




MPC for Privacy Preserving Measurement

Christopher Patton (Cloudflare)
ASCrypto 2023



1




MPC for Privacy Preserving Measurement

The tech industry **needs data to operate**

Use case	Data used (by whom)
----------	---------------------

2


MPC for Privacy Preserving Measurement 

The tech industry **needs data to operate**

Use case	Data used (by whom)
Browser telemetry	Which websites trigger bugs, distribute malware, etc. (browser vendor)

3

3


MPC for Privacy Preserving Measurement 

The tech industry **needs data to operate**

Use case	Data used (by whom)
Browser telemetry	Which websites trigger bugs, distribute malware, etc. (browser vendor)
Web analytics	Which features of a website app do users (dis)like the most (web developer)

4

4


MPC for Privacy Preserving Measurement 

The tech industry needs data to operate

Use case	Data used (by whom)
Browser telemetry	Which websites trigger bugs, distribute malware, etc. (browser vendor)
Web analytics	Which features of a website app do users (dis)like the most (web developer)
Connectivity	Which servers are are seeing connectivity issues (network operator)

5

5


MPC for Privacy Preserving Measurement 

The tech industry needs data to operate

Use case	Data used (by whom)
Browser telemetry	Which websites trigger bugs, distribute malware, etc. (browser vendor)
Web analytics	Which features of a website app do users (dis)like the most (web developer)
Connectivity	Which servers are are seeing connectivity issues (network operator)
Ad tech	Which ad campaigns are driving revenue (advertiser)

6

6


MPC for Privacy Preserving Measurement 

The tech industry **needs data to operate**

Use case	Data used (by whom)
Browser telemetry	Which websites trigger bugs, distribute malware, etc. (browser vendor)
Web analytics	Which features of a website app do users (dis)like the most (web developer)
Connectivity	Which servers are are seeing connectivity issues (network operator)
Ad tech	Which ad campaigns are driving revenue (advertiser)
AI	"Who" are my users (just about everyone these days)

7

7


MPC for Privacy Preserving Measurement 

The tech industry **collects more data than it needs**

Use case	Data used (by whom)	Data collected
Browser telemetry	Which websites trigger bugs, distribute malware, etc. (browser vendor)	Which web pages are users visiting (and what happens when they do)
Web analytics	Which features of a website app do users (dis)like the most (web developer)	What users are doing on your website
Connectivity	Which servers are are seeing connectivity issues (network operator)	Which servers are users connecting to (when a connection failure happens)
Ad tech	Which ad campaigns are driving revenue (advertiser)	Cross-site activity (saw an ad on one site and made a purchase on another)
AI	"Who" are my users (just about everyone these days)	Features (and labels) for (supervised) learning

8

8

MPC for Privacy Preserving Measurement 


Data minimization

Collect what you use and nothing more.

measurements	m_1, \dots, m_N	"Which users visited example.com on Thursday"
aggregate	$f(m_1, \dots, m_N)$	"How many users visited example.com on Thursday"

9

9

MPC for Privacy Preserving Measurement 

The PPM working group at IETF

- IETF: "Internet Engineering Task Force"
 - Specifies many of the protocols that undergird the Internet (IP, TCP, DNS, TLS, QUIC, HTTP, ...)
 - Open standardization process involving industry, academia, governments, *and you!*

10

10

MPC for Privacy Preserving Measurement CLOUDFLARE

The PPM working group at IETF

- PPM: "Privacy Preserving Measurement"
 - **2017**: Henry Corrigan-Gibbs, Dan Boneh propose [Prio](#)
 - **2018**: [Mozilla experiment with Prio](#) for origin telemetry
 - **2020**: [Google, Apple, and ISRG deploy Prio](#) alongside their COVID exposure notification apps to provide metrics for health authorities
 - **2021**: Birds-of-a-Feather session (IETF 112)
 - **2022 March**: First working group meeting (IETF 113)
 - **2022 May**: Working group adopts its first draft

datatracker.ietf.org/doc/draft-ietf-ppm-dap

draft-gpaw-priv-ppm 00 01
draft-ietf-ppm-dap 00 01 02 03 04 05 06 06

Oct 2021 Mar 2022 May 2022 Jul 2022 Sep 2022 Dec 2022 Mar 2023 Jul 2023 Aug 2023

11

11

MPC for Privacy Preserving Measurement CLOUDFLARE

The PPM working group at IETF


- PPM's goals
 - Lower the cost of data minimization
 - Engineering effort, financial burden, energy consumption, ...
 - Provide a deployment path for emerging **multi-party computation (MPC)** techniques
 - Progress in engineering dovetails with progress in cryptography ⇒ **we're building a world of MPC, and we need your help!**

Science

Engineering

12


12

MPC for Privacy Preserving Measurement 

- **Computing on secret-shared data**
- Security goals
- The tools we have
- The tools we're working on
- Practitioner's view of MPC
- How to contribute

13

13

MPC for Privacy Preserving Measurement 

Computing on secret shared data

measurement
m_1
m_2
...
m_i
...
m_N

God: compute the sum of the measurements

$f(m_1, \dots, m_N) = m_1 + \dots + m_N$

14

14

MPC for Privacy Preserving Measurement CLOUDFLARE

Computing on secret shared data

measurement
m_1
m_2
...
m_i
...
m_N

Secret sharing: Instead of sending m_i to the server in the clear, **shard** m_i into **secret shares** $[m_i]_1, [m_i]_2$ and send each share to a different server.

Goal: compute the sum of the measurements

$f(m_1, \dots, m_N) = m_1 + \dots + m_N$

15

15

MPC for Privacy Preserving Measurement CLOUDFLARE

Computing on secret shared data

measurement	first share	second share
m_1	$[m_1]_1 = m_1 - r_1$	$[m_1]_2 = r_1$
m_2	$[m_2]_1 = m_2 - r_2$	$[m_2]_2 = r_2$
...
m_i	$[m_i]_1 = m_i - r_i$	$[m_i]_2 = r_i$
...
m_N	$[m_N]_1 = m_N - r_N$	$[m_N]_2 = r_N$

Each Client shards its **measurement** into **input shares**


Each r_i sampled randomly from $[0..q]$

Each input share is indistinguishable from random \Rightarrow no information leakage!

Addition modulo q

16

16

r_1
MPC for Privacy Preserving Measurement 


Computing on secret shared data

measurement	first share	second share
m_1	$[m_1]_1 = m_1 - r_1$	$[m_1]_2 = r_1$
m_2	$[m_2]_1 = m_2 - r_1$	$[m_2]_2 = r_2$
...
m_i	$[m_i]_1 = m_i - r_i$	$[m_i]_2 = r_i$
...
m_N	$[m_N]_1 = m_N - r_N$	$[m_N]_2 = r_N$
	$[a]_1 = [m_1]_1 + \dots + [m_N]_1$	

First Aggregator sums up its input shares to get its **aggregate share**

17

17

MPC for Privacy Preserving Measurement 

Computing on secret shared data

measurement	first share	second share
m_1	$[m_1]_1 = m_1 - r_1$	$[m_1]_2 = r_1$
m_2	$[m_2]_1 = m_2 - r_1$	$[m_2]_2 = r_2$
...
m_i	$[m_i]_1 = m_i - r_i$	$[m_i]_2 = r_i$
...
m_N	$[m_N]_1 = m_N - r_N$	$[m_N]_2 = r_N$
	$[a]_1 = [m_1]_1 + \dots + [m_N]_1$	$[a]_2 = [m_1]_2 + \dots + [m_N]_2$

Second Aggregator sums up its input shares to get its **aggregate share**

18

18

MPC for Privacy Preserving Measurement CLOUDFLARE

Computing on secret shared data

measurement	first share	second share
m_1	$[m_1]_1 = m_1 - r_1$	$[m_1]_2 = r_1$
m_2	$[m_2]_1 = m_2 - r_1$	$[m_2]_2 = r_2$
...
m_i	$[m_i]_1 = m_i - r_i$	$[m_i]_2 = r_i$
...
m_N	$[m_N]_1 = m_N - r_N$	$[m_N]_2 = r_N$
	$[a]_1 = [m_1]_1 + \dots + [m_N]_1$	$[a]_2 = [m_1]_2 + \dots + [m_N]_2$

Collector sums up aggregate shares to get **aggregate result**
 $[q_1] + [q_2] = f(m_1, \dots, m_N)$


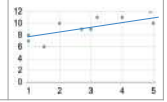
19

19

MPC for Privacy Preserving Measurement CLOUDFLARE

Computing on secret shared data

- Affine-aggregatable encodings ([Prio](#))
 - Many **aggregation functions** can be represented as a linear function of (some encoding of) the measurements

type	measurements	aggregate result
Count	1, 1, 0, 1, 0, 1	5
Mean, standard deviation	182, 160, 190, 170, 175	175, 11
Histogram	-7 \Rightarrow [1, 0, 0] 23 \Rightarrow [0, 1, 0] 45 \Rightarrow [0, 1, 0] 59 \Rightarrow [0, 0, 1]	
Linear regression	(1, 7), (2, 10), (3, 9), (4, 11), ..., (5, 10)	

20


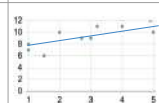
20

MPC for Privacy Preserving Measurement CLOUDFLARE

Computing on secret shared data

- Affine-aggregatable encodings (Prio)
 - Many **aggregation functions** can be represented as a linear function of (some encoding of) the measurements

This simple approach is not sufficient: need interaction.

type	measurements	aggregate result
Count	1, 1, 0, 1, 0, 1	5
Mean, standard deviation	182, 160, 190, 170, 175	175, 11
Histogram	-7 ⇒ [1, 0, 0] 23 ⇒ [0, 1, 0] 45 ⇒ [0, 1, 0] 59 ⇒ [0, 0, 1]	
Linear regression	(1, 7), (2, 10), (3, 9), (4, 11), ..., (5, 10)	

21

21

MPC for Privacy Preserving Measurement CLOUDFLARE

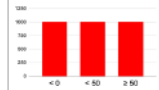
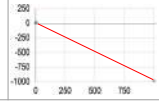
Need for interactivity: input validation

Secret sharing of 1 (mod a 128-bit prime):

- 2042029539322534402155356
25362756584204
- 1360794129886850227313301
48005144182006

Secret sharing of 1132410957436894378792879891027759
56821:

- 1706516663212822713104429
85250513842899
- 2828717963433456295157107
77220162880131

type	measurements	aggregate result
Count	1, 1, 0, 1, 0, 999	1002
Mean, standard deviation	182, 160, 190, 170, 999	340, 368
Histogram	-7 ⇒ [1, 0, 0] 23 ⇒ [0, 1, 0] 45 ⇒ [0, 1, 0] [999, 999, 999]	
Linear regression	(1, 7), (2, 10), (3, 9), (4, 11), ..., (999, -999)	

22

22

MPC for Privacy Preserving Measurement CLOUDFLARE

Need for interactivity: non-linear computation

- Not all functions we'd like to compute are linear
 - Heavy hitters:** Among the measurements uploaded by Clients, find the subset that were uploaded at least t times (for some threshold t)

```
def heavy_hitters(measurements: list[str], t: int) -> set[str]:
    hh = defaultdict(lambda: 0)
    for measurement in measurements:
        hh[measurement] += 1
    return set(map(lambda x: x[0], filter(lambda x: x[1] >= t, hh.items())))

# Test
assert heavy_hitters(['hi', 'there', 'oh', 'hi'], 2) == {'hi'}
```

23

23

MPC for Privacy Preserving Measurement CLOUDFLARE

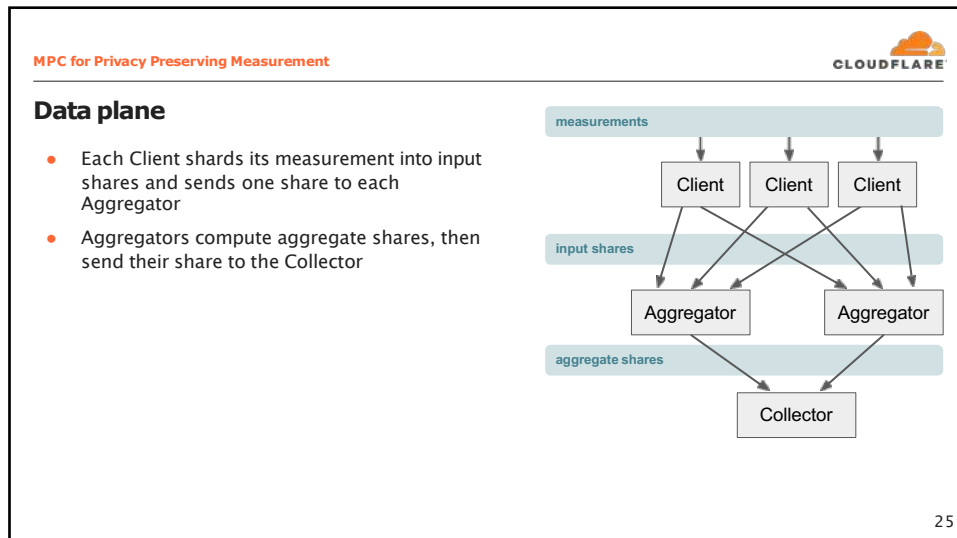
Data plane

- Each Client shards its measurement into input shares and sends one share to each Aggregator

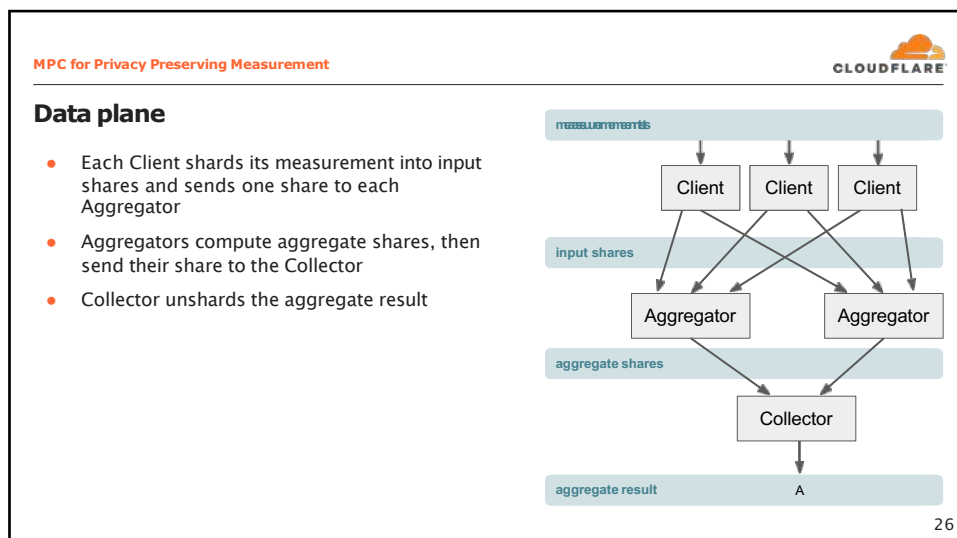
The diagram illustrates the data plane flow. At the top, a light blue bar labeled 'measurements' has three downward arrows pointing to three separate boxes labeled 'Client'. Below the clients is another light blue bar labeled 'input shares'. From each of the three 'Client' boxes, two arrows point to the 'input shares' bar, indicating that each client's measurement is split into two shares. Finally, from the 'input shares' bar, two arrows point to two boxes labeled 'Aggregator', showing that each aggregator receives one share from every client.

24

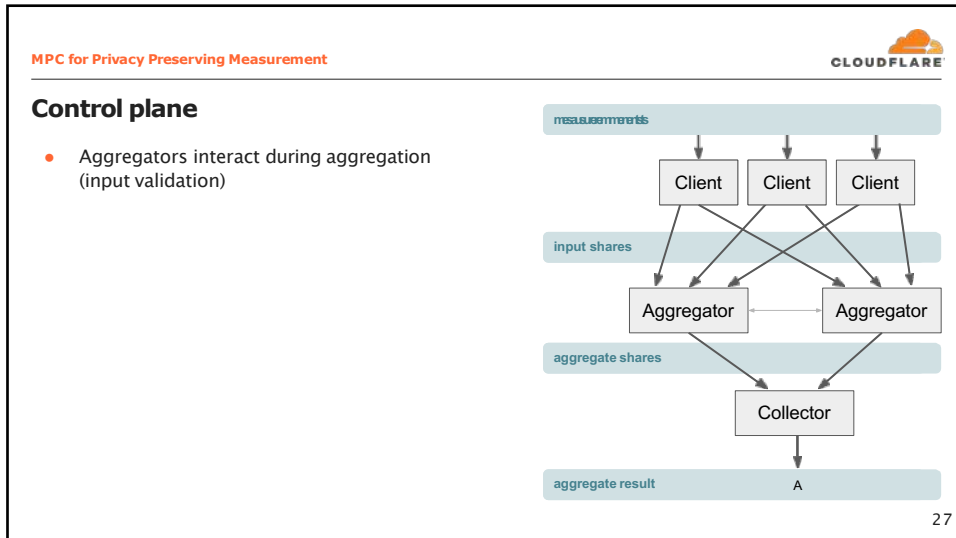
24



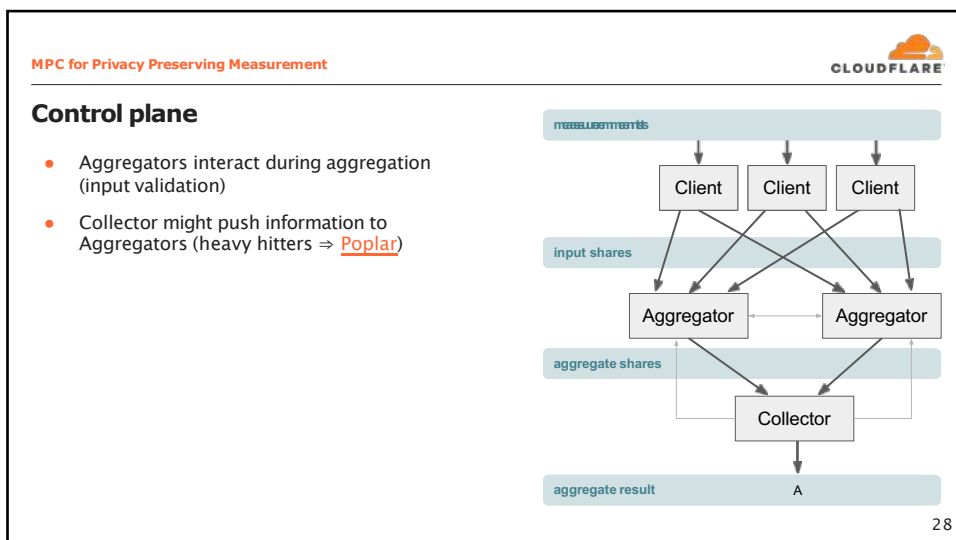
25



26



27



28

MPC for Privacy Preserving Measurement CLOUDFLARE

Control plane

- Aggregators interact during aggregation (input validation)
- Collector might push information to Aggregators (heavy hitters ⇒ [Poplar](#))
- Collector might push information to Clients (federated learning ⇒ [dpsa4fi](#))

The diagram illustrates the control plane architecture. At the top, a 'measurements' layer feeds into three 'Client' boxes. These clients send 'input shares' to two 'Aggregator' boxes. The aggregators then send 'aggregate shares' to a 'Collector' box. The collector sends back information to both aggregators and also pushes information to the clients. Finally, the collector outputs an 'aggregate result' labeled 'A'. The diagram is divided into four horizontal layers: measurements, input shares, aggregate shares, and aggregate result.

29

29

MPC for Privacy Preserving Measurement CLOUDFLARE

Control plane

- Aggregators interact during aggregation (input validation)
- Collector might push information to Aggregators (heavy hitters ⇒ [Poplar](#))
- Collector might push information to Clients (federated learning ⇒ [dpsa4fi](#))

When thinking about the control plane, we need to carefully consider information leakage!

This diagram is identical to the one on slide 29, showing the control plane architecture with Clients, Aggregators, and a Collector, and layers for measurements, input shares, aggregate shares, and aggregate results.

30

30

MPC for Privacy Preserving Measurement CLOUDFLARE

- Computing on secret-shared data
- **Security goals**
 - The tools we have
 - The tools we're working on
 - Practitioner's view of MPC
 - How to contribute

31

31

MPC for Privacy Preserving Measurement CLOUDFLARE

Privacy

- Threat model
 - All Clients and one Aggregator are honest
 - Collector and one Aggregator are controlled by the attacker
 - Attacker controls the network (except transmission of input shares to the honest Aggregator)

The diagram illustrates the privacy threat model in MPC for Privacy Preserving Measurement. It shows a flow of data and control across several stages:

- measurements**: The initial data input.
- input shares**: Data is distributed to three **Client** nodes.
- aggregate shares**: Data is processed by two **Aggregator** nodes.
- Collector**: A central node that receives data from the aggregators.
- aggregate result**: The final output of the process.

Red arrows indicate the flow of data, while black arrows indicate the flow of control or information. The diagram shows that the Collector and one Aggregator are controlled by the attacker, while all Clients and one Aggregator are honest.

32

32

MPC for Privacy Preserving Measurement CLOUDFLARE

Privacy

- Threat model
 - All Clients and one Aggregator are honest
 - Collector and one Aggregator are controlled by the attacker
 - Attacker controls the network (except transmission of input shares to the honest Aggregator)
- Security goal
 - A computationally-bounded attacker's view of the protocol execution is efficiently simulatable given the aggregate result*

Def.: For every efficient attacker A there is an efficient **simulator** S such that $\text{View}_A(m_1, \dots, m_n)$ and $S(m_1, \dots, m_n)$ are computationally indistinguishable.

*Depending on the scheme there may be additional information leakage. 33


33

MPC for Privacy Preserving Measurement CLOUDFLARE

Differential privacy

- Threat model
 - All Clients and one Aggregator are honest
 - Collector and one Aggregator are controlled by the attacker
 - Attacker controls the network (except transmission of input shares to the honest Aggregator)
- Security goal
 - A computationally-bounded attacker's view of the protocol execution is efficiently simulatable ~~given the aggregate result~~ **by a differentially private simulator**

Def. [DMNS06]: A randomized algorithm is **differentially private (DP)** if its output does not depend "too much" on the value of any input of its inputs.



[DMNS06] Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". TCC 2006 34

*Depending on the scheme there may be additional information leakage.

34

MPC for Privacy Preserving Measurement CLOUDFLARE

Differential privacy

Achieve DP by adding carefully calibrated noise into the aggregate result.

Def. [DMNS06]: A randomized algorithm is **differentially private (DP)** if its output does not depend "too much" on the value of any input of its inputs.

[DMNS06] Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". TCC 2006

35

35

MPC for Privacy Preserving Measurement CLOUDFLARE

Robustness

- Threat model
 - Aggregators and Collector execute the protocol correctly
 - Attacker controls a fraction of Clients
- Security goal
 - Collector correctly computes the aggregate result of honest Clients' measurements

36

36

Different threat model for different security goals

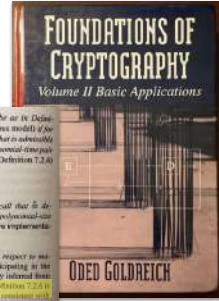
- Privacy: malicious Aggregator and Collector (one Aggregator is semi-honest)
- Robustness: malicious Clients (Aggregators and Collector are semi-honest)
 - Malicious Aggregator can change the result (break robustness), but won't violate privacy by doing so

Definition 7.2.6 (security in the malicious model). Let f and Π be as in Definition 7.2.2. Protocol Π is said to securely compute f in the malicious model if for every probabilistic polynomial-time pair of adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that is adversarially secure for f (of Definition 7.2.5), there exist a probabilistic polynomial-time simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ that is adversarial for the ideal model (of Definition 7.2.4) such that


$$\text{PRIM}_{\mathcal{A}, \Pi, \mathcal{S}}(x, y) \approx \text{PRIM}_{\mathcal{S}, \mathcal{A}, f}(x, y)$$

where $x, y \in \{0, 1\}^*$ such that $|x| = |y|$ and $|f| = \text{poly}(|x|)$. Recall that \approx denotes computational indistinguishability by (non-uniform) families of polynomial-time circuits. When the context is clear, we sometimes refer to Π as a secure implementation of f .

One important property that Definition 7.2.6 implies is privacy with respect to malicious adversaries. That is, all that an adversary can learn by participating in the protocol, while using an arbitrary (cheating) strategy, can be essentially inferred from the corresponding output alone. Another property that is implied by Definition 7.2.6 is correctness, which means that the output of the honest party must be consistent with an input pair in which the element corresponding to the honest party equals the party's actual input. Furthermore, the element corresponding to the adversary must be (almost) independent of the honest party's input. We stress that both properties are implied by Definition 7.2.6, but the latter is not implied by correctness alone.



- Computing on secret-shared data
- Security goals
- **The tools we have**
- The tools we're working on
- Practitioner's view of MPC
- How to contribute


MPC for Privacy Preserving Measurement 

Fully linear proofs [BBCG+19]

- Conventional zero-knowledge proofs:
 - Prover publishes commitment to X and proof Π that X is "valid"
 - Verifier uses Π to check validity of the committed value
- Fully linear proofs (FLPs):
 - Same idea except the data is *secret-shared* instead of committed

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019. 39

39

MPC for Privacy Preserving Measurement 

Fully linear proofs [BBCG+19]

Syntax:

```

 $\Pi := \text{Prove}(X)$  // proof generation
 $V := \text{Query}(X, \Pi; \mathcal{Q})$  // query generation
 $d := \text{Decide}(V)$  // decision

```

Full linearity: $\text{Query}(X, \Pi; \mathcal{Q})$ is equal to:

- Split Π, X into shares $[\Pi]_i, [X]_i$ for all i
- $[V]_i := \text{Query}([X]_i, [\Pi]_i; \mathcal{Q})$ for all i
- Return $[V]_1 + \dots + [V]_s$

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019. 40

40

MPC for Privacy Preserving Measurement CLOUDFLARE

Fully linear proofs [BBCG+19]

Syntax:

$\Pi := \text{Prove}(X)$ // proof generation
 $V := \text{Query}(X, \Pi; \hat{q})$ // query generation
 $d := \text{Decide}(V)$ // decision

Full linearity: $\text{Query}(X, \Pi; \hat{q})$ is equal to:

- Split Π, X into shares $[X]_i, [\Pi]_i$ for all i
- $[V]_i := \text{Query}([X]_i, [\Pi]_i; \hat{q})$ for all i
- Return $[V]_1 + \dots + [V]_s$

Prio*

```

graph TD
    X --> Client
    Client -- "[X]_1, [Pi]_1" --> Agg1[Aggregator]
    Client -- "[X]_2, [Pi]_2" --> Agg2[Aggregator]
    Agg1 -- "[V]_1" --> Agg2
    Agg2 -- "[V]_2" --> Agg1
    Agg1 --> d1[d]
    Agg2 --> d2[d]
  
```

*Slightly different from the original paper [CGB17], but reflects the [draft specification](#).

[CGB17] Boneh and Corrigan-Gibbs. "Prio: Private, Robust, and Scalable Computation of Aggregate." NSDI 2017.
 [BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019.

41

41

MPC for Privacy Preserving Measurement CLOUDFLARE

Fully linear proofs [BBCG+19]

Syntax:

$\Pi := \text{Prove}(X)$ // proof generation
 $V := \text{Query}(X, \Pi; \hat{q})$ // query generation
 $d := \text{Decide}(V)$ // decision

Full linearity: $\text{Query}(X, \Pi; \hat{q})$ is equal to:

- Split Π, X into shares $[X]_i, [\Pi]_i$ for all i
- $[V]_i := \text{Query}([X]_i, [\Pi]_i; \hat{q})$ for all i
- Return $[V]_1 + \dots + [V]_s$

Honest verifier zero-knowledge: There is a simulator \mathcal{S} whose output is statistically close to the following experiment (for all X):

- $\Pi := \text{Prove}(X)$
- Choose q at random
- $V := \text{Query}(X, \Pi; \hat{q})$
- Return (V, \hat{q})


Soundness: For all invalid inputs $X \notin \mathcal{L}$ and proofs Π , the probability that the following experiment outputs `valid` is small:

- Choose q at random
- $V := \text{Query}(X, \Pi; \hat{q})$
- Return $\text{Decide}(V)$

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019.

42

42


MPC for Privacy Preserving Measurement 

Fully linear proofs [BBCG+19]

- Constructing FLPs
 - Define validity via a **circuit** C : If $X \in \mathcal{L}$, then $C(X)=0$;
but if $X \notin \mathcal{L}$ then $C(X) \neq 0$

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019. 43

43

MPC for Privacy Preserving Measurement 

Fully linear proofs [BBCG+19]


- Constructing FLPs
 - Define validity via a **circuit** C : If $X \in \mathcal{L}$, then $C(X)=0$;
but if $X \notin \mathcal{L}$ then $C(X) \neq 0$

```
def counter(x: F) -> F:
  return x * (x-1)

# Test
assert counter(0) == 0
assert counter(1) == 0
assert counter(999) != 0
```

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019. 44

44

MPC for Privacy Preserving Measurement 

Fully linear proofs [BBCG+19]

- Constructing FLPs
 - Define validity via a (randomized) **circuit** C : If $X \in \mathcal{L}$ then $C(X)=0$; but if $X \notin \mathcal{L}$ then $C(X) \neq 0$ (w.h.p.)

```
def counter(x: F) -> F:
  return x * (x-1)


# Test
assert counter(0) == 0
assert counter(1) == 0
assert counter(999) != 0
```

```
def histogram(x: list[F], r: list[F]) -> F:
  rng_chk = sum(r[0]**i * x[i] * (x[i]-1) for i in range(len(x)))
  sum_chk = sum(x) * (sum(x)-1)
  return rng_chk + r[1]*sum_chk

# Test
assert histogram([0, 0, 0, 0], rand_vec(2)) == 0
assert histogram([0, 0, 1, 0], rand_vec(2)) == 0
assert histogram([0, 0, 999, 0], rand_vec(2)) != 0
assert histogram([1, 0, 1, 0], rand_vec(2)) != 0
```

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019. 45

45

MPC for Privacy Preserving Measurement 

Fully linear proofs [BBCG+19]

- Constructing FLPs
 - Define validity via a (randomized) **circuit** C : If $X \in \mathcal{L}$ then $C(X)=0$; but if $X \notin \mathcal{L}$ then $C(X) \neq 0$ (w.h.p.)

```
def counter(x: F) -> F:
  return
  # Test
  assert counter(0) == 0
  assert counter(1) == 0
  assert counter(999) != 0
```


```
def histogram(x: list[F], r: list[F]) -> F:
  rng_chk = sum(r[0]**i * x[i] * (x[i]-1) for i in range(len(x)))
  sum_chk = sum(x) * (sum(x)-1)
  return r[1] * rng_chk + r[1]**2 * sum_chk

# Test
assert histogram([0, 0, 0, 0], rand_vec(2)) == 0
assert histogram([0, 0, 1, 0], rand_vec(2)) == 0
assert histogram([0, 0, 999, 0], rand_vec(2)) != 0
assert histogram([1, 0, 1, 0], rand_vec(2)) != 0
```

Problem: circuits usually involve non-linear operations \Rightarrow can't compute these on secret shared data

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019. 46

46

MPC for Privacy Preserving Measurement 

Fully linear proofs [BBCG+19]

- Constructing FLPs
 - Define validity via a (randomized) **circuit** C : If $X \in \mathcal{L}$ then $C(X)=0$; but if $X \notin \mathcal{L}$ then $C(X) \neq 0$ (w.h.p.)
 - Proof Π encodes a polynomial p for which $p(i)$ is the output of the i -th non-linear operation

```
def counter(x: F) -> F:
  return

# Test
assert counter(0) == 0
assert counter(1) == 0
assert counter(999) != 0
```


Observation:
Polynomial evaluation is linear!

```
def histogram(x: list[F], r: list[F]) -> F:
  rng_chk = sum(r[0]**i *          for i in range(len(x)))
  sum_chk =
  return r[1] * rng_chk + r[1]**2 * sum_chk

# Test
assert histogram([0, 0, 0, 0], rand_vec(2)) == 0
assert histogram([0, 0, 1, 0], rand_vec(2)) == 0
assert histogram([0, 0, 999, 0], rand_vec(2)) != 0
assert histogram([1, 0, 1, 0], rand_vec(2)) != 0
```

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019. 47

47

MPC for Privacy Preserving Measurement 

Fully linear proofs [BBCG+19]

- Constructing FLPs
 - Define validity via a (randomized) **circuit** C : If $X \in \mathcal{L}$ then $C(X)=0$; but if $X \notin \mathcal{L}$ then $C(X) \neq 0$ (w.h.p.)
 - Proof Π encodes a polynomial p for which $p(i)$ is the output of the i -th non-linear operation
 - Verifier(s):
 - (Each) Verifier evaluates (its share of) $C(X)$ using (its share of) p
 - Run probabilistic test to check that p is well-formed (using q)

```
def counter(x: F) -> F:
  return

# Test
assert counter(0) == 0
assert counter(1) == 0
assert counter(999) != 0
```

[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019. 48

48

MPC for Privacy Preserving Measurement CLOUDFLARE

Distributed point functions [GI14]

- Point function: $f(\alpha)=\beta$ and $f(X)=0$ for all $X \neq \alpha$
 - Distributed point function: secret-sharing of a point function
 - $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
 - $[f(X)]_i = \text{Eval}(P, K_i, X)$ for all X, i
 - Use case: aggregate by **label**
 - Measurement is a pair (α, β) where α is the Client's label (user agent, geolocation, etc.) and β is the Client's contribution to the aggregate result

```
def agg_by_label(measurements: list[tuple[str, int]], x: str) -> int:
    return sum(map(lambda m: m[1], filter(lambda m: m[0] == x, measurements)))

# Test
assert agg_by_label([('EC', 1), ('US', 13), ('EC', 99)], 'EC') == 100
```

[GI14] Gilboa and Ishai. "Distributed Point Functions and their Applications." EUROCRYPT 2014. 49

49

MPC for Privacy Preserving Measurement CLOUDFLARE

Incremental distributed point functions [BBCG+21]

- Incremental point function: $f(l)=\beta$ for any **prefix** l of α and $f(l)=0$ otherwise
 - Incremental DPF: secret-sharing of an incremental point function
 - $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
 - $[f(l)]_i = \text{Eval}(P, K_i, l)$ for all l, i
 - Use case: ϵ -heavy-hitters (Poplar)
 - Client: let α be the measurement (a bit string) and let $\beta=1$
 - Aggregators count how many begin with a given prefix l
 - Traverse the **prefix tree** of the measurement to the leaves with prefix count at least t

ia.cr/2021/017

[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 50

50


MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?



[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 51

51


MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?



[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 52

52

MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?

[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 53

53

MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?

[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 54

54


MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?



[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 55

55

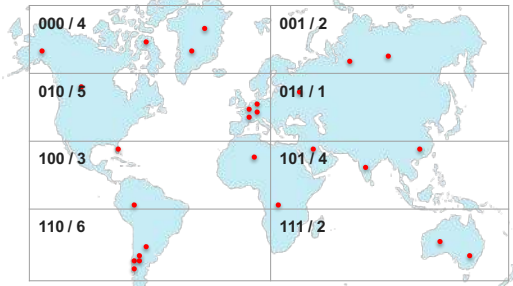
MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?



[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 56

56

MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?

Candidate prefixes: 000, ~~004~~, 010, ~~044~~, ~~400~~, 101, 110, ~~444~~ / threshold: 4

[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 57

57

MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?

Candidate prefixes: 0000, 0001, 0100, 0101, 1010, 1011, 1100, 1101 / threshold: 4

[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 58

58

MPC for Privacy Preserving Measurement CLOUDFLARE

Poplar [BBCG+21]

Syntax:

- $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
- $[\beta]_i = \text{Eval}(P, K_i, J)$ for all i

E.g.: From which regions are users experiencing high latency?

Candidate prefixes: ~~0000~~, ~~0001~~, ~~0100~~, 0101, ~~1010~~, ~~1011~~, 1100, ~~1101~~ / threshold: 4

[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 59

59

MPC for Privacy Preserving Measurement CLOUDFLARE

Constructing IDPFs

- P, K_1 and P, K_2 are concise representations of binary trees: α -path nodes are *secret shares* of β , and off-path nodes are *equal*

[BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters." IEEE S&P 2021. 60

60

MPC for Privacy Preserving Measurement CLOUDFLARE

- Computing on secret-shared data
- Security goals
- The tools we have
- **The tools we're working on**
- Practitioner's view of MPC
- How to contribute

61

61

MPC for Privacy Preserving Measurement CLOUDFLARE

Verifiable Incremental distributed point functions

- IDPF with verifiability of **one-hotness**
 - $(P, K_1, K_2) := \text{Gen}(\alpha, \beta)$
 - $(f(l_1), \dots, f(l_p), \pi) = \text{Eval}(P, K_1, \mathbb{D})$ for all $l = (l_1, \dots, l_p), i$
 - $\pi_i = \pi_j$ implies $f(l_i), \dots, f(l_j)$ is a one-hot vector
- Also need to verify that the non-zero value is **in-range**
 - [MST23] solve this in their protocol (PLASMA) for the special case that $\beta=1$; **what about the general case?**

Vector $\mathbf{V}=(V_1, V_2, \dots)$ is **one-hot** if at most value V_i is non-zero, e.g.:


0000000000, 0000000000

```

graph TD
    Root[ ] --- 0[0]
    Root --- 1[1]
    0 --- 00[00]
    0 --- 01[01]
    1 --- 10[10]
    1 --- 11[11]
    00 --- 5719[5719]
    01 --- 22939[22939]
    10 --- 32019[32019]
    11 --- 23487[23487]
    style 32019 fill:#f96
    
```

[MST23] Mouris et al. "PLASMA: Private, Lightweight Aggregated Statistics against Malicious Adversaries." ePrint 2023/080. 62

62

MPC for Privacy Preserving Measurement 


Function secret sharing

- FSS [BG116]: split f into shares such that $[f(X)]_1, \dots, [f(X)]_n$ can be evaluated for any X
 - Possible to construct efficient schemes for specific classes of functions (e.g., (incremental) point functions, decision trees, ...)
 - **Q1:** Efficient and generic approach for transforming privacy-only FSS to verifiable FSS?
 - **Arithmetic sketching** [BBCG+23] (generalizes sketching scheme from Poplar for achieving robustness with IDPFs)

[BG116] Boyle et al. "Function Secret Sharing: Improvements and Extensions." CCS 2016.
 [BBCG+23] Boneh et al. "Arithmetic Sketching." CRYPTO 2023.

63

63

MPC for Privacy Preserving Measurement 


Boolean-to-arithmetic conversion

- Common use case for Prio: aggregating vectors of counters
 - Prio+ [ABJ+22]: Clients send XOR shares of each counter; Aggregators interact to convert the shares into a finite field \Rightarrow *huge* improvement for Client
 - **Q2:** Private boolean-to-arithmetic conversion in the presence of a malicious server? (Authors seem to only claim semi-honest privacy.)

[ABJ+22] Addanki et al. "Prio+: Privacy Preserving Aggregate Statistics via Boolean Shares." SCN 2022.

64


64

MPC for Privacy Preserving Measurement 

Sorting

- Sort rows of a secret-shared database by a key
 - Use case: **last-touch attribution (IPA)**
 - For each purchase, find the most recent ad impression that can be linked to it \Rightarrow figure out which ad impressions are most effective
 - 3-party, honest majority protocol of [CHI+19] is being evaluated.
 - Q3:** Is a 2-party protocol possible (with our requirements)?


match key	time	source	trigger
89b0	12:45	c54c	0000
2d14	13:10	c54c	0000
89b0	14:44	3d32	0000
89b0	13:37	0000	153e



match key	time	source	trigger
89b0	14:44	3d32	0000
89b0	13:37	0000	153e
89b0	12:45	c54c	0000
2d14	13:10	c54c	0000

[CHI+19] Chida et al. "An Efficient Secure Three-Party Sorting Protocol with an Honest Majority". ePrint 2019/695. 65

65

MPC for Privacy Preserving Measurement 

Standardized DP mechanisms

- Bridging the DP and MPC communities:
 - Secret-sharing the noise [EIKN22, KKL+23]
 - Algorithms for sampling from non-uniform distributions (e.g., discrete Gaussian [CKS20])
 - Collective experience with privacy/utility trade-off

[EIKN21] Eriguchi et al. "Efficient Noise Generation Protocols for Differentially Private Multiparty Computation." FC 2021.
 [KKL+23] Keeler et al. "DPrio: Efficient Differential Privacy with High Utility for Prio." PETS 2023.
 [CKS20] Canonne et al. "The Discrete Gaussian for Differential Privacy." NuerIPS 2020. 66

66

MPC for Privacy Preserving Measurement CLOUDFLARE

- Computing on secret-shared data
- Security goals
- The tools we have
- The tools we're working on
- **Practitioner's view of MPC**
- How to contribute

67

67

MPC for Privacy Preserving Measurement CLOUDFLARE

Number of parties

- More parties \Rightarrow increased complexity
 - 2 parties is ideal: fits neatly into client-server communication pattern
 - 3 parties is probably workable
 - ≥ 4 parties is untested (for PPM)
- Redundancy doesn't seem super useful (so far)
 - In theory, [Prio](#) allows any number of Aggregators
 - Simple way to get honest-majority robustness: re-run 2-party protocol with each pair of three parties [[BBCG+19](#), [MST23](#)]

68

[[BBCG+19](#)] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019.
 [[MST23](#)] Mouris et al. "PLASMA: Private, Lightweight Aggregated Statistics against Malicious Adversaries." ePrint 2023/080.

68

MPC for Privacy Preserving Measurement CLOUDFLARE

Number of rounds

- Carefully consider how much state is required (and for how long)
 - In the client-server setting:
 - 1-round MPC (e.g., FLP verification) might be *stateless* for the server!
 - For $r \geq 2$, the protocol is necessarily stateful \Rightarrow difference between r -round and $(r+1)$ -round MPC is negligible

69

69


MPC for Privacy Preserving Measurement CLOUDFLARE

Malicious versus semi-honest security

- This dichotomy is an artifact of the textbook definition of "secure computation"
- We would like malicious security, *but not at any cost* (whether more parties, more rounds, higher bandwidth, or more CPU is required needs to be considered)
- Set aside secure computation and think about the attacker's incentives:
 - We *must have* privacy against malicious Aggregators
 - We *don't always need* robustness against malicious Aggregators

70


70

MPC for Privacy Preserving Measurement 

- Computing on secret-shared data
- Security goals
- The tools we have
- The tools we're working on
- Practitioner's view of MPC
- **How to contribute**

71

71

MPC for Privacy Preserving Measurement 

How to contribute

- Join the mailing list: <https://www.ietf.org/mailman/listinfo/Ppm>
- Join #ppm in the IETF slack: <https://ietf.slack.com/>
- Drafts:
 - DAP: <https://datatracker.ietf.org/doc/draft-ietf-ppm-dap/>
 - VDAF: <https://datatracker.ietf.org/doc/draft-irtf-cfrg-vdaf/>
 - Individual drafts in progress for DP, dealing with Sybil attacks, other approaches, ...


72

72

MPC for Privacy Preserving Measurement CLOUDFLARE

Who am I?

- PhD from University of Florida (under Tom Shrimpton) in 2020
 - Practice-oriented provable security [Rog09]: bridging provable and real-world security of cryptographic protocols
- Joined Cloudflare in 2020
 - Cryptography engineering: design, analysis, specification, implementation, and deployment of cryptographic protocols
- **I am not an MPC expert!**
 - Please interrupt with questions, or to correct the record



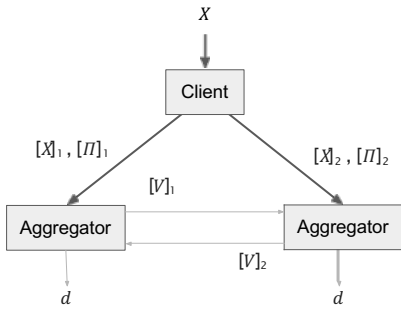
73

73

MPC for Privacy Preserving Measurement CLOUDFLARE

Fully linear proofs [BBCG+19]


- Prio*
 - Client on input X :
 - $\Pi := \text{Prove}(X)$ // proof generation
 - Split Π, X into shares $[\Pi]_i, [X]_i$
 - Send $[X]_i, [\Pi]_i$ to Aggregator i
 - Aggregators sample q at random
 - Aggregator i on $[X]_i, [\Pi]_i$:
 - $[V]_i := \text{Query}([X]_i, [\Pi]_i; q)$
 - Broadcast $[V]_i$
 - Aggregator i on $[V]_1, \dots, [V]_N$:
 - Return $d := \text{Decide}([V]_1 + \dots + [V]_N)$



*Slightly different from the original paper [CGB17], but reflects the [draft specification](#).

74

74

MPC for Privacy Preserving Measurement


Differential privacy

- Threat model
 - All Clients and one Aggregator are honest
 - Collector and one Aggregator are controlled by the attacker
 - Attacker controls the network (except transmission of input shares to the honest Aggregator)
- Security goal
 - A computationally-bounded attacker's view of the protocol execution is efficiently simulatable *given the aggregate result* by a differentially private simulator

Def. [DMNS06]: A randomized algorithm is **differentially private (DP)** if its output does not depend "too much" on the value of any input of its inputs.

Simulator S gets the measurements as input and outputs a simulation of the attacker's view: it is ϵ -DP if for all τ and **neighboring** M, M' :

$$\Pr[S(M) = \tau] \leq e^\epsilon \cdot \Pr[S(M') = \tau]$$

Achieve DP by adding carefully calibrated noise into the aggregate result.

*Depending on the scheme there may be additional information leakage.

75