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

Blood pressure

Twitter usage
Today: Non-private aggregation

Each user has a *private* data point.
Today: Non-private aggregation
Today: Non-private aggregation
Today: Non-private aggregation

\[
B(T) = c_1 \cdot T + c_0
\]
Today: Non-private aggregation

The app provider learned more than it needed

Blood pressure

Twitter usage

$B(T) = c_1 \cdot T + c_0$
Today: Non-private aggregation

Blood pressure vs. Twitter usage graph
This work:
Private aggregation

Blood pressure vs. Twitter usage chart

App store
StressTracker
This work: Private aggregation

Clients send an encrypted share of their data to each aggregator

Blood pressure

Twitter usage

App store

StressTracker
This work:
Private aggregation

Clients send an encrypted share of their data to each aggregator

Blood pressure

Twitter usage
This work:
Private aggregation

Clients send an encrypted share of their data to each aggregator
This work: Private aggregation

Blood pressure

Twitter usage

\[ B(T) = c_1 \cdot T + c_0 \]

The aggregators learn no private client data
Private aggregation

\[ f(x_1, \ldots, x_N) \]

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
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.
Prio is the first system to achieve all four.

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

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

\[ f(x_1, \ldots, x_N) \]

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 \(\Rightarrow\) 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 $\Rightarrow$ Can compute $f(\cdot)$ privately
   …for many $f$’s of interest

See the paper
Related systems

- **Additively homomorphic encryption**
  Succinct sketches (2016), …

- **Multi-party computation** [GMW87], [BGW88]
  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.

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

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

Assume that the servers are non-colluding.
Equivalently: that at least one server is honest.
Private sums: A “straw-man” scheme

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] …
Private sums:
A “straw-man” scheme

Secret sharing
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

Secret sharing
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

Secret sharing
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
Private sums:
A “straw-man” scheme

<table>
<thead>
<tr>
<th>Server A</th>
<th>Server B</th>
<th>Server C</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>-12</td>
<td>-2</td>
</tr>
</tbody>
</table>

0
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

\[
\begin{align*}
\text{Server A} & : 15 \\
\text{Server B} & : -12 \\
\text{Server C} & : -2 \\
\end{align*}
\]
Private sums: A “straw-man” scheme

Server A: 15
Server B: -12
Server C: -2

-10 7 3
Private sums: A “straw-man” scheme

Server A
15 - 10

Server B
-12 + 7

Server C
-2 + 3

0
Private sums: A “straw-man” scheme

Server A
15-10+…

Server B
-12+7+…

Server C
-2+3+…
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

\[ S_A + S_B + S_C = 15 + -10 + \ldots \]
\[ = 1 + 0 + \ldots + 1 \]
Private sums: A “straw-man” scheme

\[ S_A + S_B + S_C = 15 + (-10) + \ldots + 1 \]

Servers learn the sum of client values and learn *nothing else.*
Private sums: A “straw-man” scheme

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

\[
S_A + S_B + S_C = 15 + (-10) + \ldots
= 1 + 0 + \ldots + 1
\]
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 $x_i$s.

Privacy: Any proper subset of the servers learns nothing but the sum of the $x_i$s.

Efficiency: Follows by inspection.
Computing private sums

**Exact correctness:** If everyone follows the protocol, servers compute the sum of all $x_i$s.

**Privacy:** Any proper subset of the servers learns nothing but the sum of the $x_i$s.

**Efficiency:** Follows by inspection.

**Robustness:** ???
Private sums: A “straw-man” scheme

Server A: 15-10
Server B: -12+7
Server C: -2+3
Private sums: A “straw-man” scheme

Server A

15-10

Server B

-12+7

Server C

-2+3

x is supposed to be a 0/1 value
Private sums: A “straw-man” scheme

Server A: 15-10
Server B: -12+7
Server C: -2+3
Private sums:
A “straw-man” scheme

15-10
-12+7
-2+3
Private sums:
A “straw-man” scheme

An evil client needn’t follow the rules!
Private sums: A “straw-man” scheme

An evil client needn’t follow the rules!

10 + 4 + 7 = 21
Private sums: A “straw-man” scheme

Server A: 15 - 10 = 10
Server B: -12 + 7 = -4
Server C: -2 + 3 = 1

Total: 10, 4, 7
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)

$x = 1$
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

\[ x = 1 \]
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)

\[ x = 1 \]
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

In this example, the servers want to ensure that their shares sum to 0 or 1 …without learning $x$. 

$x = 1$
**Contribution 1**

Secret-shared non-interactive proofs (SNIPs)

More generally, servers

- hold shares of the client’s private value \( x \)
- hold an arbitrary public predicate \( \text{Valid}(\cdot) \) – expressed as an arithmetic circuit
- want to test if “\( \text{Valid}(x) \)” holds, without leaking \( x \)
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

More generally, servers
- hold shares of the client’s private value $x$
- hold an arbitrary public predicate $\text{Valid}(\cdot)$ – expressed as an arithmetic circuit
- want to test if “$\text{Valid}(x)$ holds, without leaking $x$ For our running example:
  $\text{Valid}(x) = \text{“} x \in \{0,1\} \text{“}$
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

More generally, servers
• hold shares of the client’s private value $x$
• hold an arbitrary public predicate $\text{Valid}(\cdot)$ – expressed as an arithmetic circuit
• want to test if “$\text{Valid}(x)$” holds, without leaking $x$
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

\[ x = 1 \]
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

\[ x = 1 \]

\[ \prod_a \]

\[ x_a \quad X_b \quad X_c \]
Contribution 1
Secret-shared non-interactive proofs (SNIPs)
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

\[ x = 1 \]

\[ \Pi_a \quad \Pi_b \quad \Pi_c \]

\[ X_a \quad X_b \quad X_c \]
**Contribution 1**
Secret-shared non-interactive proofs (SNIPs)

\[
\pi_a, \; x_a \quad \pi_b, \; x_b \quad \pi_c, \; x_c
\]

\[x = 1\]
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

\[ x = 1 \]
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

Servers gossip

$x = 1$
**Contribution 1**

Secret-shared non-interactive proofs (SNIPs)

\[ x = 1 \]
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

Server A: $\Pi_a, X_a$
Server B: $\Pi_b, X_b$
Server C: $\Pi_c, X_c$

Ok.

$x = 1$
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

\[ \pi_a, \ x_a \quad \pi_b, \ x_b \quad \pi_c, \ x_c \]

\[ x = 1 \]
Contribution 1
Secret-shared non-interactive proofs (SNIPs)

\[
x = 1
\]
**Contribution 1**
Secret-shared non-interactive proofs (SNIPs)

\[ \pi_a, x_a \]
\[ \pi_b, x_b \]
\[ \pi_c, x_c \]

\[ x = 1 \]
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)

Fail

\[ x = 1 \]
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)

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

\[ x = 1 \]
We need a proof system

Prover

A “valid” $x$

Verifiers

$\Pi_a, \Pi_b, \Pi_c$

$X_a, X_b, X_c$
We need a proof system

A "valid" $x$

Prover

$\pi_a, \pi_b, \pi_c$

Verifiers

$X_a, X_b, X_c$

Valid($x$) holds?
We need a proof system

Prover

A “valid” $x$

Verifiers

$\pi_a, \pi_b, \pi_c$

$X_a, X_b, X_c$
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

Could run secure multiparty computation to check that $\text{Valid}(x) = 1$. [GMW87], [BGW88]
How SNIPs work

Could run secure multiparty computation to check that $\text{Valid}(x) = 1$.

[GMW87], [BGW88]
How SNIPs work
How SNIPs work

Server A

Server B

Server C

\[ X_a \quad X_b \quad X_c \]
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

Server A

Server B

Server C

\( X_a \)

\( X_b \)

\( X_c \)
How SNIPs work

Server A

Server B

Server C

$X_a$

$X_b$

$X_c$

Servers check that the transcripts are valid and consistent.
How SNIPs work

Servers check that the transcripts are valid and consistent.
How SNIPs work

- Server A
- Server B
- Server C

π<sub>a</sub> x<sub>a</sub>

π<sub>b</sub> x<sub>b</sub>

π<sub>c</sub> x<sub>c</sub>

Servers check that the transcripts are valid and consistent.

Checking a transcript is much easier than generating it!
How SNIPs work
How SNIPs work
How SNIPs work

“Randomized digest” of the transcript
How SNIPs work
How SNIPs work

Server A

Server B

Server C

$D_a$

$D_b$

$D_c$
How SNIPs work

Server A

Server B

Server C

Dₐ

D₏

Dₐ
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.
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.
<table>
<thead>
<tr>
<th>M = # of multiplication gates in \text{Valid}(\cdot) circuit</th>
<th>Public-key ops.</th>
<th>Communication</th>
<th>Slow-down</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client</td>
<td>Server</td>
<td>C-to-S</td>
</tr>
</tbody>
</table>

- \text{Valid}(\cdot) circuit
<table>
<thead>
<tr>
<th>M = # of multiplication gates in \text{Valid}(\cdot) circuit</th>
<th><strong>Public-key ops.</strong></th>
<th><strong>Communication</strong></th>
<th><strong>Slow-down</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dishonest-maj. MPC [CLOS02], [DPSZ12], ...</td>
<td>0</td>
<td>$\Theta(M)$</td>
<td>0</td>
</tr>
</tbody>
</table>

$5,000\times$ at server
<table>
<thead>
<tr>
<th>M = # of multiplication gates in $\text{Valid}(\cdot)$ circuit</th>
<th>Public-key ops.</th>
<th>Communication</th>
<th>Slow-down</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client</td>
<td>Server</td>
<td>C-to-S</td>
</tr>
<tr>
<td>Dishonest-maj. MPC</td>
<td>0</td>
<td>$\Theta(M)$</td>
<td>0</td>
</tr>
<tr>
<td>[CLOS02], [DPSZ12], ...</td>
<td>$\Theta(M)$</td>
<td>$\Theta(M)$</td>
<td>$\Theta(M)$</td>
</tr>
<tr>
<td>Commits + NIZKs</td>
<td>$\Theta(M)$</td>
<td>$\Theta(M)$</td>
<td>$\Theta(M)$</td>
</tr>
<tr>
<td>M = # of multiplication gates in $\text{Valid}(\cdot)$ circuit</td>
<td>Public-key ops.</td>
<td>Communication</td>
<td>Slow-down</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>----------------</td>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>Client</td>
<td>Server</td>
<td>C-to-S</td>
</tr>
<tr>
<td>Dishonest-maj. MPC [CLOS02], [DPSZ12], ...</td>
<td>0</td>
<td>$\Theta(M)$</td>
<td>0</td>
</tr>
<tr>
<td>Commits + NIZKs [FS86], [CP92], [CS97], ...</td>
<td>$\Theta(M)$</td>
<td>$\Theta(M)$</td>
<td>$\Theta(M)$</td>
</tr>
<tr>
<td>Commits + SNARKs [GGPR13], [BCGTV13], ...</td>
<td>$\Theta(M)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>M = # of multiplication gates in Valid(·) circuit</td>
<td>Public-key ops.</td>
<td>Communication</td>
<td>Slow-down</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>Client</td>
<td>Server</td>
<td>C-to-S</td>
</tr>
<tr>
<td>Dishonest-maj. MPC [CLOS02], [DPSZ12], ...</td>
<td>0</td>
<td>$\Theta(M)$</td>
<td>0</td>
</tr>
<tr>
<td>Commits + NIZKs [FS86], [CP92], [CS97], ...</td>
<td>$\Theta(M)$</td>
<td>$\Theta(M)$</td>
<td>$\Theta(M)$</td>
</tr>
<tr>
<td>Commits + SNARKs [GGPR13], [BCGTV13], ...</td>
<td>$\Theta(M)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>This work: SNIPs</td>
<td>0</td>
<td>0</td>
<td>$\Theta(M)$</td>
</tr>
</tbody>
</table>

Notes:
- Commits + NIZKs and Commits + SNARKs use SNARKs [GGPR13], [BCGTV13], ...
- This work: SNIPs uses SNARKs [GGPR13], [BCGTV13], ...

Slow-down:
- 5,000x at server
- 50x at server
- 500x at client
- 1x

Additional notes:
- $\Theta(M)$ represents a term proportional to $M$
- $O(1)$ represents a constant time complexity
<table>
<thead>
<tr>
<th>M = # of multiplication gates in \text{Valid}(\cdot) circuit</th>
<th>Public-key ops.</th>
<th>Communication</th>
<th>Slow-down</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client</td>
<td>Server</td>
<td>C-to-S</td>
</tr>
<tr>
<td>Dishonest-maj. MPC</td>
<td>0</td>
<td>(\Theta(M))</td>
<td>0</td>
</tr>
<tr>
<td>[CLOS02], [DPSZ12], ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commits + NIZKs</td>
<td>(\Theta(M))</td>
<td>(\Theta(M))</td>
<td>(\Theta(M))</td>
</tr>
<tr>
<td>[FS86], [CP92], [CS97], ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commits + SNARKs</td>
<td>(\Theta(M))</td>
<td>(O(1))</td>
<td>(O(1))</td>
</tr>
<tr>
<td>[GGPR13], [BCGTV13], ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This work: SNIPs</td>
<td>0</td>
<td>0</td>
<td>(\Theta(M))</td>
</tr>
</tbody>
</table>

For specific \text{Valid}() circuits, it is possible to eliminate this cost [BGI16]
<table>
<thead>
<tr>
<th>M = # of multiplication gates in ( \text{Valid}(\cdot) ) circuit</th>
<th>Public-key ops.</th>
<th>Communication</th>
<th>Slow-down</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client</strong></td>
<td><strong>Server</strong></td>
<td><strong>C-to-S</strong></td>
<td><strong>S-to-S</strong></td>
</tr>
<tr>
<td>Dishonest-maj. MPC</td>
<td>( 0 )</td>
<td>( \Theta(M) )</td>
<td>( 0 )</td>
</tr>
<tr>
<td>[CLOS02], [DPSZ12], …</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commits + NIZKs</td>
<td>( \Theta(M) )</td>
<td>( \Theta(M) )</td>
<td>( \Theta(M) )</td>
</tr>
<tr>
<td>[FS86], [CP92], [CS97], …</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commits + SNARKs</td>
<td>( \Theta(M) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>[GGPR13], [BCGTV13], …</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>This work: SNIPs</strong></td>
<td>( 0 )</td>
<td>( 0 )</td>
<td>( \Theta(M) )</td>
</tr>
</tbody>
</table>

This work: SNIPs
From sums to more complex aggregates

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

- 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

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

E.g., for privately measuring telemetry data.
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
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)
Five-server cluster in five Amazon data centers

Submissions processed/s vs. Submission length (0/1 integers)

- Submission length: $2^4, 2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}$
- Submissions processed/s: 10000, 1000, 100, 10, 1

NIZK
Five-server cluster in five Amazon data centers

Submission length (0/1 integers)

Submissions processed/s

Prio

NIZK
Five-server cluster in five Amazon data centers

![Graph showing submissions processed per second versus submission length (0/1 integers). The graph compares Prio and NIZK methods. The x-axis represents the submission length in terms of powers of 2, and the y-axis represents submissions processed per second on a log scale. The graph shows a clear decrease in submissions processed per second as submission length increases for both methods, with NIZK consistently higher than Prio.]
Five-server cluster in five Amazon data centers

Submission length (0/1 integers)

Submissions processed/s

50x performance improvement
Five-server cluster in five Amazon data centers

Submission length (0/1 integers)

Submissions processed/s

- No robustness
- Prio
- NIZK
Five-server cluster in five Amazon data centers

![Diagram](image-url)
Five-server cluster in five Amazon data centers

Within 10x of no privacy

Submission length (0/1 integers)

Submission processed/s

- Prio
- NIZK
- No privacy
- No robustness
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 bandwidth usage 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 *bandwidth usage* 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

Blood pressure vs. Twitter usage

StressTracker
With Prio…

![Blood pressure vs Twitter usage scatter plot](image)

- App store
- StressTracker
With Prio…

Blood pressure vs Twitter usage graph showing a positive correlation.
With Prio...

Blood pressure

Twitter usage

App store

StressTracker
Blood pressure

With Prio…

Twitter usage

$B(T) = c_1 \cdot T + c_0$
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!

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

Henry Corrigan-Gibbs
henrycg@cs.stanford.edu