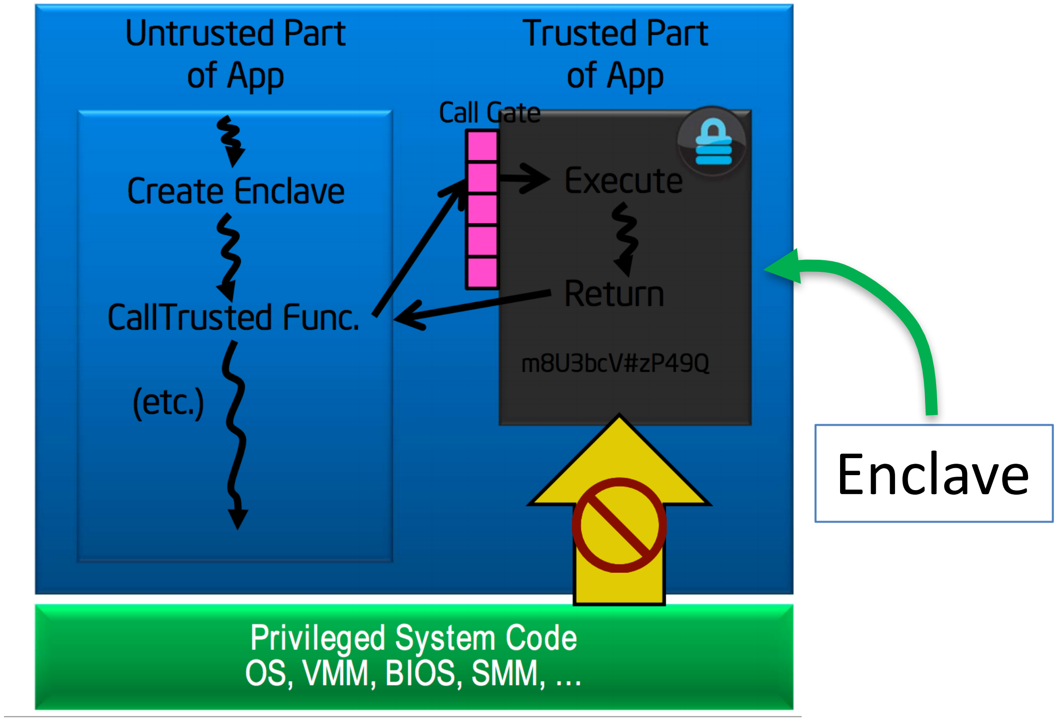
most common rare genes

KMT2D, COL6A1

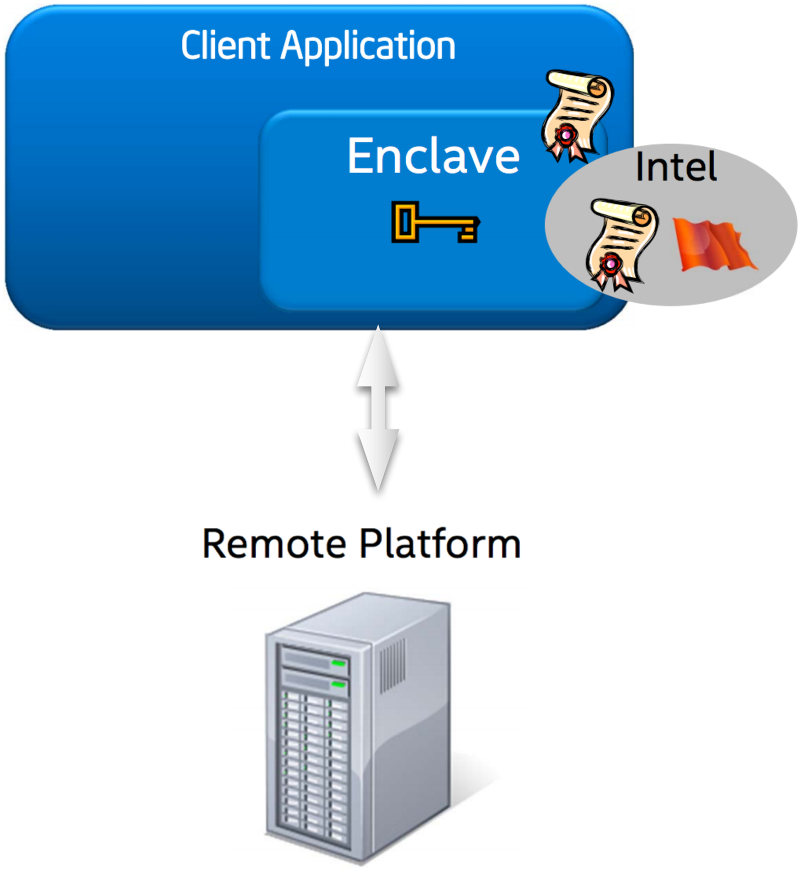
Nothing else is revealed about the individual genomes !!

Can we do this with Intel's SGX ?

Enclave Application



Remote Attestation



Source: ISCA 2015 tutorial slides for Intel SGX

Iron: Functional encryption and obfuscation using Intel SGX

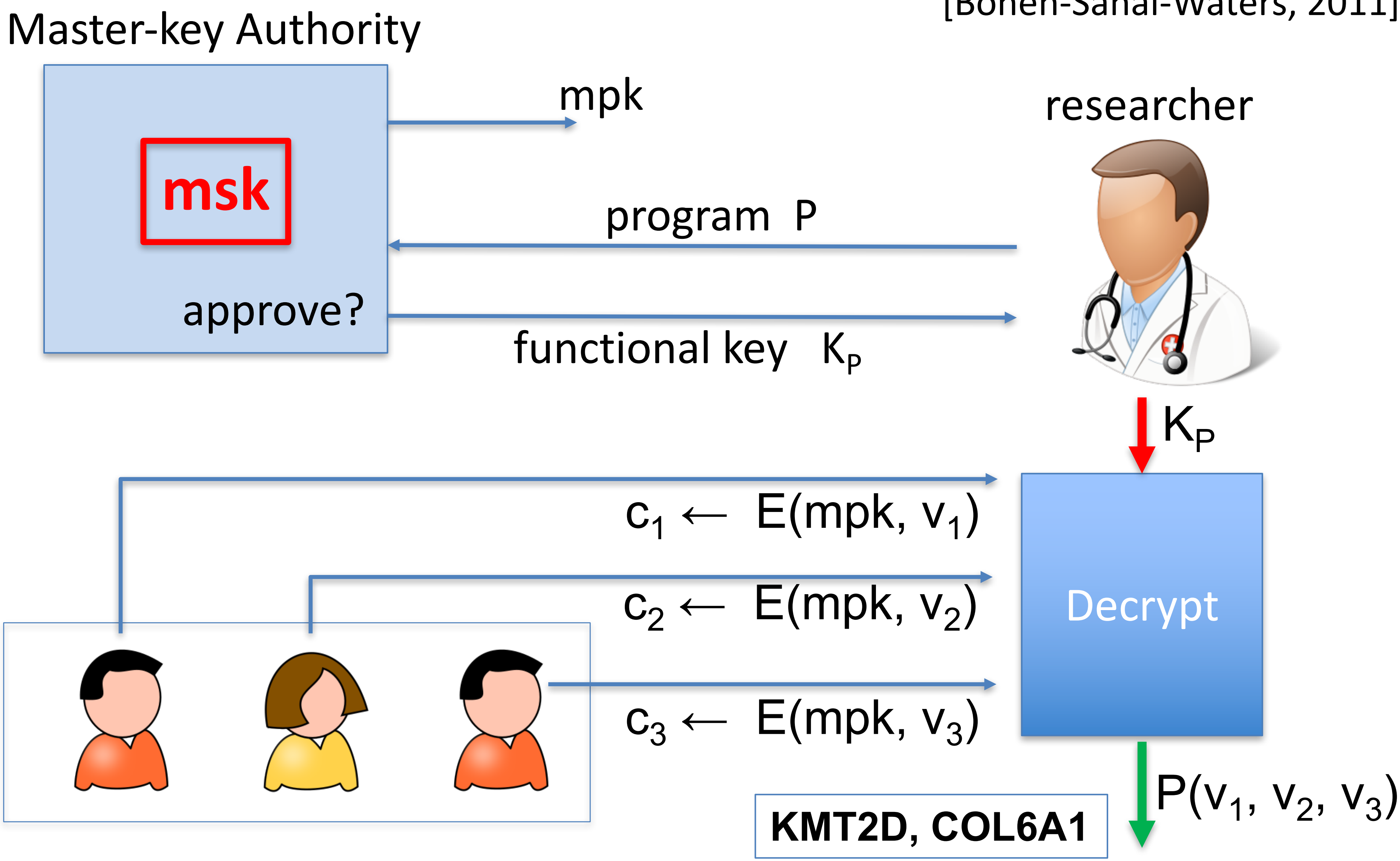
Ben Fisch, Dhinakaran Vinayagamurthy,

Dan Boneh, Sergey Gorbunov

In proc. ACM CCS 2017

Functional Encryption

[Boneh-Sahai-Waters, 2011]



Functional Encryption

[Boneh-Sahai-Waters, 2011]

Why is functional encryption hard?

no interaction during decryption

can't use MPC techniques

researcher

K_P

$c_1 \leftarrow E(mpk, v_1)$

$c_2 \leftarrow E(mpk, v_2)$

$c_3 \leftarrow E(mpk, v_3)$

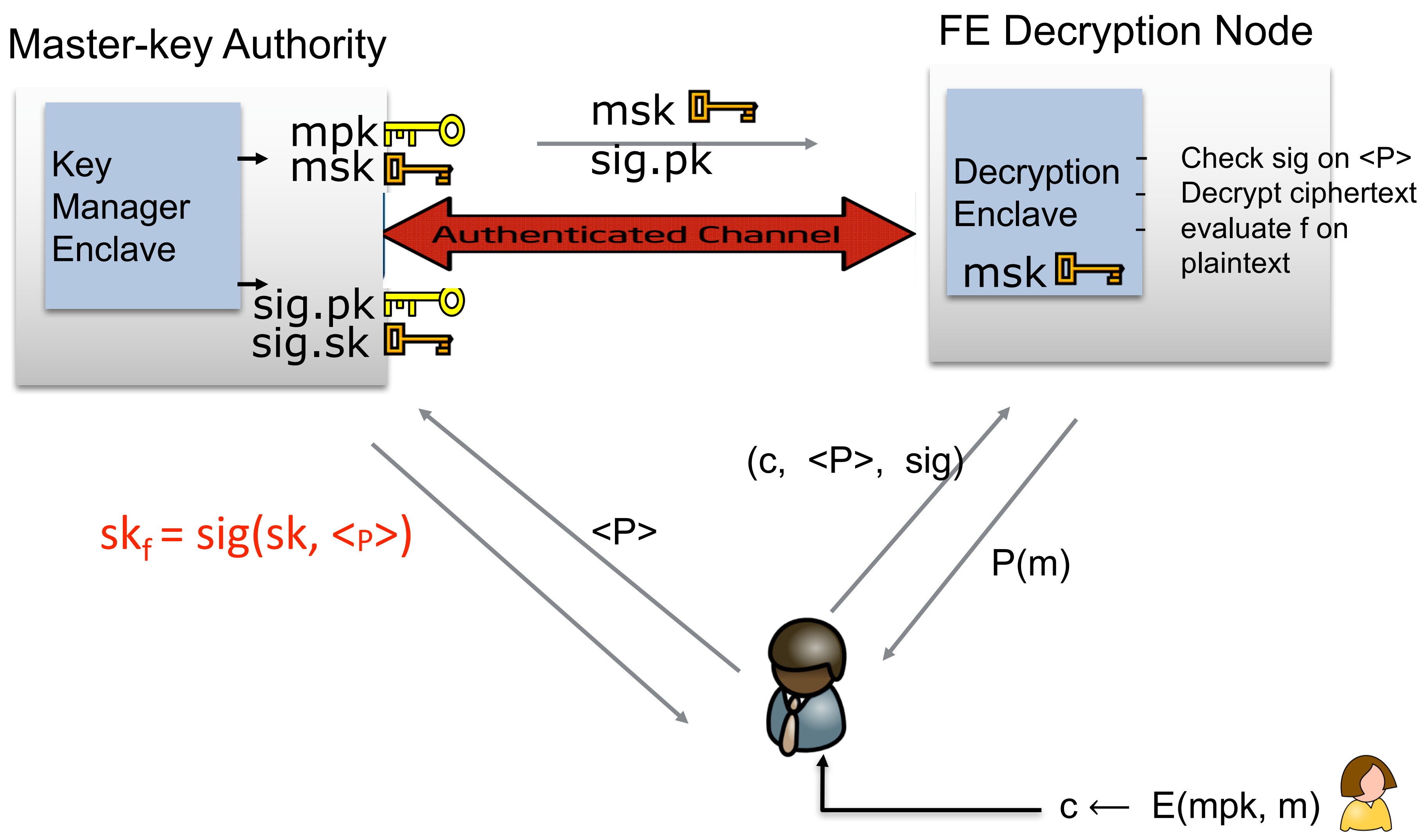
Decrypt

$P(v_1, v_2, v_3)$

Satisfy regulators?
(GDPR)

SGX Functional Encryption: approach

mpk: multi-input func. enc. public key



But not so simple …

- Enclave memory access pattern leaks and can break FE security

- How to represent the program P:
  - Cannot move code into enclave after EINIT
  - Difficult to safely implement interpreter in enclave: performance and memory access pattern leak

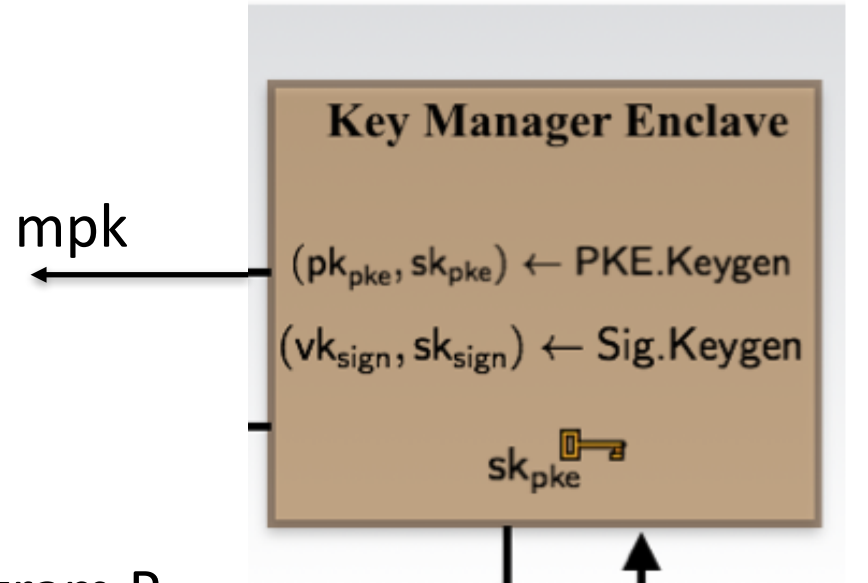
- Side channel attacks (timing, power)

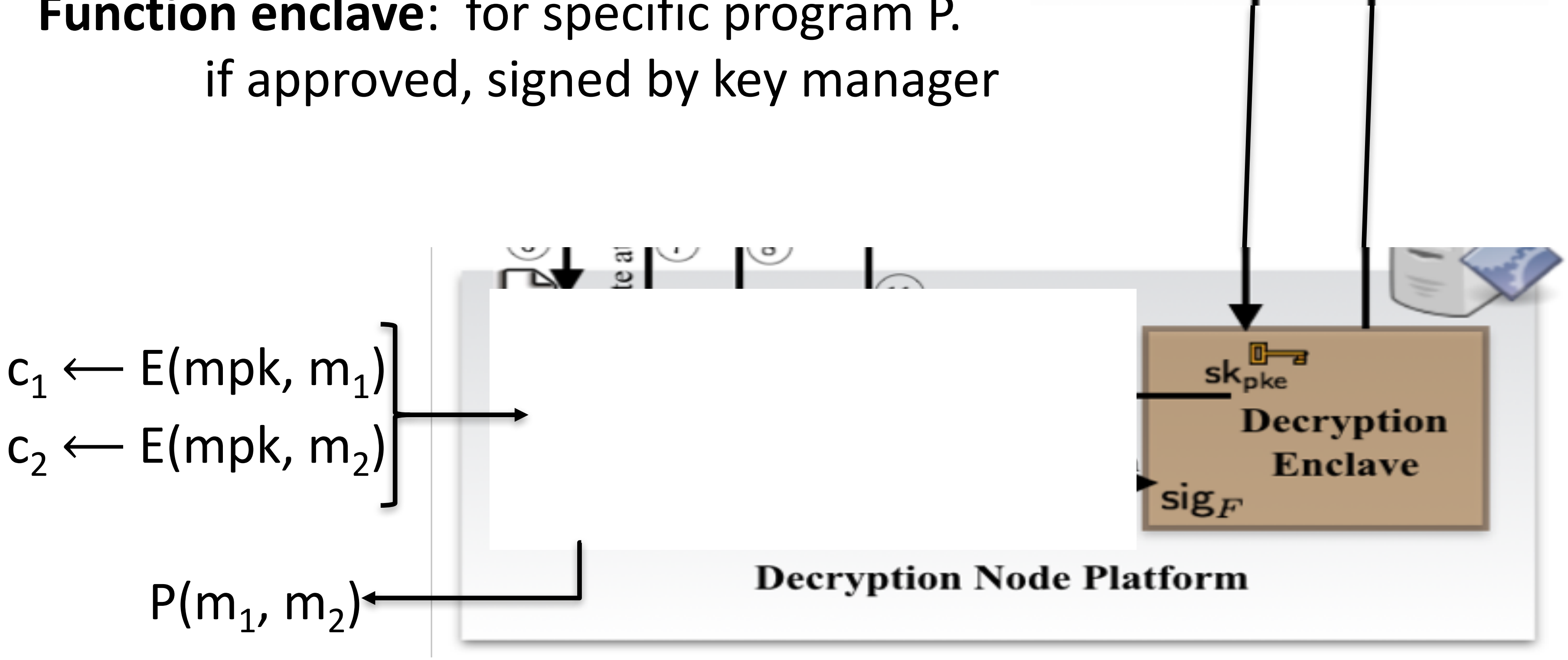
Iron architecture

Key manager enclave:
manage master key

Decryption enclave:
initialized at startup

Function enclave: for specific program P.
if approved, signed by key manager





Security

- Formally model the SGX HW interface:
  Setup, Load, Run, Run&Report, Run&Quote, ReportVerify, QuoteVerify

  Builds on HW security models of:

  Pass et. al. [PST'17] , Bahmani et. al. [BBB+'16]

- MIFE simulation-based security, assuming:
  adversary cannot distinguish
  black-box HW interface and real SGX

Side-channel atacks

Security proof does not capture side channel attacks on SGX

- Cache-timing attacks [CD16] leak memory access patterns at cache-line granularity
- Page-fault attacks [XCP15] leak memory access patterns at 4KB page granularity
- Branch shadowing attacks [LSG+16] can directly view branch history (saved for pipeline branch prediction)

DEFENSE: only sign function enclaves whose memory access pattern is independent of sensitive data
(e.g. ORAM based)

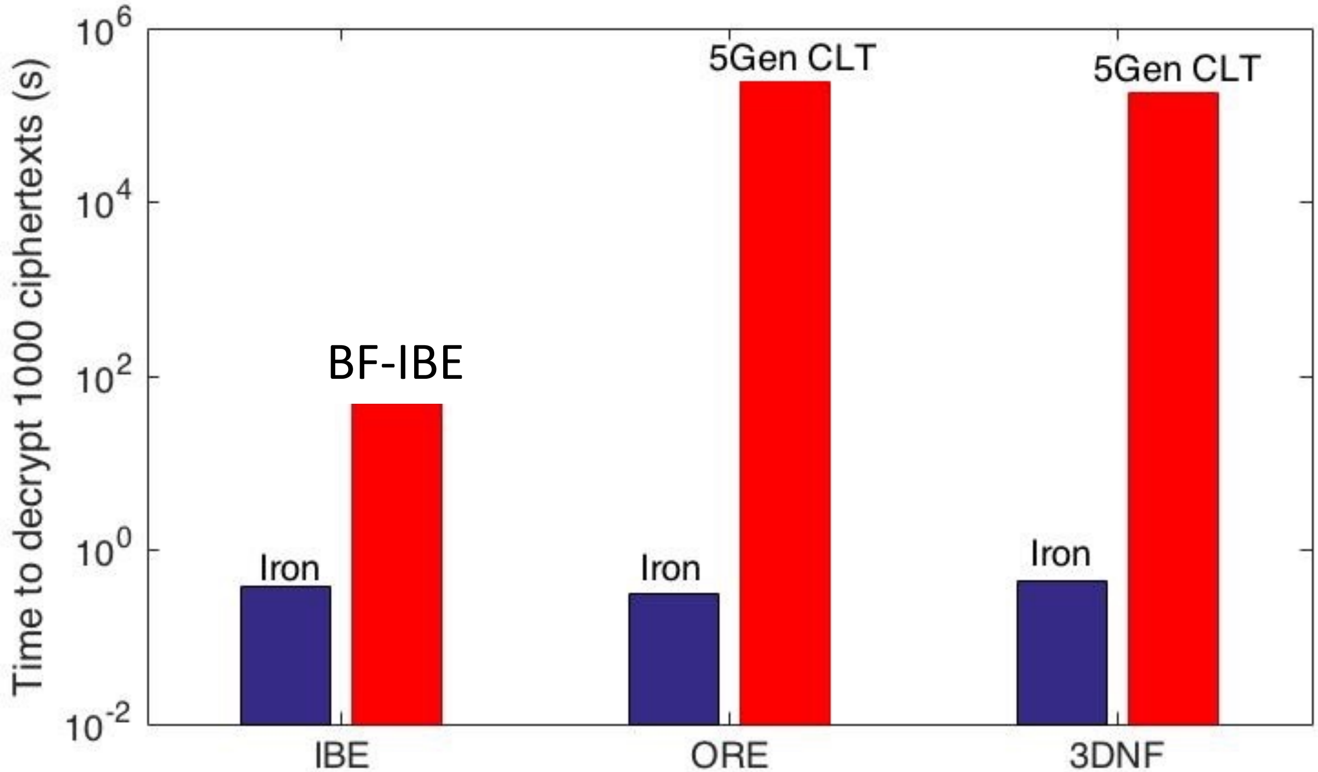
Implementation and Evaluation

- C++ using the Intel(R) SGX SDK 1.6 for Windows

  Intel Skylake i7-6700, 3.40 GHz, 8 GiB RAM, Windows Server 2012 R2 Standard

- Function enclave implementation is *data-oblivious* to resist side-channels

Comparing Iron to cryptographic constructions



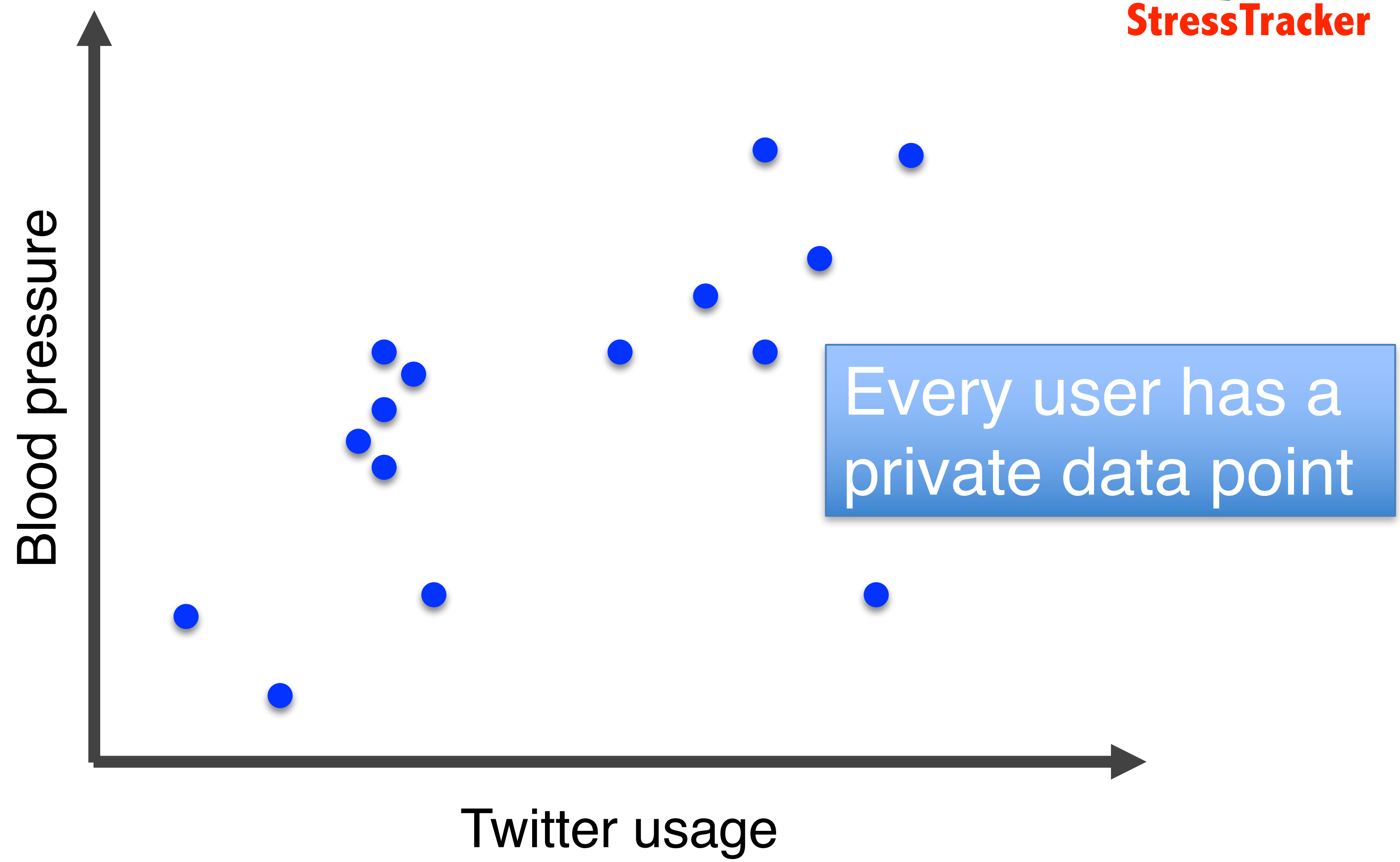
Private data aggregation

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

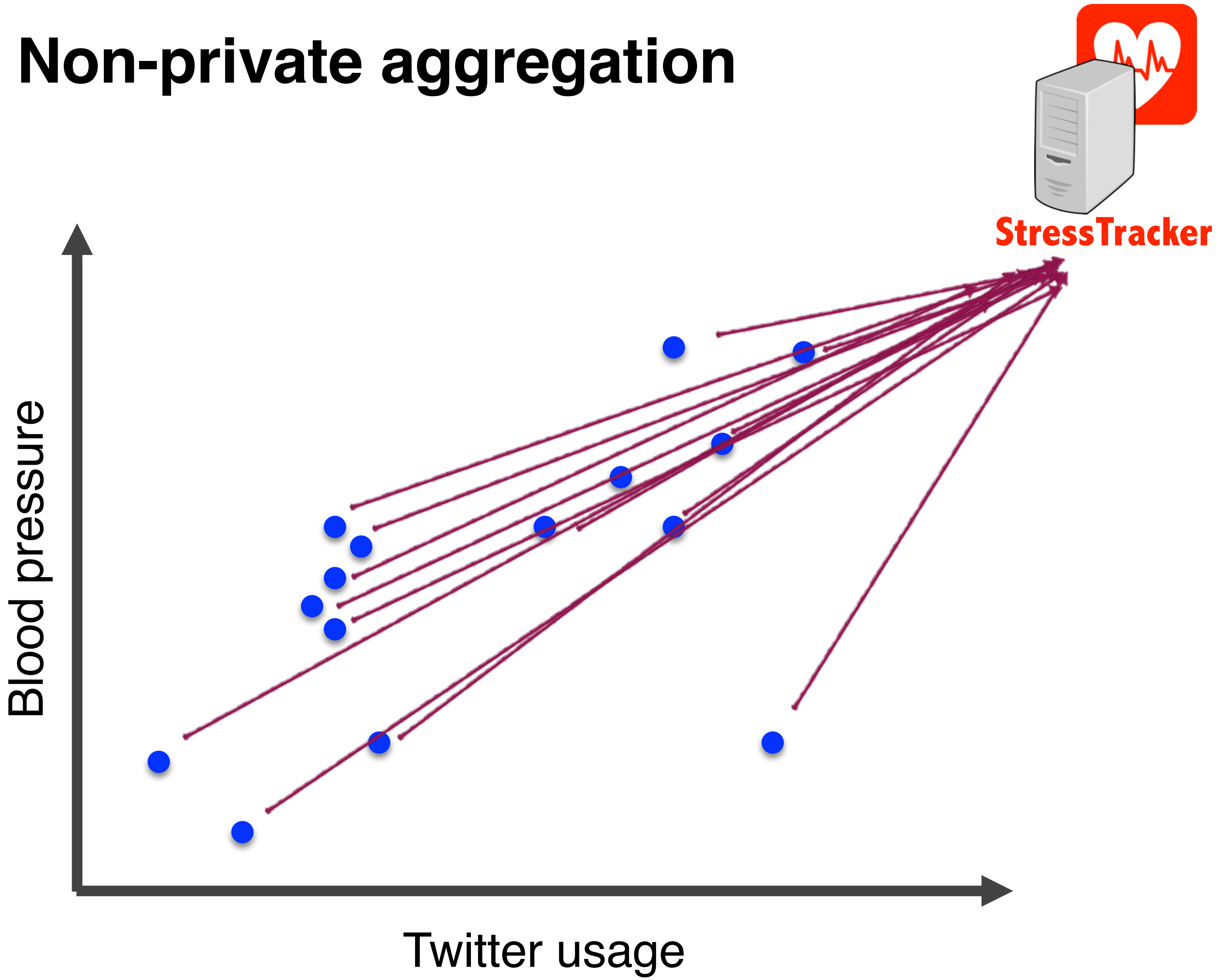
Joint work with Henry Corrigan-Gibbs

NSDI 2017

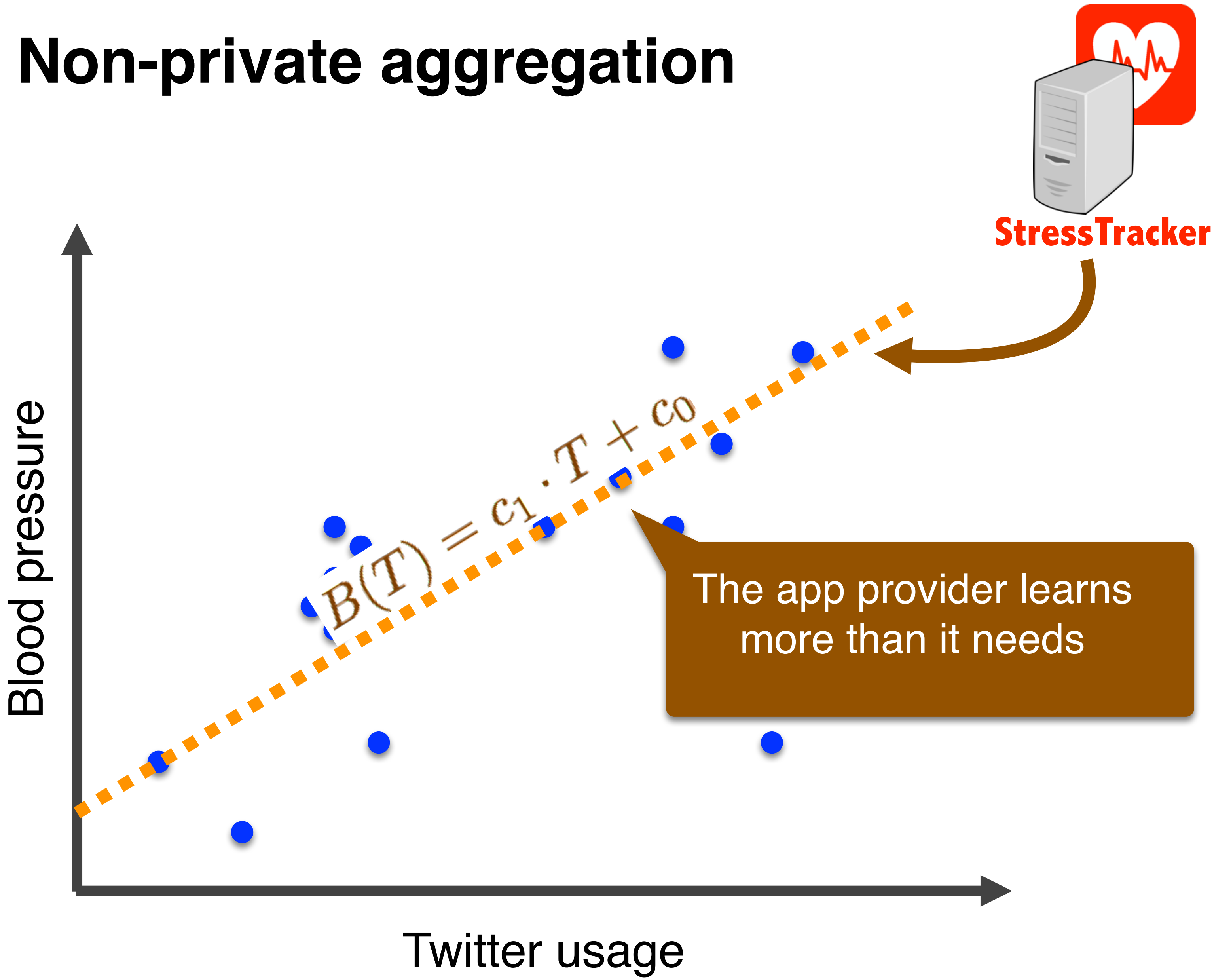
Today: Non-private aggregation



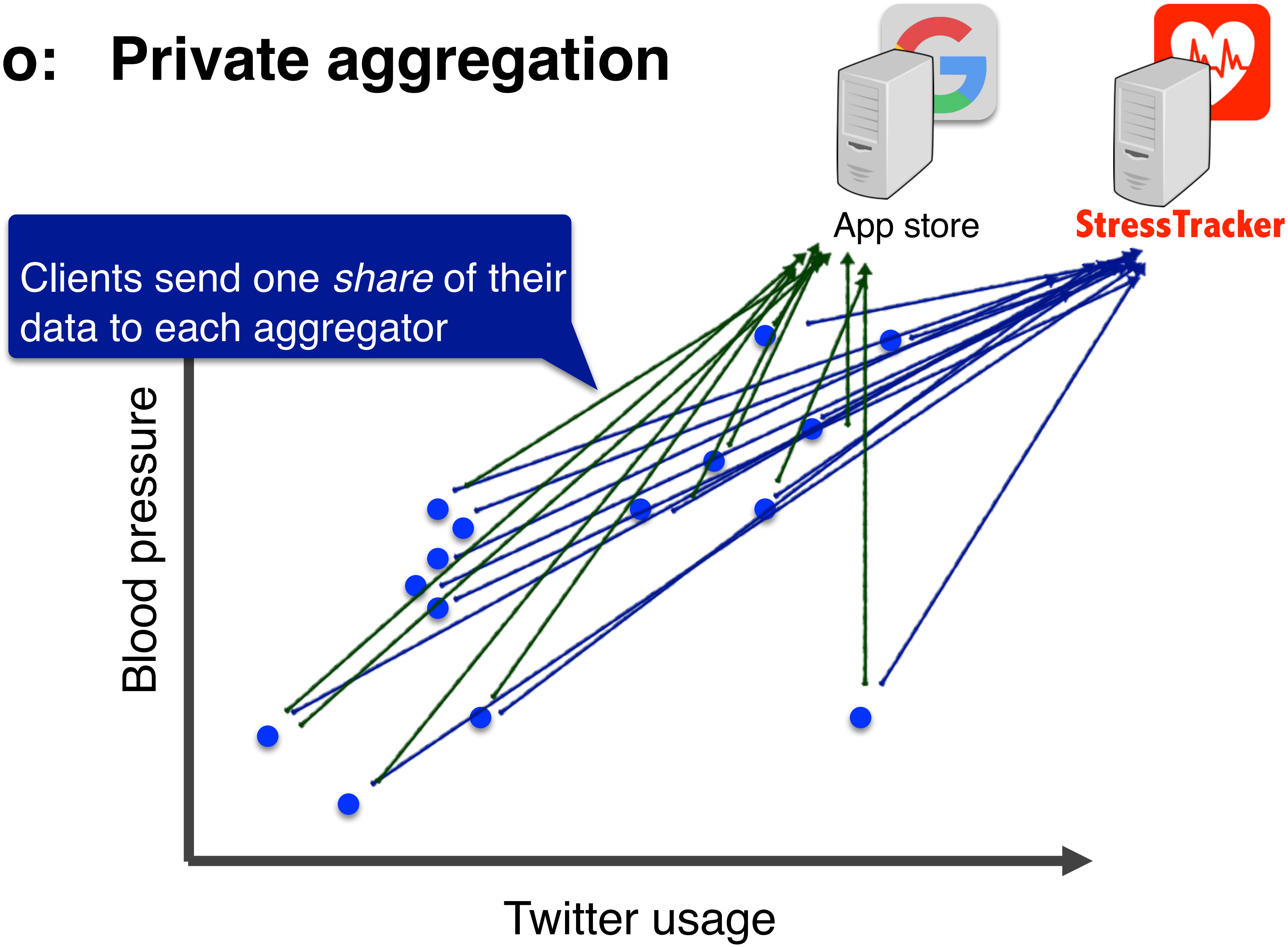
Today: Non-private aggregation



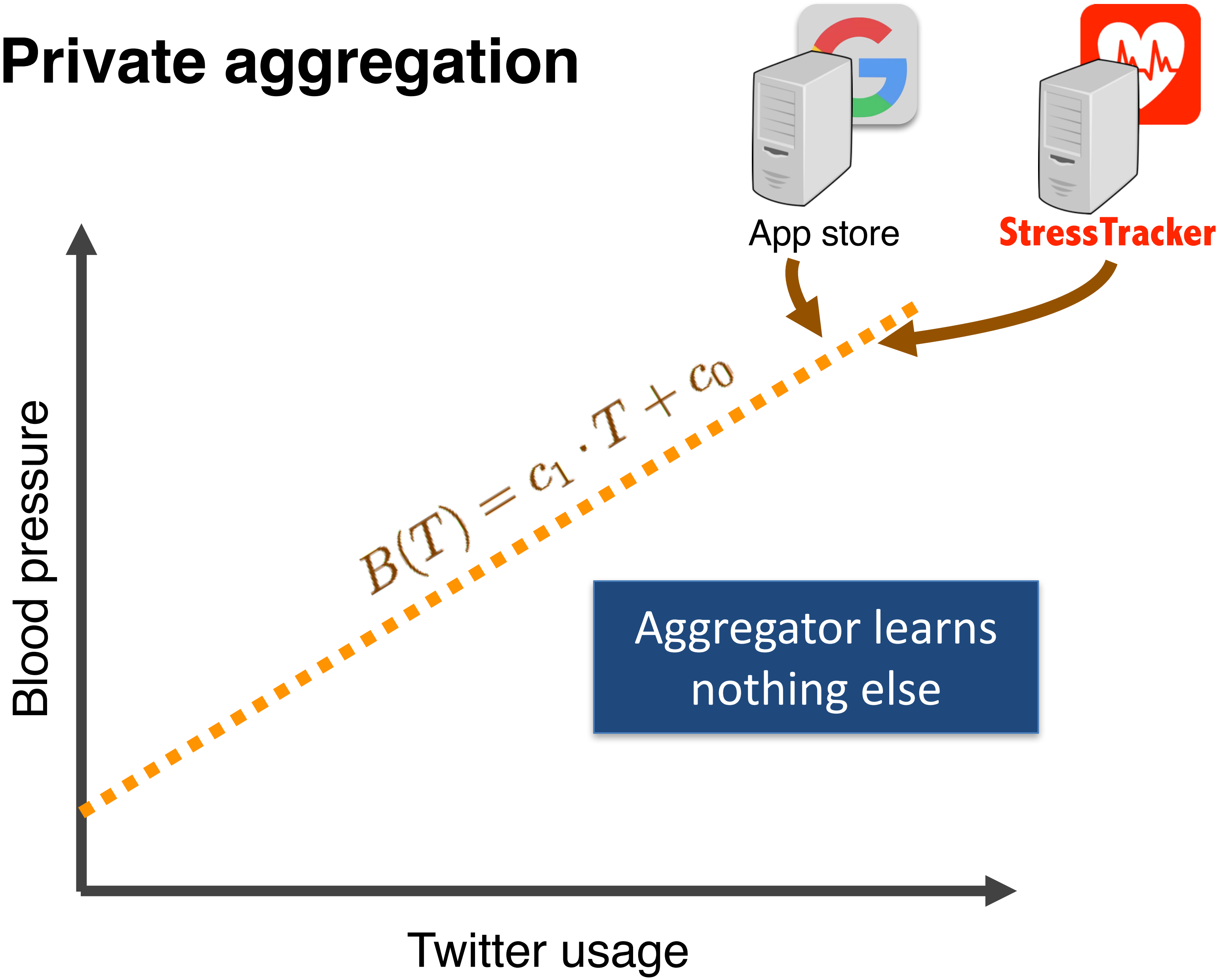
Today: Non-private aggregation



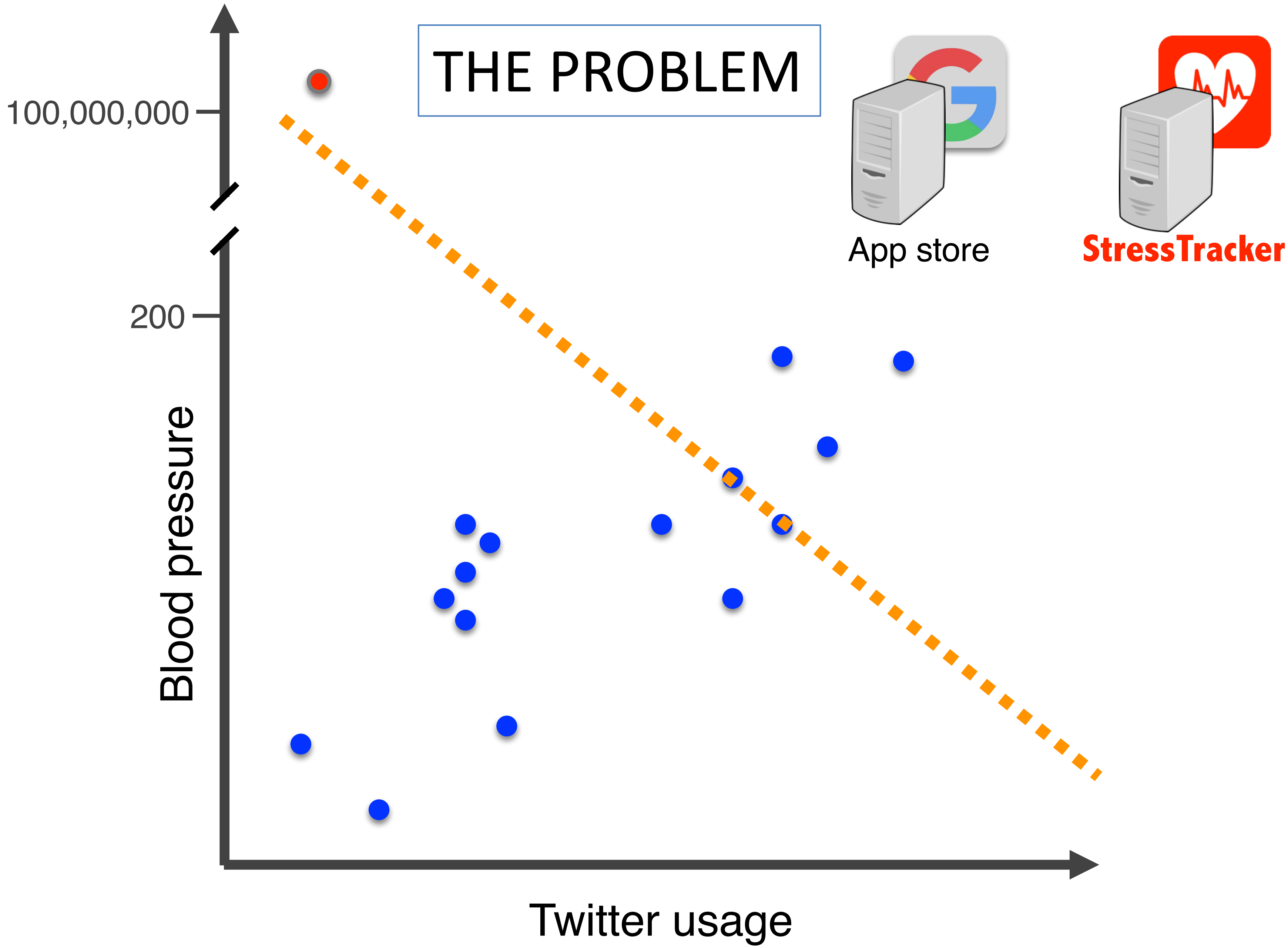
Prio: Private aggregation



Prio: Private aggregation



THE PROBLEM



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

Private aggregation

Exact correctness: if all servers are honest they learn $f(x_1,\ldots,x_n)$

Privacy: if one server is honest they learn <u>only</u> $f(x_1,\ldots,x_n)$

<u>Robustness</u>: malicious clients have bounded influence

Scalable: no public-key crypto (other than TLS)

Prio contributions

Achieves all four goals

1. Robustness using
   secret-shared non-interactive proofs (SNIPs)

   - Every client efficiently proves to servers that its submission is well formed
   - Takes advantage of non-colluding servers (verifiers)

2. Aggregatable encodings

   Compute sums privately $\Longrightarrow$
   compute $f(\cdot)$ privately for many f's of interest

Existing approaches

- 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)

Private aggregation needed in many settings

Private client value (x_i)	**Aggregate** $f(x_1, \ldots, x_N)$
Location data (phones/cars)	• Number of devices in location L • Ten most popular locations • Locations with weakest signal strength
Web browsing history	• Most common bug-triggering websites • Websites with TLS certificate errors
Health information	• Min, max, avg, stddev heart rate • ML model relating BP to Twitter usage
Text messages	• Min, max, average number per day • ML model relating time of day to emotion

Warm-up: Computing private sums

Every device i holds a value $x_i \in \mathbb{F}$

Think: integers modulo a prime p

Cloud wants to compute

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

<u>without</u> learning any users' private value $x_i \in \mathbb{F}$

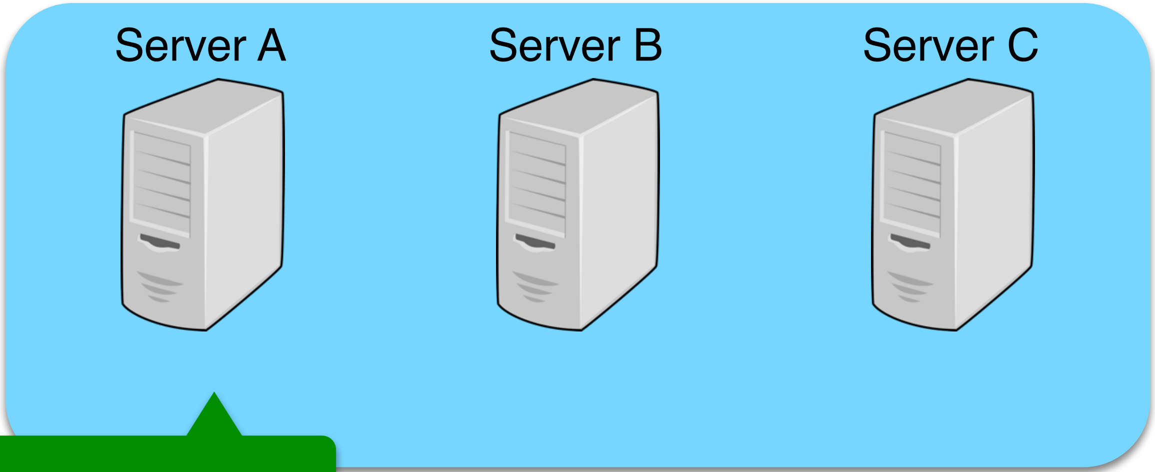
Example: Privately measuring traffic congestion

$$x_i = \begin{cases} 1 & \text{if user } i \text{ is on Golden Gate Bridge} \\ 0 & \text{otherwise} \end{cases}$$

$x_1 + \ldots + x_N$ gives number of users on bridge

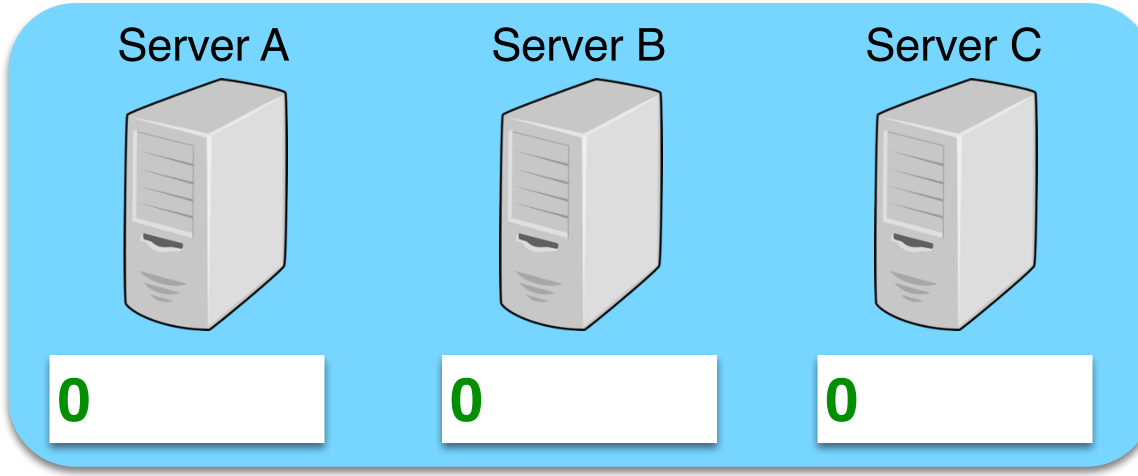
Private sums: A “straw-man” scheme

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



Assume that at least one server is honest.

Private sums:
A “straw-man” scheme



$[x_1]_a \quad [x_1]_b \quad [x_1]_c$

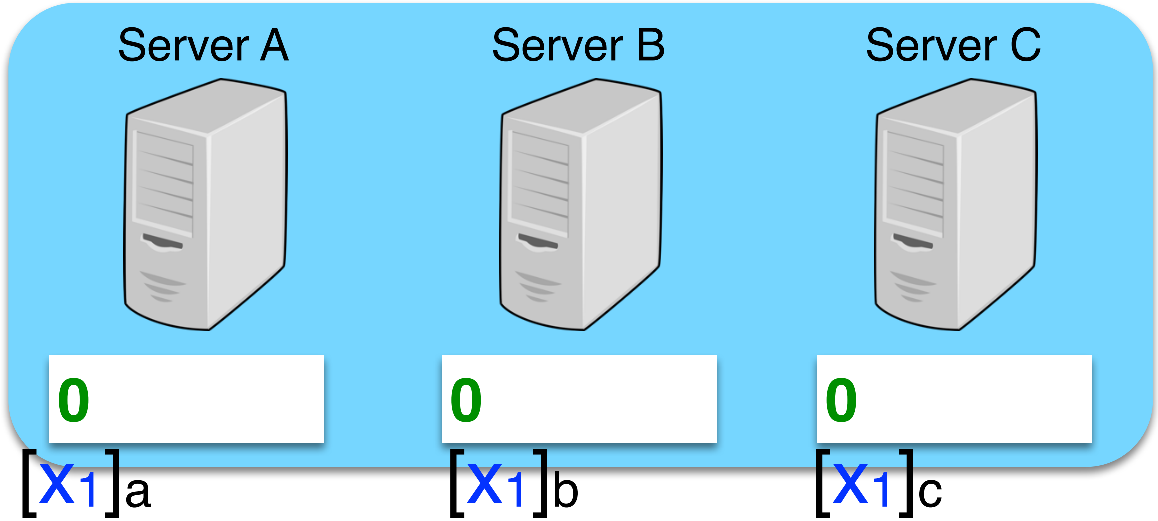
$x_1 \in \mathbb{F}$

Split into shares s.t.

$x_1 = [x_1]_a + [x_1]_b + [x_1]_c \quad \in \mathbb{F}$

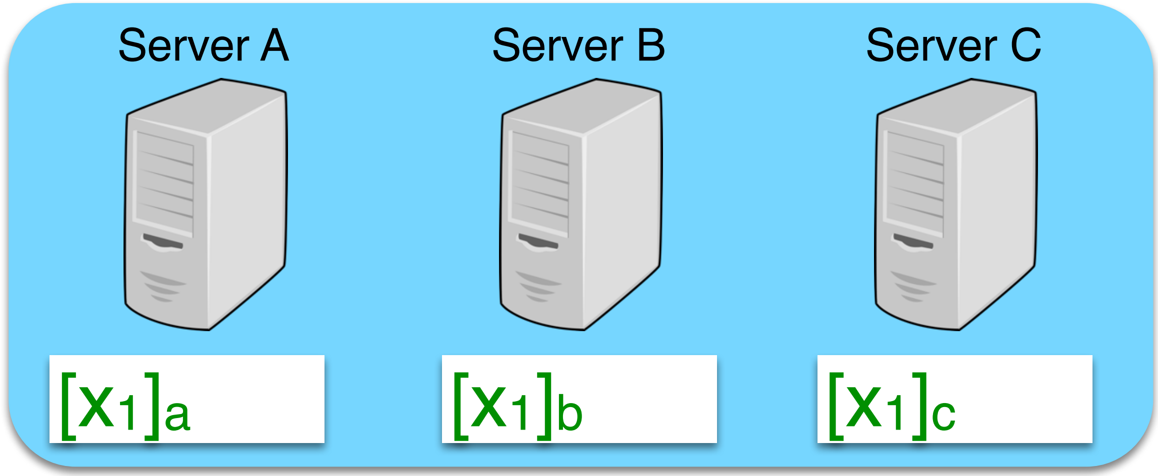
[x] means
“additive share of x”

Private sums: A “straw-man” scheme



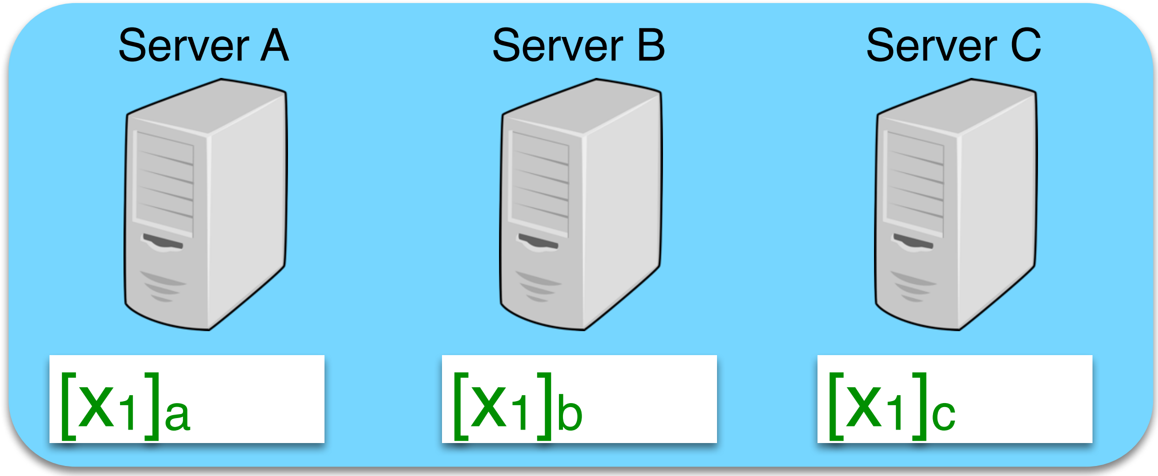
$x_1 \in \mathbb{F}$

Private sums: A "straw-man" scheme



$x_1 \in \mathbb{F}$

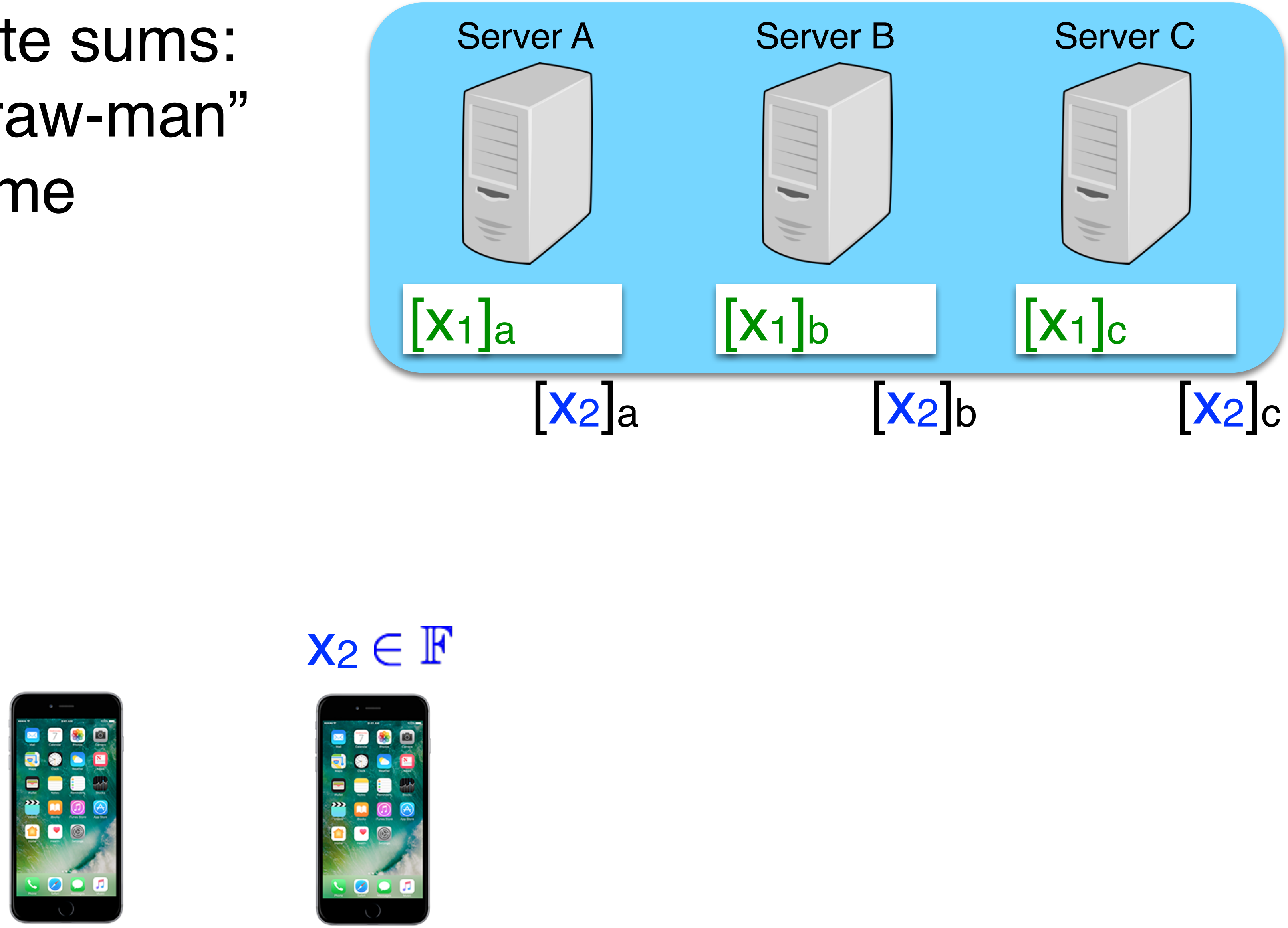
Private sums: A “straw-man” scheme



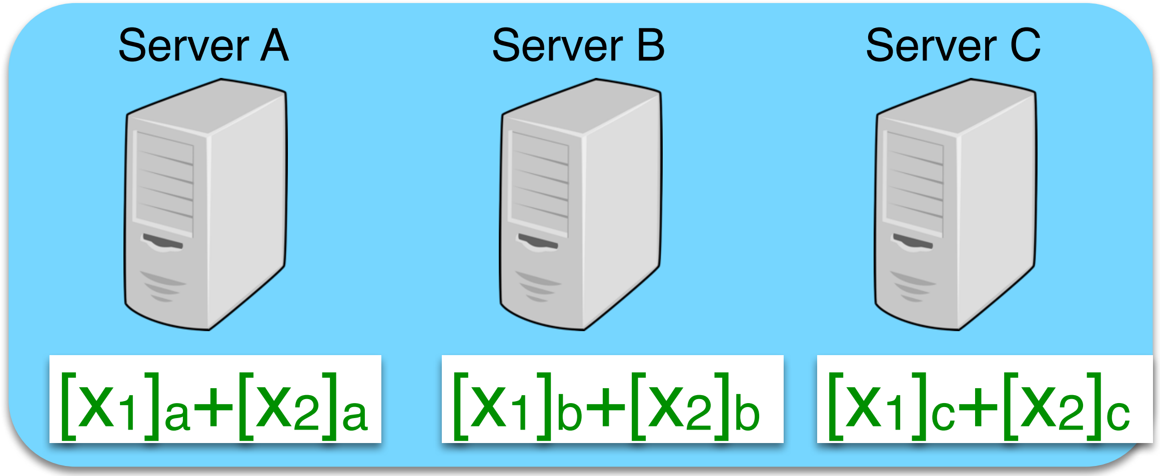
$[x_2]_a$ $[x_2]_b$ $[x_2]_c$

$x_2 \in \mathbb{F}$

Private sums: A "straw-man" scheme

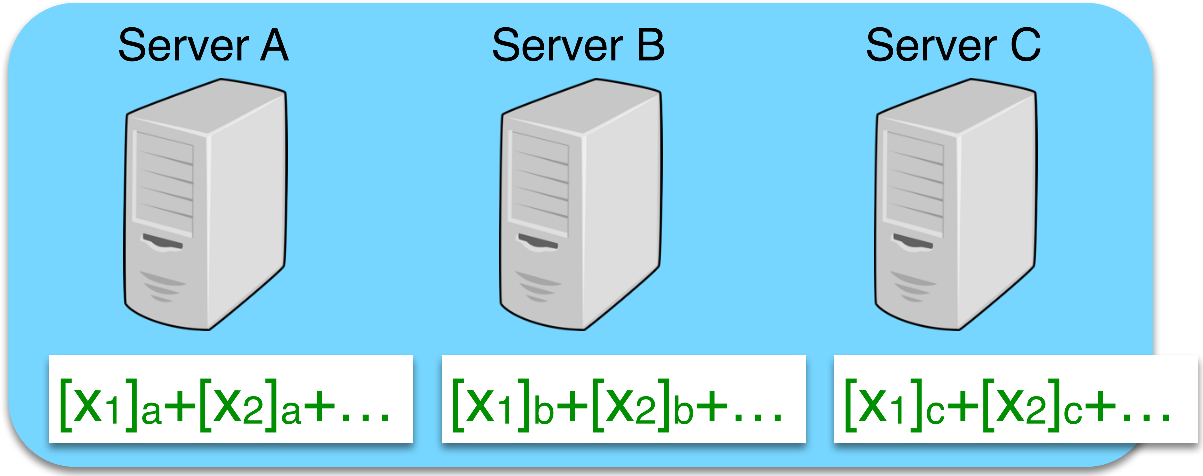


Private sums: A “straw-man” scheme

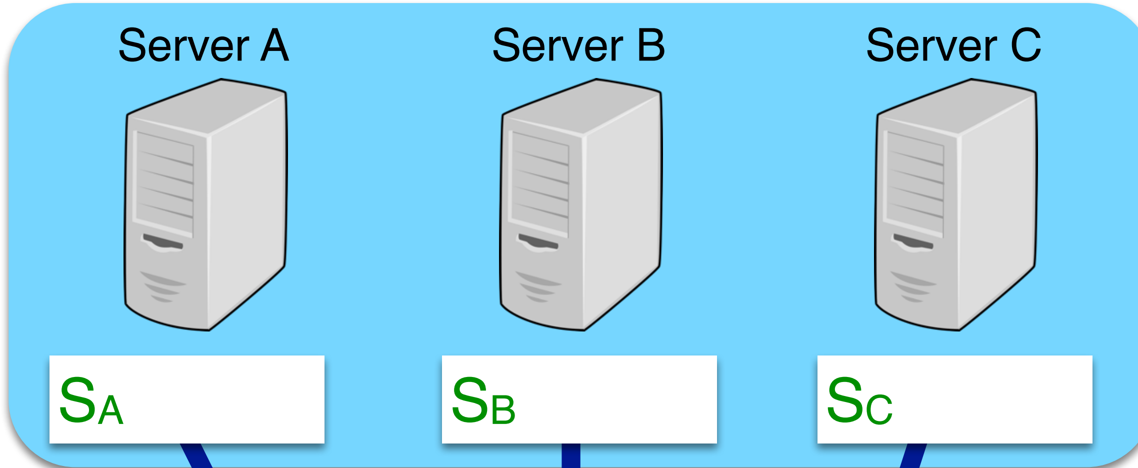


$x_2 \in \mathbb{F}$

Private sums: A “straw-man” scheme



Private sums: A “straw-man” scheme



$$S_A + S_B + S_C = [x_1]_a + [x_1]_b + [x_1]_c + \ldots \in \mathbb{F}$$
$$= x_1 + x_2 + \ldots + x_N$$

Learn that three phones are on the Bridge—but not which three

Strawman computing private sums

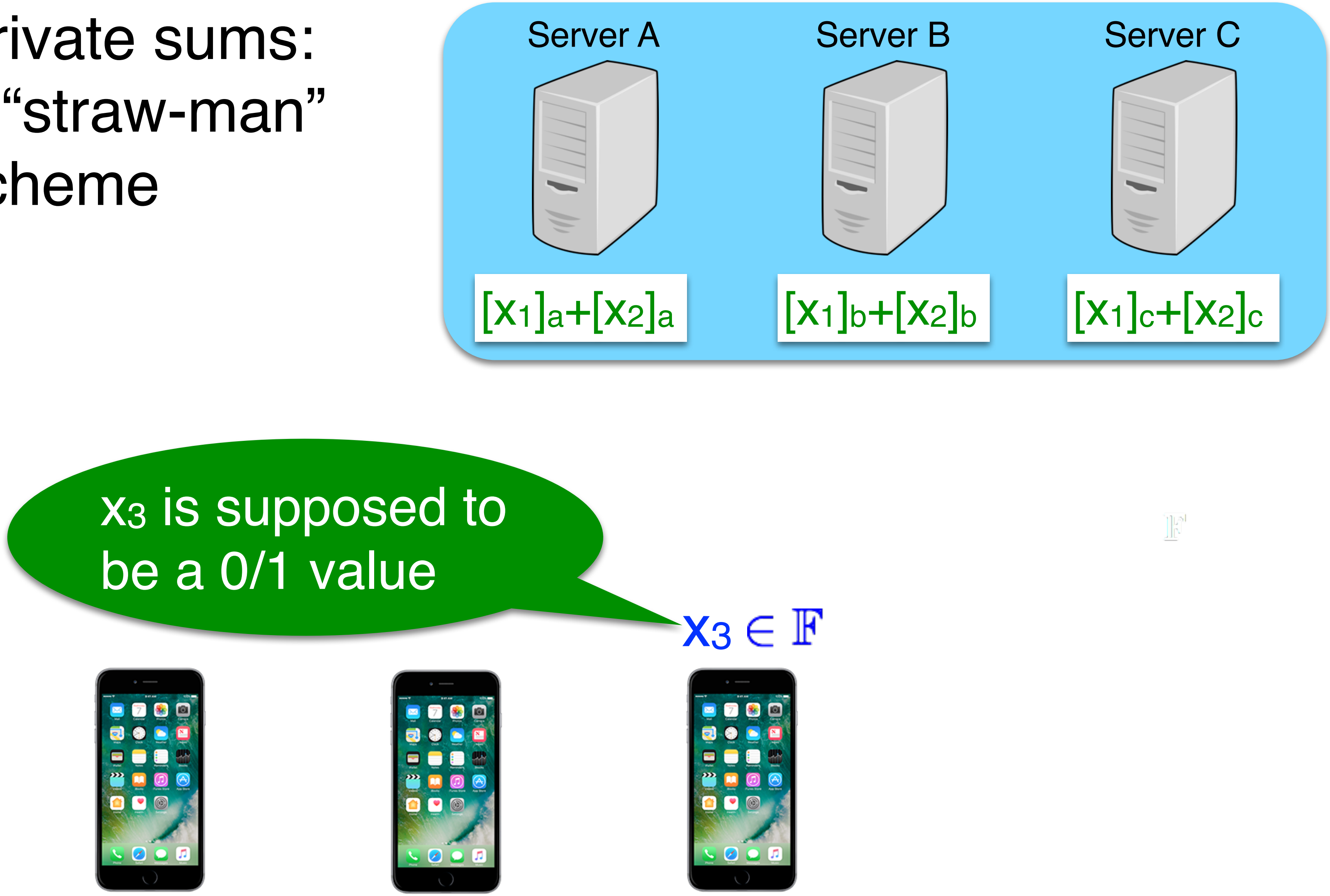
Correctness: if everyone follows the protocol, servers compute the sum of all x_is.

Privacy: any proper subset of the servers can simulate everything given
(a) the public parameters, and
(b) the sum of the x_is.

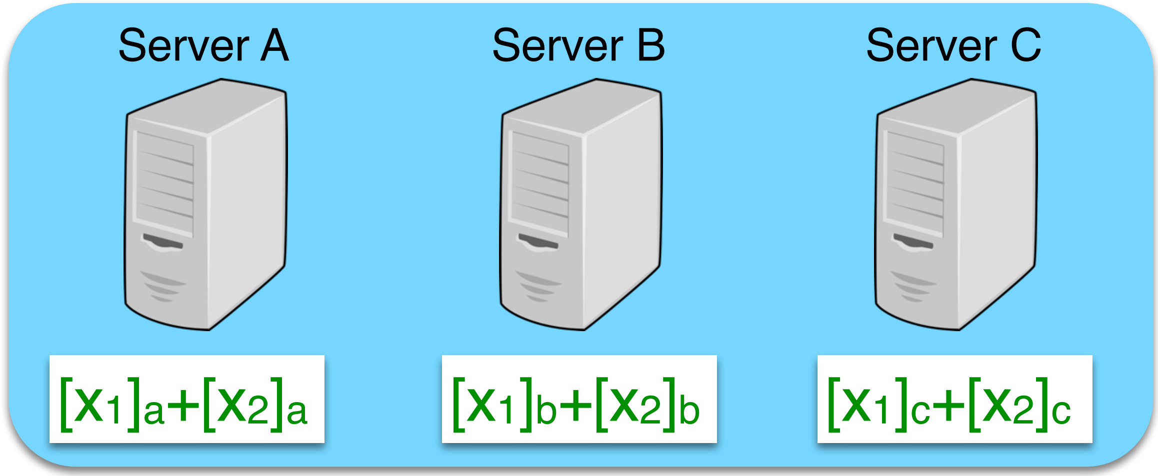
Scalability: by inspection.

Robustness: **???**

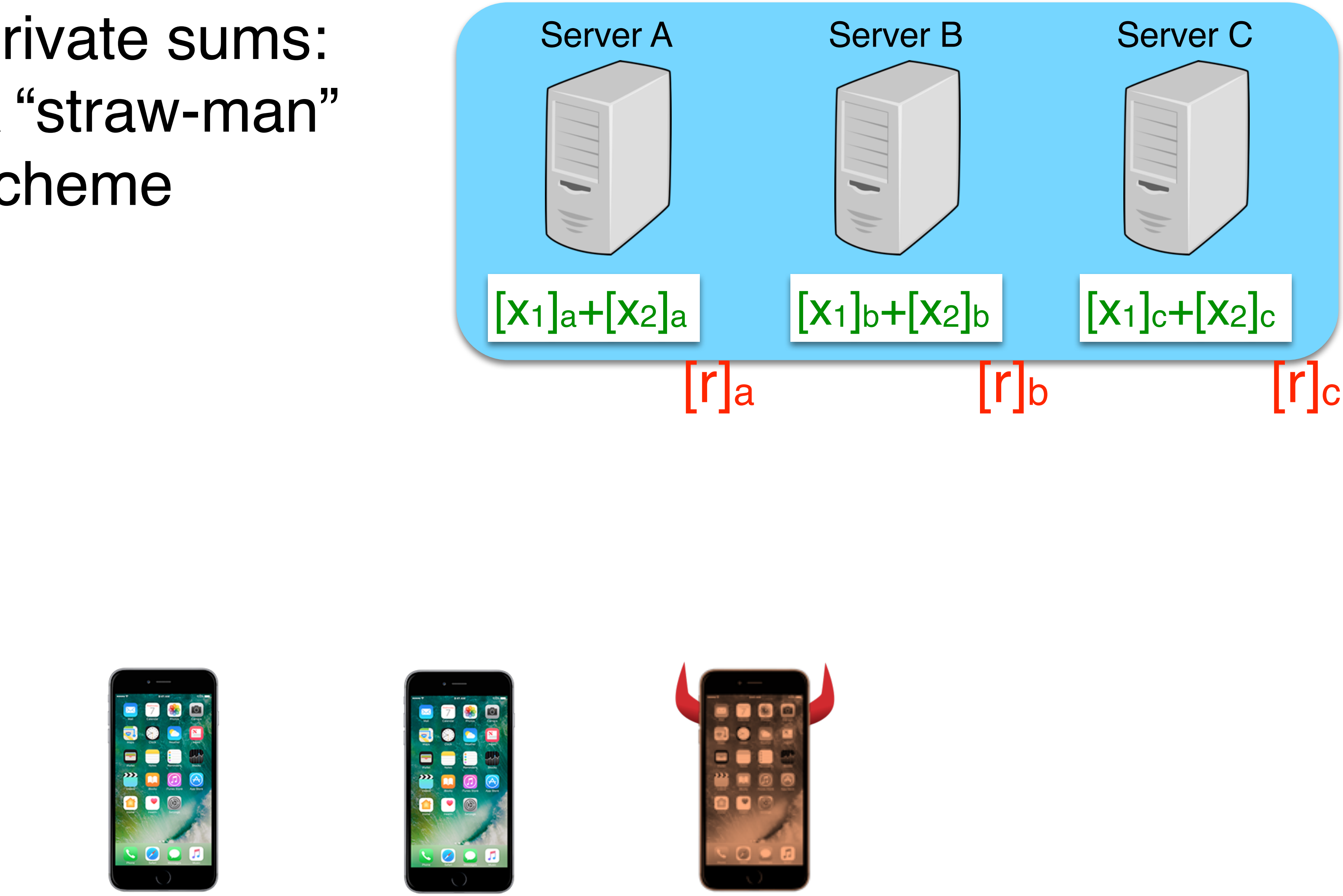
Private sums:
A “straw-man” scheme



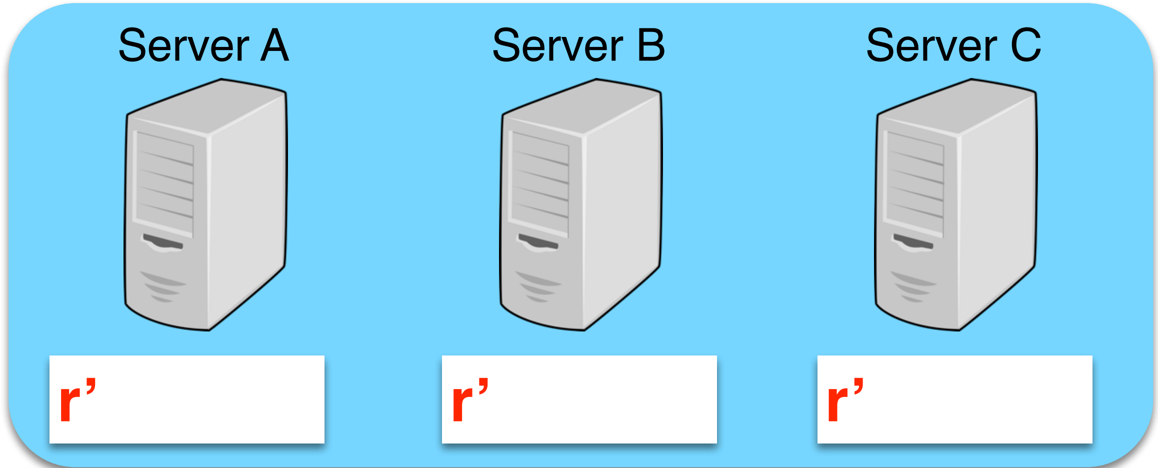
Private sums: A "straw-man" scheme



Private sums: A “straw-man” scheme



Private sums: A "straw-man" scheme

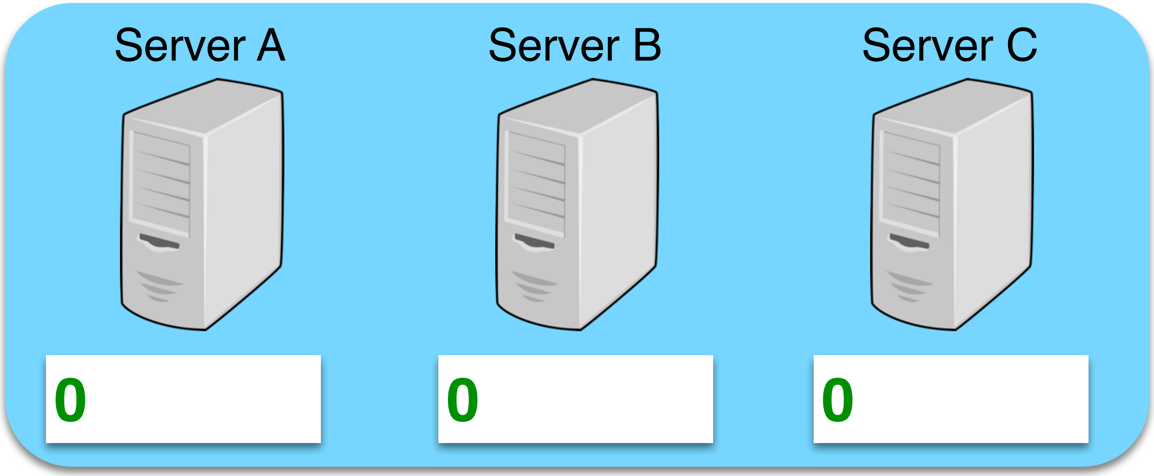


A single bad client **can undetectably corrupt** the sum

Users have **incentives** to cheat

Typical defenses (NIZKs) are costly

Solution:
SNIP Proofs

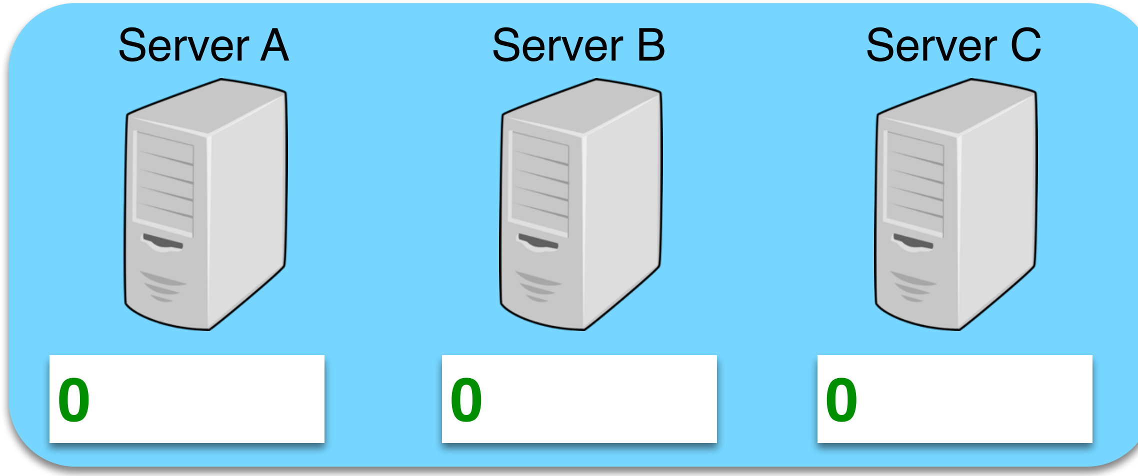


$[x]_a$ $[x]_b$ $[x]_c$

$x \in \mathbb{F}$

x is supposed to be a 0/1 value

Solution:
SNIP Proofs



$[\mathbf{x}]_a$ $\quad$ $[\mathbf{x}]_b$ $\quad$ $[\mathbf{x}]_c$

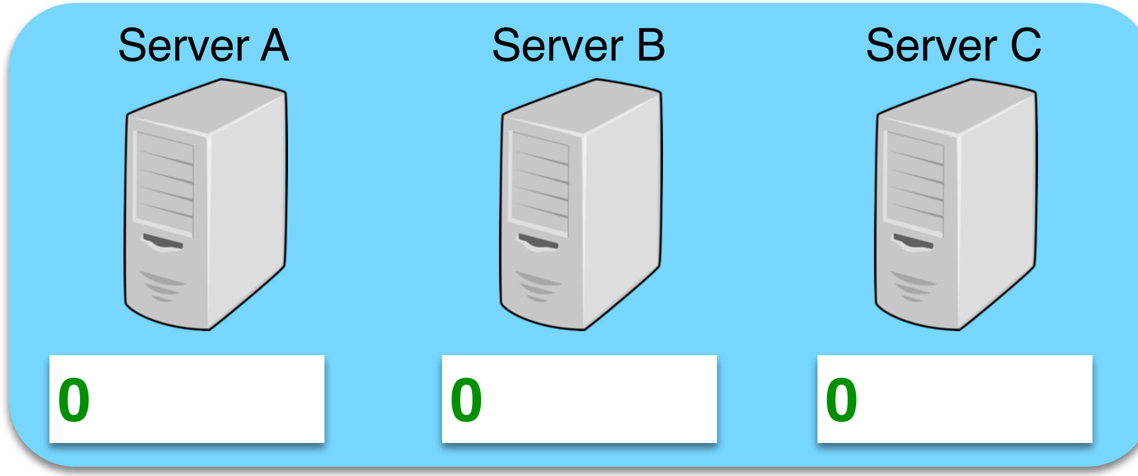
Without learning **x**,
the servers want to ensure that:

$[x]_a + [x]_b + [x]_c \in \{0,1\}$

Remember: these are big integers mod *p*

$x \in \mathbb{F}$

Solution:
SNIP Proofs



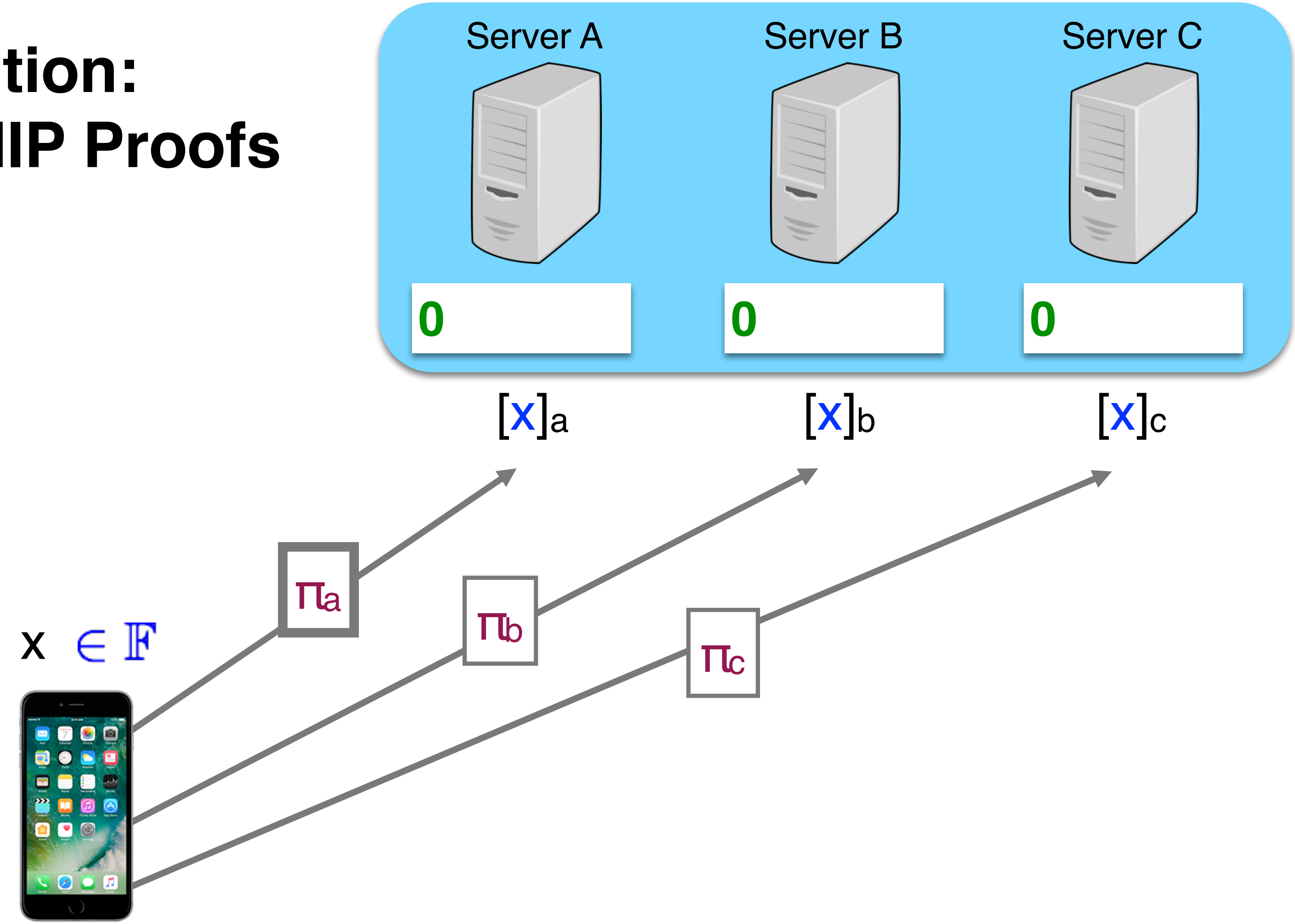
$[x]_a$ $[x]_b$ $[x]_c$

$x \in \mathbb{F}$

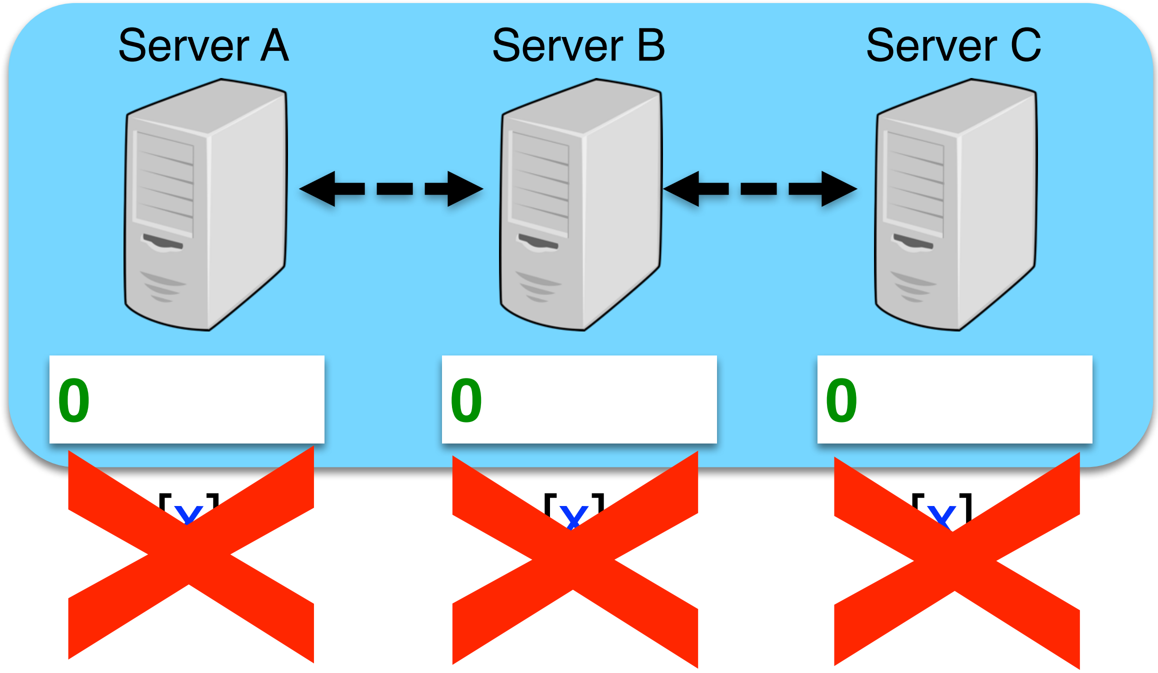
- Servers hold shares of x and a public predicate Valid(·)
- Servers want to test if "Valid(x) = 0" without leaking anything else about x
- The Valid predicate can be an arbitrary circuit:

$$\text{Valid}(x_1,x_2) = \text{“}3 < x_1 < 19 \text{ and } x_2 \in \{0,1,2\}\text{”}$$

Solution: SNIP Proofs



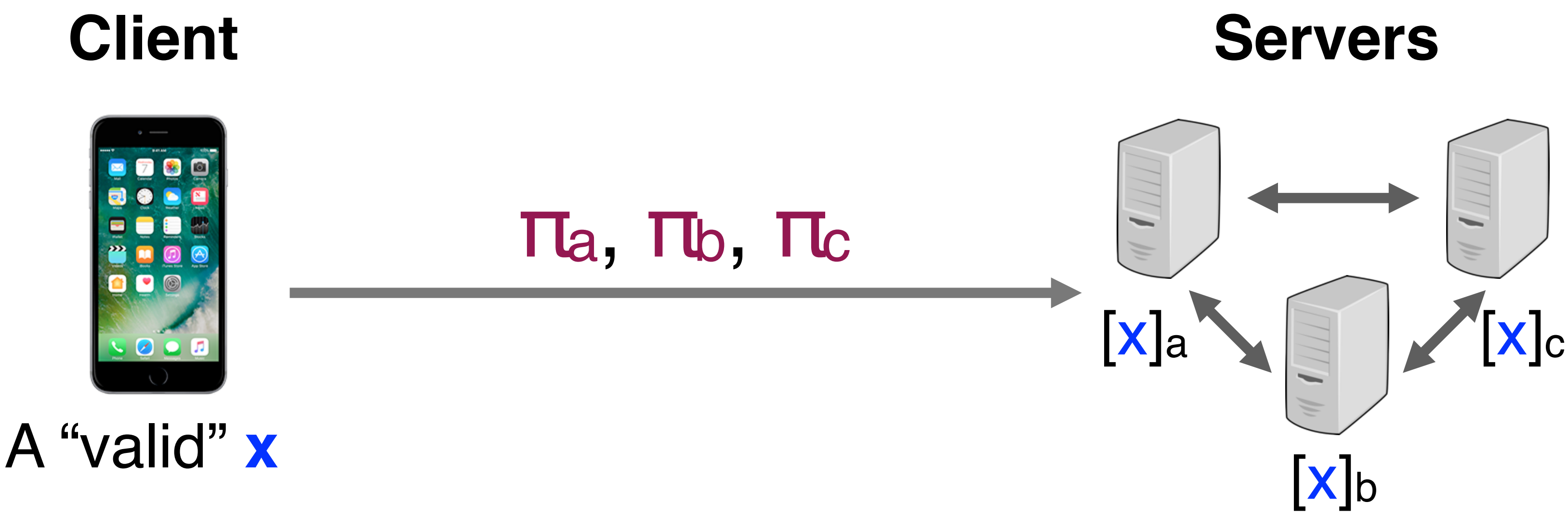
Solution:
SNIP Proofs



x $\in \mathbb{F}$

Prio servers detect and reject malformed client submissions

$\Rightarrow$ a client can influence aggregates by at most $\pm$ 1

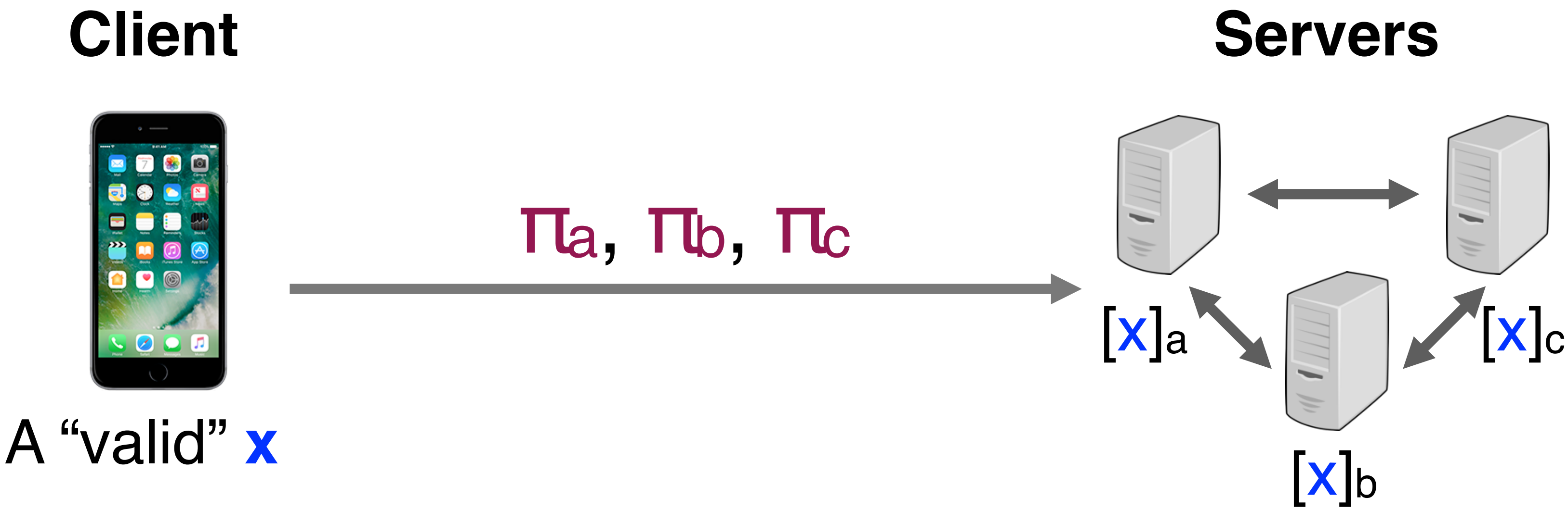


Security goals for SNIPs

Completeness: Honest client convinces honest servers

Soundness: Dishonest client almost never convinces honest servers

Zero-knowledge: Any proper subset of malicious servers learns nothing about x, except that x is valid



Existing techniques	Limitations
Full blown MPC	Heavy setup and comm.
Commitments + NIZKs	High server work
Commitments + SNARKs	
Func. secret sharing [BGI’1	
SNIP	

Info. theoretic techniques
$\Rightarrow$ little comp. overhead
O(1) server-to-server comm.
$|\pi_a|$ is linear in circuit size

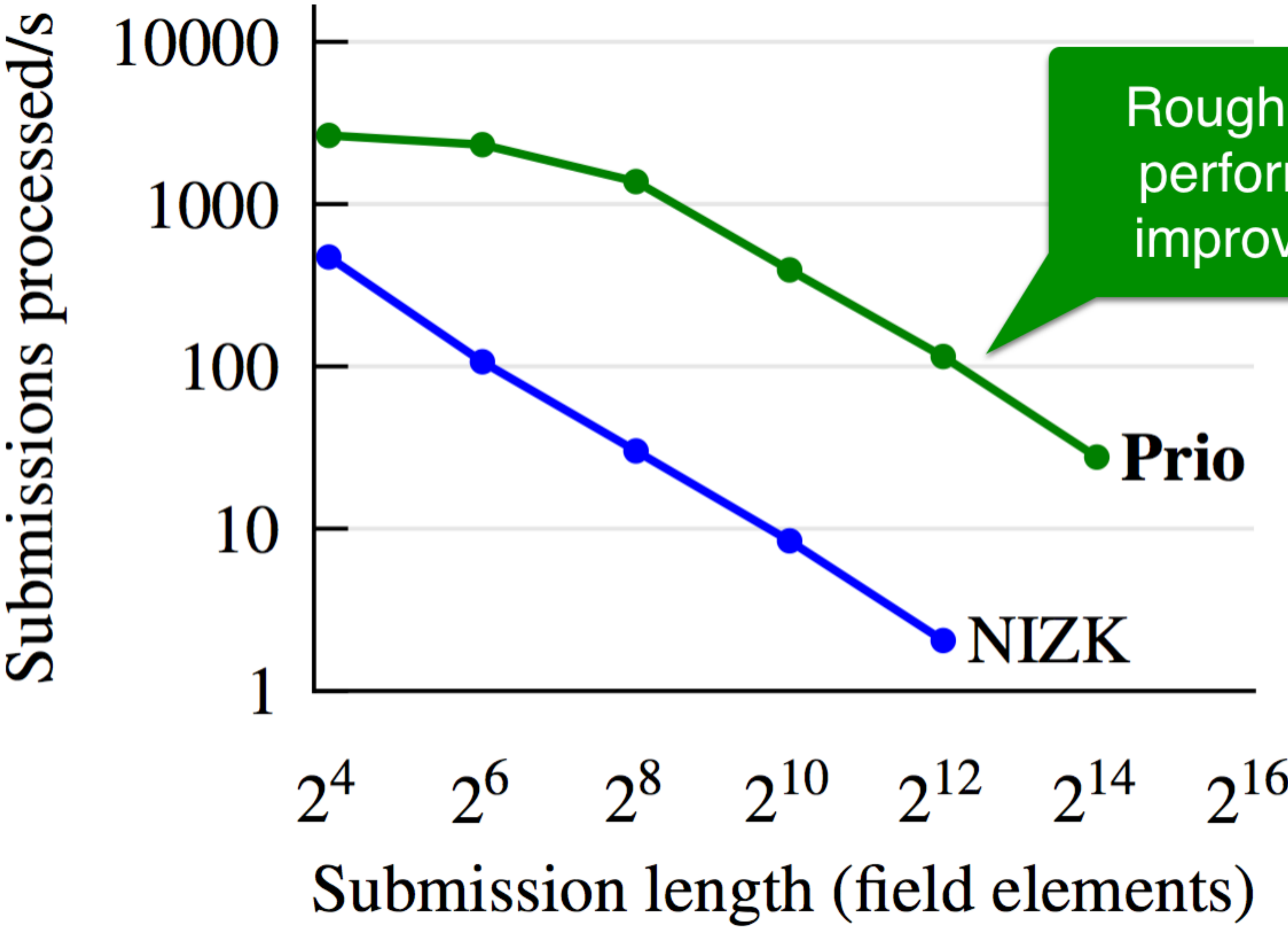
SNIPs: How?

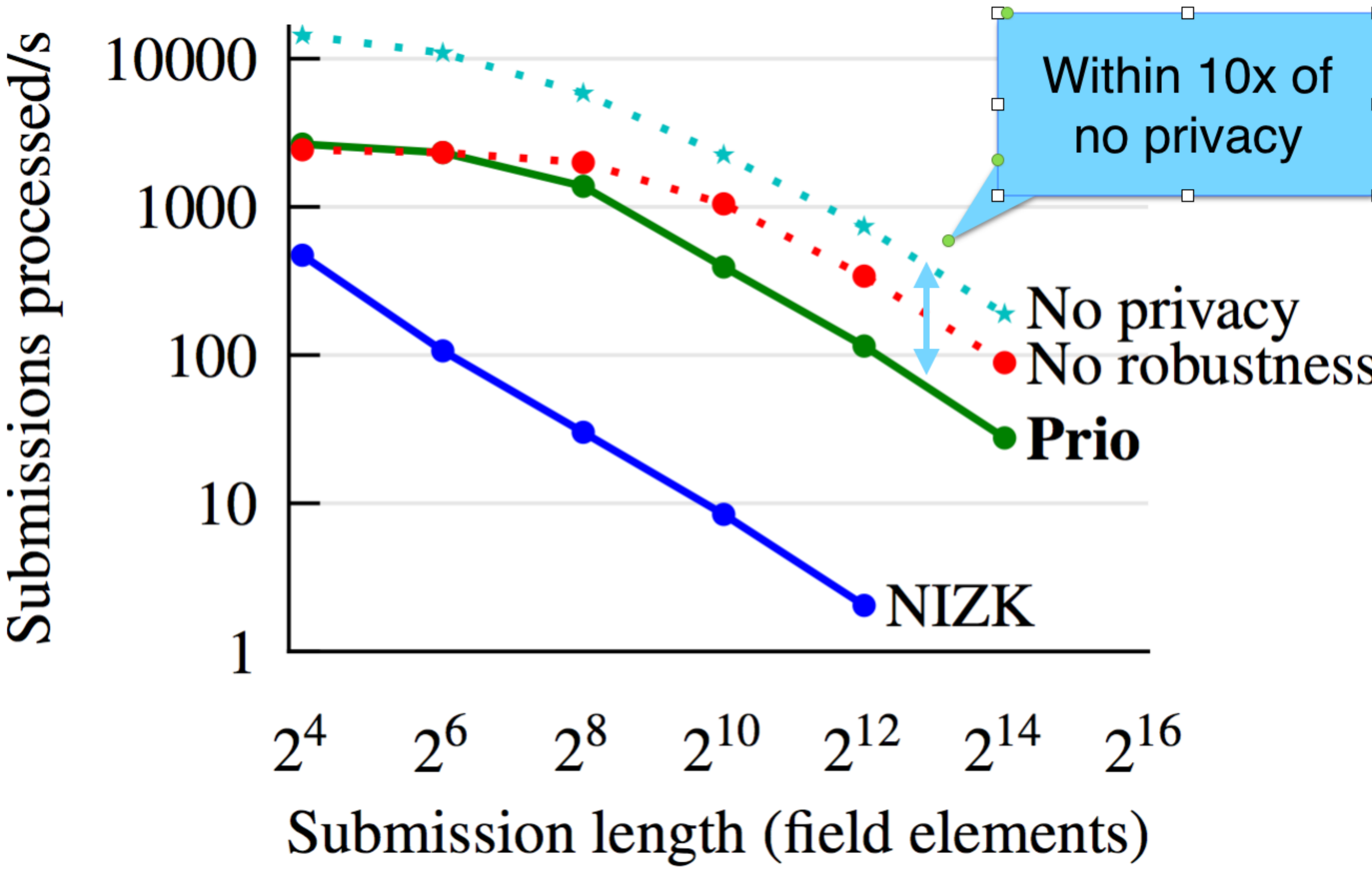
Step 1: reduce verifying circuit to verifying a <u>single</u> multiplication

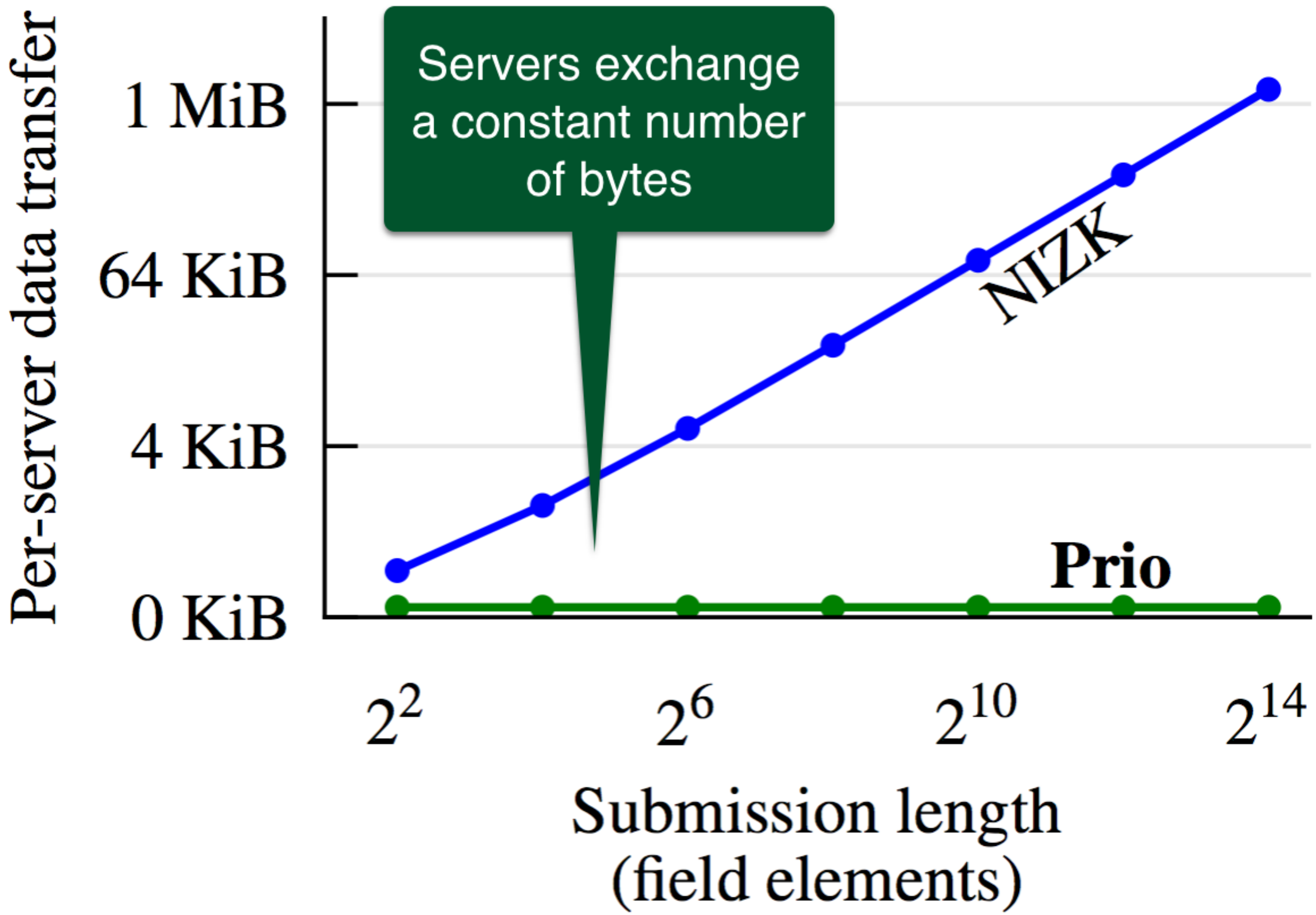
Step 2: Use "Beaver triple" <u>supplied by client</u> to verify the multiplication

Step 3: Inject additional entropy to defend against malicious servers (similar to AMD codes)

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 (field elements)


Servers exchange a constant number of bytes
Per-server data transfer
1 MiB
64 KiB
4 KiB
0 KiB
NIZK
Prio
2^2
2^6
2^{10}
2^{14}
Submission length
(field elements)

Complex statistics

Computing private sums $\Rightarrow$

can compute many other interesting aggregates

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

- Average
- Variance
- Standard deviation
- Most popular value (approx) – small universe
- “Heavy hitters” (approx)

… and even more statistics

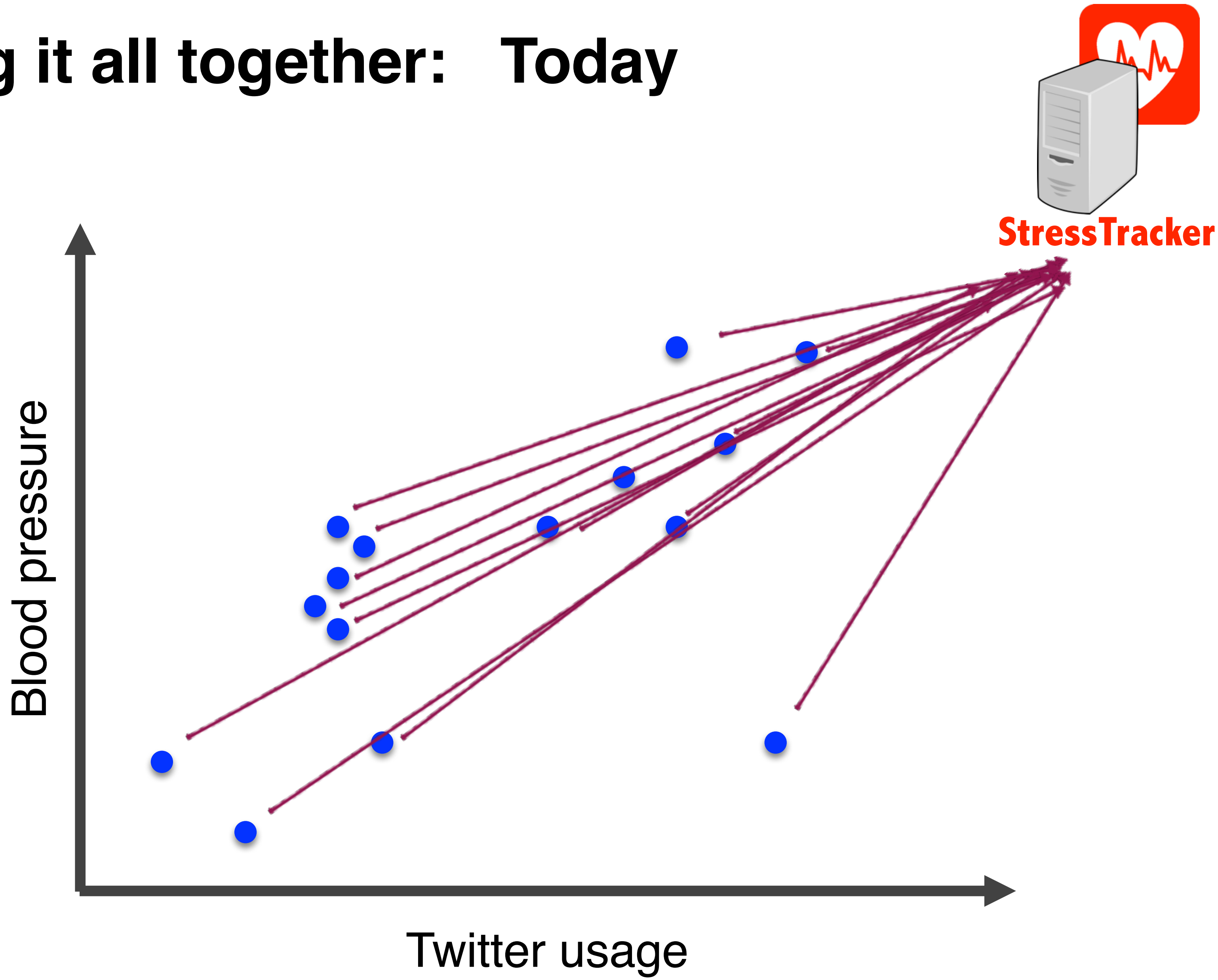
Prio can aggregate a richer class of statistics:

- Approximate **min** and **max**
- Most popular value in a large universe
- Quality of arbitrary machine learning model (R^2)
- **Least-squares regression**

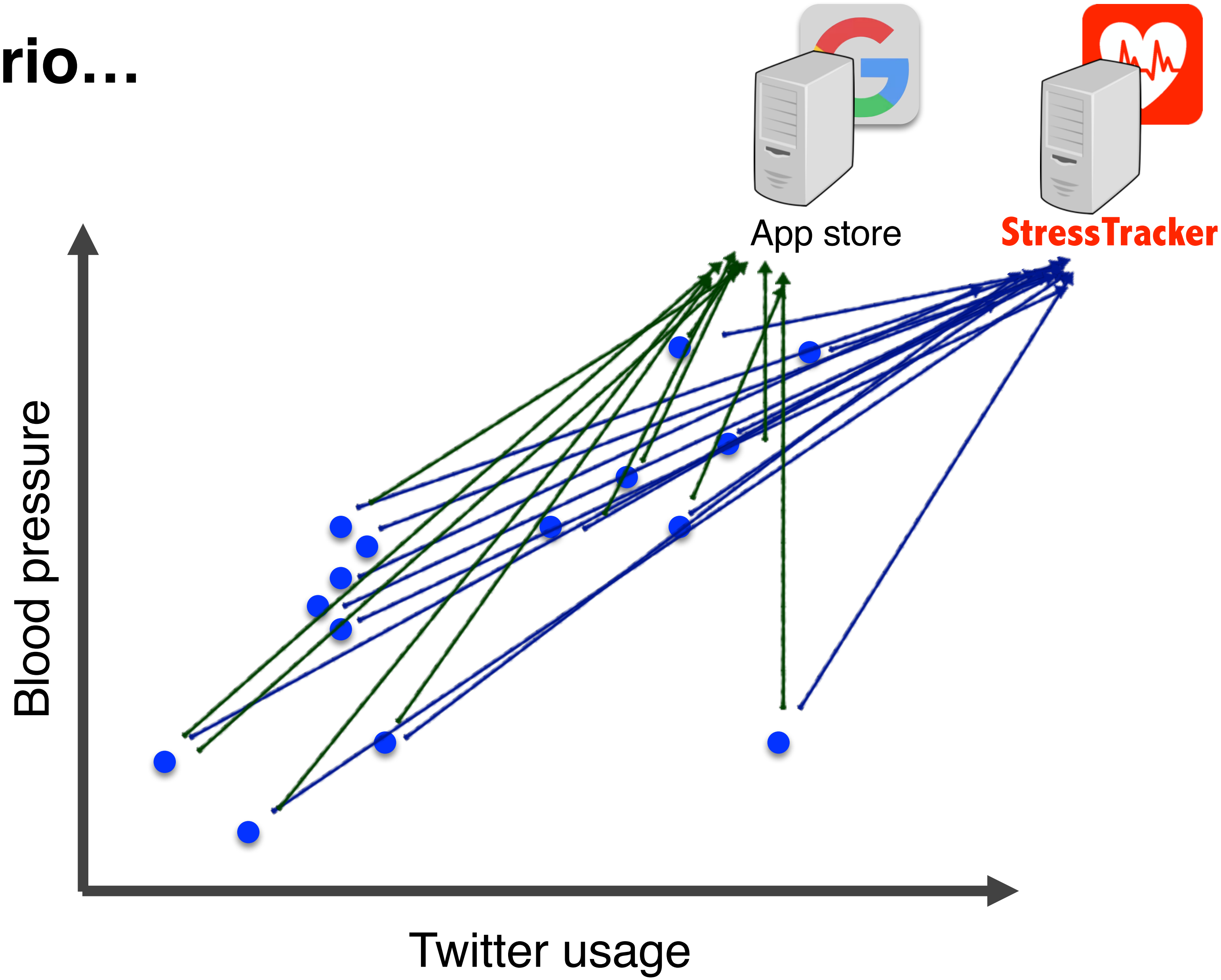
Prio supports a rich set of aggregation functions

Some limitations: cannot compute exact max

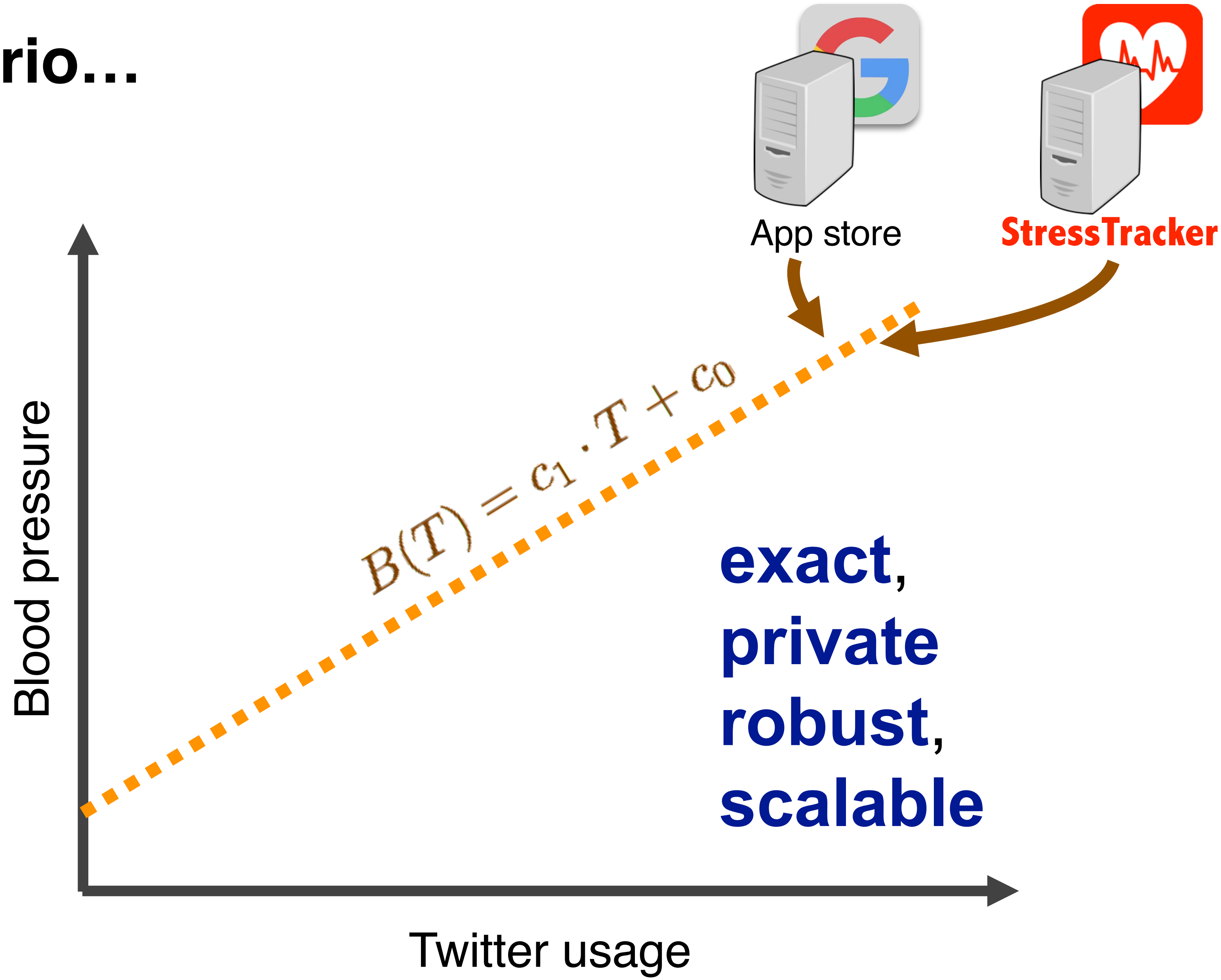
Putting it all together: Today



With Prio…



With Prio…



exact,
private
robust,
scalable

THE END