COM-506

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

Federico Villa, Gabriele Stentella, Octave Charrin

Context

- Many modern devices collect data and send it to cloud services.
- Storing private data, the services create a single point of failure.
- Huge threat for privacy and security.
- The services need aggregate statistics.

Collect data from mobile apps.

Private compute services.

Spread data over multiple countries.

How do we split trust in a way that protects privacy **and** maintains functionality?

Introduction

Idea: the clients send an encrypted share of their data points to each aggregator.

How?

Goals:

1. Servers learn the output of the aggregation function (correctness).
2. But learn nothing more (privacy).
3. The system is robust ⇒ detects incorrect submissions.
4. The protocol is efficient and scalable
   ⇒ no heavy public-key cryptography operations.

Previous approaches

Randomized response

- Clients flip their bits with fixed probability $p < 0.5$
- Every bit leaks information (especially for low p). ⇒ weak privacy
- With p too high the aggregation becomes useless.
- Bounded client contribution.

Encryption

- Stronger privacy guarantees.
- Unbounded client contribution.
- Not scalable.

Prio - overview

- Small number of servers, large number of clients.
- Built using Secret-shared Non-Interactive Proofs (SNIPs) and Affine-aggregatable Encodings (AFEs).

Assumptions on the network

- PKI and basic cryptographic primitives.
- No synchrony.
- Adversary monitors the network and controls the packets.

Prio - simplified

We use $[x]_s$ to denote the *s*th share of *x*:

$$x = \sum_s [x]_s$$

Input: one bit integer x_i

Aggregation: sum $\sum_i x_i$

1. Private value secret-shared between *s* servers. $x_i = [x_i]_1 + \cdots + [x_i]_s \in \mathbb{F}_p$
2. Each server add the share to its internal accumulator.
3. The servers publish the accumulators.
4. The sum of the accumulators is the desired aggregation.

- Privacy from secret sharing.
- **No robustness.**
- Only sum.

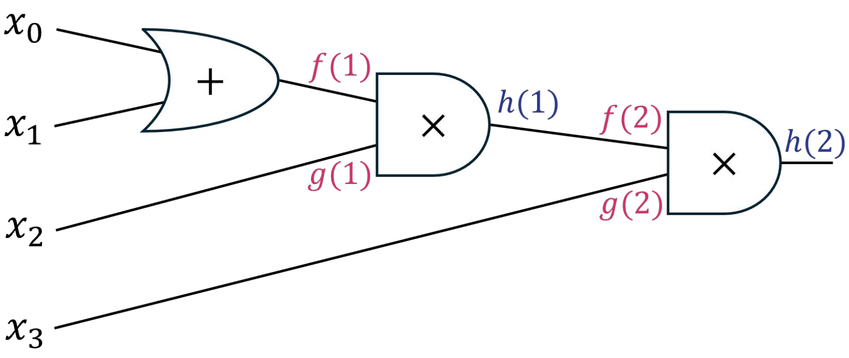
SNIPs: Secret-shared Non-Interactive Proofs

- Linear additive secret-sharing over field $\mathbb{F}$
- Validation predicate Valid ⇒ encoded in an arithmetic circuit

SNIP protocol

1. Client evaluates the circuit.
2. Servers check consistency.
3. Polynomial validation ⇒ polynomial identity test.
   Multiplication of shares.
4. Final computation and verification.

1. Client evaluates the circuit



- Three randomized polynomials f, g, h.
- M multiplication gates.
- Left input u_t
- Right input v_t
- $h(t) = f(t) \cdot g(t) = u_t \cdot v_t \quad \forall t \in \{1, \ldots, M\}$
- $u_0, v_0 \sim \mathbb{F}$

shares of the coefficients of h

Output

$$[f(0)]_i \quad [g(0)]_i \quad [h]_i$$

2. Servers check consistency

- Internal derivation of values $[f]_i,\ [g]_i$
- If all parties are honest: $f \cdot g = h$
- In case of malicious client: $\hat{h} \neq \hat{f} \cdot \hat{g}$

3. Polynomial validation

Goal: Detect with high probability a cheating client.

1. Sample a random value from the field.
2. Evaluate polynomials on the random value.
3. Get shares of $\sigma = r \cdot (\hat{f}(r) \cdot \hat{g}(r) - \hat{h}(r))$
4. Check the sum of those shares is 0.

If $\hat{h} \neq \hat{f} \cdot \hat{g}$ then the polynomial represented by σ is of degree at most $2M + 1$: with random evaluation, we detect the cheat with probability

$$\geq 1 - \frac{2M+1}{|\mathbb{F}|}$$

Beaver's Multi-Party Computation

Clients choose the triple

$$(a, b, c) \in \mathbb{F}^3$$

and send shares to the servers.

4. Final computation and verification

- Share the values of the shares of the output of Valid
- Check that they sum up to 1.

SNIP proof tuple

$$\pi = (f(0), g(0), h, \underbrace{a, b, c}_{\text{Beaver's triple}})$$

Efficiency

- Server-to-server communication cost same as local cost of circuit evaluation.
- Client-to-server communication linear in the size of the circuit.

Desired Properties of a useful SNIP

- **Correctness**: If all parties are honest, the servers will accept x.

- **Soundness**: If all servers are honest, and if Valid(x) != 1, then the servers will almost always reject x, no matter how the client cheats.

Formal definition

1. Run the adversary $\mathcal{A}$. For each server i, the adversary outputs a set of values:

    - $[x]_i \in \mathbb{F}^L$,
    - $([f(0)]_i, [g(0)]_i) \in \mathbb{F}^2$,
    - $[h]_i \in \mathbb{F}_{2M}[X]$ of degree at most $2M$, and
    - $([a]_i, [b]_i, [c]_i) \in \mathbb{F}^3$.

2. The *Master server* chooses a random $r \xleftarrow{\$} \mathbb{F}$. Each server compute their shares $[f]_i$ and $[g]_i$ as in the real protocol, and evaluate $[f(r)]_i$, $[r \cdot g(r)]_i$, $[r \cdot h(r)]_i$, and $[h(M)]_i$.

3. The servers compute $h(M) = \sum_i [h(M)]_i$, and

$$\sigma = r \cdot (f(r)g(r) - h(r)) + (c - ab)$$

4. We say that the adversary wins the game if:

$$h(M) = 1, \quad \sigma = 0, \quad \text{and} \quad \texttt{Valid}(x) \neq 1$$

Soundness:

$$\Pr[\mathcal{A} \text{ Wins}] \leq \frac{2M+1}{|\mathbb{F}|}$$

A Wins if:

$h(M) = 1, \quad \sigma = 0, \quad \text{and} \quad \texttt{Valid}(x) \neq 1$

$$\sigma = P(r)$$

$$P(t) = t \cdot Q(t) + (c - ab)$$

$$Q(t) = f(t)g(t) - h(t)$$

Case $fg \neq h$:

- P is a non-zero polynomial of degree at most $2M$+1.
- The choice of r is independent of (a, b, c) and Q, since the adversary must produce these values before r is chosen.

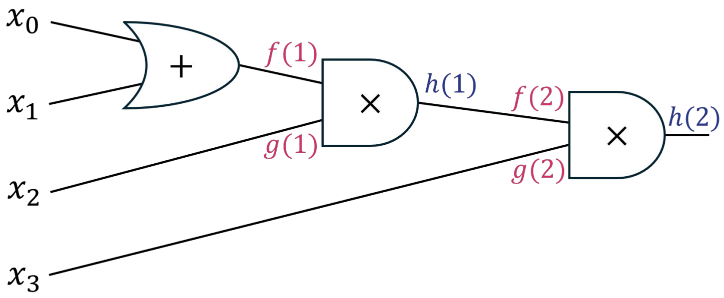
$\Rightarrow$ The choice of r is independent of P.

- P has at most $2M$+1 zeros in $\mathbf{F}$.

$\Rightarrow \Pr[\, P(r) = \sigma = 0 \,] \leq (2M+1)/|\mathbf{F}|$

Case $fg = h$:

- By induction: $h(M) = \text{Valid}(x)$ *(wlog assume that the circuit ends with a multiplication gate)*



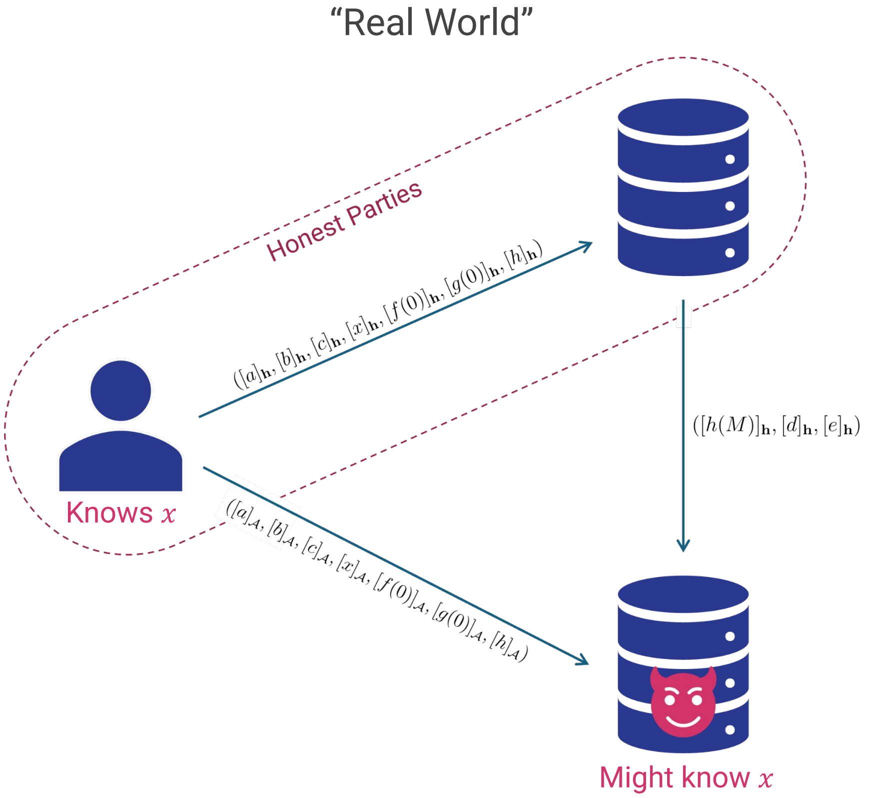
$\Rightarrow \Pr[\, h(M) = 1 \text{ and } \text{Valid}(x) \neq 1 \,] = 0$

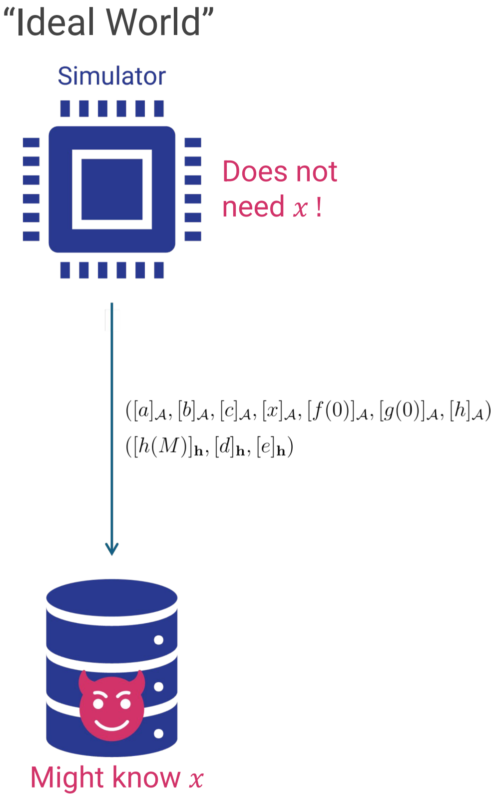
In both cases: $\Pr[\mathcal{A} \text{ Wins}] \leq \dfrac{2M+1}{|\mathbb{F}|}$

Desired Properties of a useful SNIP

- **Correctness**: If all parties are honest, the servers will accept x.
- **Soundness**: If all servers are honest, and if Valid(x) != 1, then the servers will almost always reject x, no matter how the client cheats.
- **Zero knowledge**: If the client and at least one server are honest, then the servers learn nothing about x, except that Valid(x) = 1.

Zero Knowledge - Proof Sketch





Zero Knowledge - Proof Sketch

In this game, the adversary tries to distinguish the two worlds.

- The simulator generates the initial adversary view **at random**.
- We can show that **the two views are distributed identically**.

(random sampling of r, $f(0)$ and $g(0)$ in the real world + hiding from secret sharing)

- Since the simulator does not know x:

⇒ Participating in the SNIP gives no extra information about x.

Affine-aggregatable encodings (AFEs)

So far we can:

- Compute private sums over client-provided data (Secret-sharing)
- Check arbitrary validation predicate against data (SNIP)

How can we compute more complex statistics ?

Idea: Encode private data to make the statistic computable over the sum of encoding.

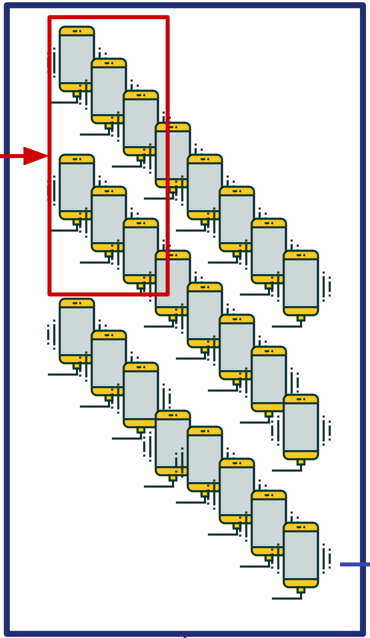
AFE concrete example

Computing the variance of b-bit integers: $\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$

- $\mathrm{Encode}(x) = (x, x^2, \beta_0, \beta_1, \ldots, \beta_{b-1})$ ← Secret-sharing
- $\mathrm{Valid}(\mathrm{Encode}(x)) = \left(x = \sum_{i=0}^{b-1} 2^i \beta_i\right) \wedge (x \cdot x = x^2) \wedge \bigwedge_{i=0}^{b-1} [\beta_i \cdot (\beta_i - 1) = 0]$ ← SNIP
- $(\sigma_0, \sigma_1) = \sum_{i=1}^{n} \mathrm{Trunc}_2(\mathrm{Encode}(x_i)) = \sum_{i=1}^{n} (x_i, x_i^2) = \left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} x_i^2\right)$
- $\mathrm{Decode}(\sigma) = \frac{1}{n}\left(\sigma_1 - (\sigma_0)^2\right)$

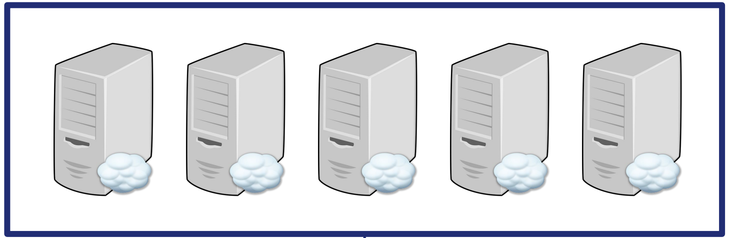
7. Prio Protocol - Setting

m malicious clients



$x_i \in \mathbb{D}$

n clients



s *honest* servers

$$f : \mathbb{D}^{n-m} \rightarrow \mathbb{A}$$

7. Prio Protocol - 4 steps

$$x_i \in \mathbb{D}$$

1. **Upload phase**

$$y_i \in \mathbb{F}^k$$

- input encoded using Affine-Aggregatable Encoding
- AFE encoded vector is split into secret shares
- SNIP proof is generated to prove data is well formed
- input shares and SNIP proof are sent to the servers

$$y_i \leftarrow Encode(x_i)$$

$$[y_i]_1, [y_i]_2, \ldots, [y_i]_s$$

$$y_i = [y_i]_1 + [y_i]_2 \ldots + [y_i]_s$$

7. Prio Protocol - 4 steps

2. Validation phase

- servers jointly verify the SNIP proofs received
  - rejects not well-formed submission
  - does not reveal information about the underlying data (except validity)
  - ensures robustness against malformed/malicious submissions

$$x_i \in \mathbb{D} \iff Valid(x_i) = 1$$

7. Prio Protocol - 4 steps

3. **Aggregation phase**

- each server initializes an accumulator to zero:
- for every valid client submission increments the accumulator
    - only truncated version of the client share carry necessary information

$$A_j \in F^{k'}$$

$$A_j \leftarrow 0$$

$$A_j \leftarrow A_j + Trunc_{k'}([y_i]_j) \in \mathbb{F}^{k'}$$

7. Prio Protocol - 4 steps

4. **Publish phase**

- servers publish their individual accumulator values $A_1, A_2, \dots, A_s$
- final aggregate is computed by summing accumulators
- final aggregate statistic obtained with AFE decoding: $Decode(\sigma) \in \mathbb{A}$

$$\sigma = \sum_{j=1}^{s} A_j = \sum_{i=1}^{n} Trunc_{k'} y_i$$

Protocol Security Properties

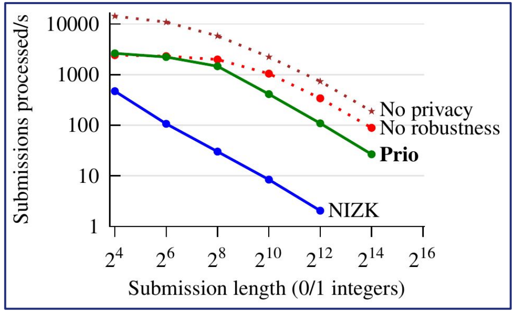
- **robustness** against malicious clients holds if:
  - SNIP construction is sound - malicious client submissions are detected via SNIPs
- **f-privacy,** only the final aggregate statistic is revealed, holds if:
  - one server is honest
  - AFE is f-private
  - SNIP is zero-knowledge
- **anonymity** holds if:
  - function f is symmetric - the order of inputs does not affect the output

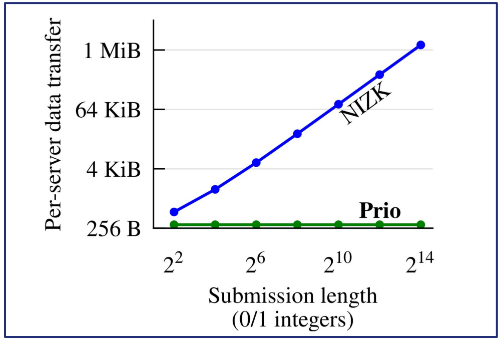
$$f(x_1, \ldots, x_{n-m}) = f(x'_1, \ldots, x'_{n-m})$$

$$(x'_1, \ldots, x'_{n-m}) = SORT(x_1, \ldots, x_{n-m})$$

8. Evaluation

- Prio Client performance:
  - *~0.03 sec* for a 100-integer submission on a workstation; *~0.1 sec* on a smartphone (2010-12 hardware).
- Prio Server throughput:
  - Outperforms NIZK-based scheme by 10x on average
  - Adding more server does not significantly affect throughput





	Workstation		Smartphone	
Field size	**87-bit**	**265-bit**	**87-bit**	**265-bit**
Multipl. in field (μs)	1.013	1.485	11.218	14.930
L = 10	0.003	0.004	0.017	0.024
L = 100	0.024	0.035	0.110	0.167
L = 1000	0.214	0.334	1.028	2.102

Time in seconds for a client to generate a Prio submission of L four-bit integers

9. Discussion - Limitations

- **Selective Denial-of-Service Attack**
- Intersection Attack
- *Robustness against faulty servers*
  - May be implemented but lowers the privacy guarantees
  - robust against *k* faulty servers (out of *s*) ⇒ protects privacy against at most *s-k-1* malicious servers

$$\cancel{x_1}, \cancel{x_2}, \ldots, x_i, \ldots, \cancel{x_n}$$

$$\downarrow$$

$$f(x_{honest}, x_{evil_1}, \ldots, x_{evil_m})$$

9. Discussion - Limitations

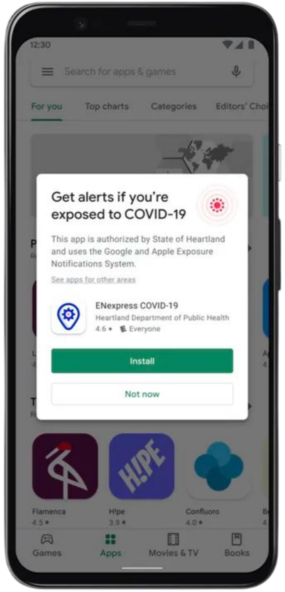
- Selective Denial-of-Service Attack
- **Intersection Attack**
- *Robustness against faulty servers*
    - May be implemented but lowers the privacy guarantees
    - robust against *k* faulty servers (out of *s*) ⇒ protects privacy against at most *s-k-1* malicious servers

$$x_1, x_2, \ldots, x_n \;\longrightarrow\; f(x_1, x_2, \ldots, x_n)$$

$$x'_1, x'_2, \ldots, x'_{n-1}, \cancel{x'_n} \;\longrightarrow\; f(x'_1, x'_2, \ldots, x'_{n-1})$$

9. Discussion - Deployments

- Many large-scale deployments since paper publication.
- During the *COVID-19* pandemic, *Apple* and *Google* introduced *Exposure Notification Privacy-preserving Analytics* to alert users about potential contact with individuals infected.
- Based on *Prio*.
- No one could access information about who received notifications or the identities of contacts.
- Aggregated insight were sent to *public health agencies*.



Conclusion

- Prio allows the aggregation of complexe statistics on private client data.
- Uses additive secret-sharing, SNIP and AFEs.
- More efficient and scalable than traditional protocols.
- Has many practical applications.

Thank you for your attention !

Appendix

Detailed computation for σ

Define the following values, where s is a constant representing the number of servers:

$$
\begin{aligned}
x &= \textstyle\sum_i [x]_i & a &= \textstyle\sum_i [a]_i \\
f(r) &= \textstyle\sum_i [f(r)]_i & b &= \textstyle\sum_i [b]_i \\
r \cdot g(r) &= \textstyle\sum_i [r \cdot g(r)]_i & c &= \textstyle\sum_i [c]_i \\
h(M) &= \textstyle\sum_i [h]_i(M) & d &= f(r) - a \\
& & e &= r \cdot g(r) - b
\end{aligned}
$$

$$
\begin{aligned}
\sigma &= \sum_i (de/s + d[b]_i + e[a]_i + [c]_i - [r \cdot h(r)]_i) \\
&= de + db + ea + c - r \cdot h(r) \\
&= (f(r) - a)(r \cdot g(r) - b) + (f(r) - a)b + (r \cdot g(r) - b)a + c - r \cdot h(r) \\
&= r \cdot (f(r)g(r) - h(r)) + (c - ab)
\end{aligned}
$$