Computing on *your* data with MPC

Christopher Patton
CAW 2024

CLOUDFLARE

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 do users (dis)like the most (web developer)
Connectivity	Connectivity issues between client and server (network operator)
Ad tech	Which ad campaigns are driving revenue (advertiser)
Machine learning	Who "are" my users

The tech industry collects more data than it uses

Use case	Data used (by whom)	Data collected
Browser telemetry	Which websites trigger bugs, distribute malware, etc. (browser vendor)	Websites visited by users
Web analytics	Which features of a website do users (dis)like the most (web developer)	What users are doing on your website
Connectivity	Connectivity issues between client and server (network operator)	Which servers are clients connecting to
Ad tech	Which ad campaigns are driving revenue (advertiser)	Cross-site activity (saw an ad on one site and made a purchase on another)
Machine learning	Who "are" my users	attributes and labels

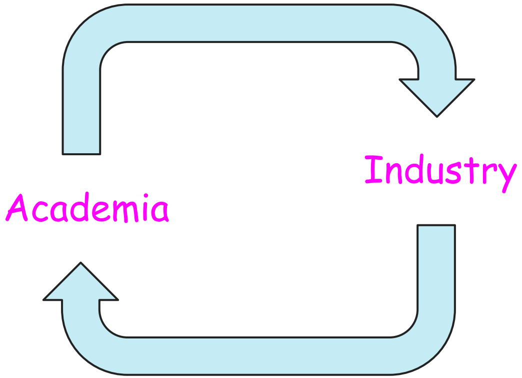
Data minimization

Collect what you use and nothing more.

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

The PPM working group at IETF

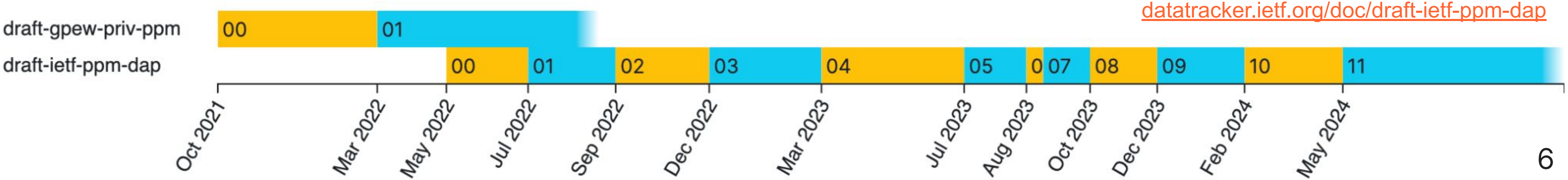
- IETF: "Internet Engineering Task Force"
  - Specifies many of the protocols that undergird the Internet (DNS, TLS, HTTP, …)
- PPM: "Privacy Preserving Measurement" working group
  - Lower the cost of data minimization
    - turn fancy crypto (MPC) into boring crypto
    - compute, bandwidth, dollars spent
  - Drive innovation by providing a deployment path for new research



The PPM working group at IETF

- **2017**: Corrigan-Gibbs and Boneh propose Prio
- **2018**: Mozilla experiments with Prio for origin telemetry
- **2020**: Google, Apple, and ISRG deploy Prio for COVID-19 exposure notification apps
- **2021**: Working group formed
- **2022**: Working group adopts its first draft
- **2023**: First deployments of DAP/Prio3 (candidate standard for Prio)
- **Goal for 2024**: Finish the base drafts

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



- **VDAF: building a box around MPC**
- Building VDAFs
- Beyond VDAFs
- MPC hot takes 🌶️

Computing on secret shared data

measurement
m_1
m_2
…
m_i
…
m_N

→

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

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	$[m_N]_1 = m_N - r_N$	$[m_N]_2 = r_N$

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

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

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	$[m_N]_1 = m_N - r_N$	$[m_N]_2 = r_N$
	$[a]_1 = [m_1]_1 + \ldots + [m_N]_1$	

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

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	$[m_N]_1 = m_N - r_N$	$[m_N]_2 = r_N$
	$[a]_1 = [m_1]_1 + \ldots + [m_N]_1$	$[a]_2 = [m_1]_2 + \ldots + [m_N]_2$

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

Computing on secret shared data

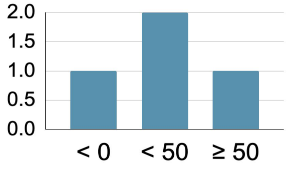
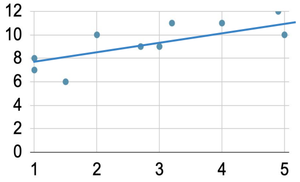
measurement	first share	second share
m_1	$[m_1]_1 = m_1 - r_1$	$[m_1]_2 = r_1$
m_2	$[m_2]_1 = m_2 - r_1$	$[m_2]_2 = r_2$
…	…	…
m_i	$[m_i]_1 = m_i - r_i$	$[m_i]_2 = r_i$
…	…	…
m_N	$[m_N]_1 = m_N - r_N$	$[m_N]_2 = r_N$
	$[a]_1 = [m_1]_1 + \ldots + [m_N]_1$	$[a]_2 = [m_1]_2 + \ldots + [m_N]_2$

The collector sums up aggregate shares to get ***aggregate result***

→ $[a]_1 + [a]_2 = m_1 + \ldots + m_N$

Computing on secret shared data

- Prio: represent ***aggregation function*** as a linear function of (some encoding of) the measurements
- 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)	

CLOUDFLARE

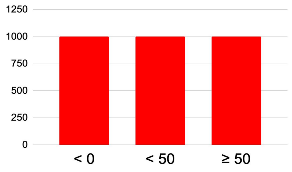
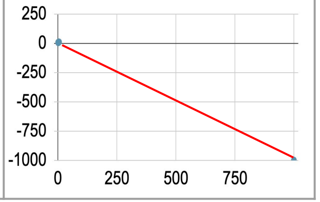
Need for interactivity: input validation

Secret sharing of 1:

- 7721925095626756828
- 10724818973787827494

Secret sharing of 10865039765974559458:

- 6499945567220489507
- 4365094198754069951

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)	

Need for interactivity: non-linear computation

- E.g., ***heavy hitters:*** Among the measurements uploaded by clients, find the subset that were uploaded at least t times (for some threshold t)

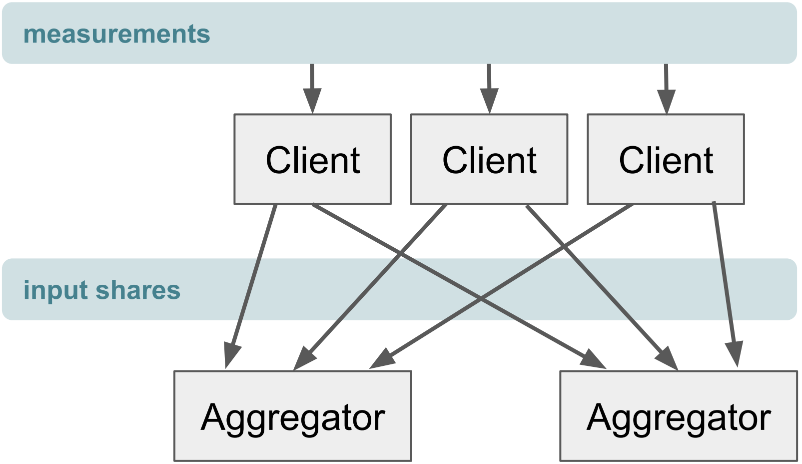
websites visited
tiktok.com
facebook.com
tiktok.com
facebook.com
myspace.com
tiktok.com
facebook.com
twitter.com

popular websites (t=3)
tiktok.com
facebook.com

CLOUDFLARE

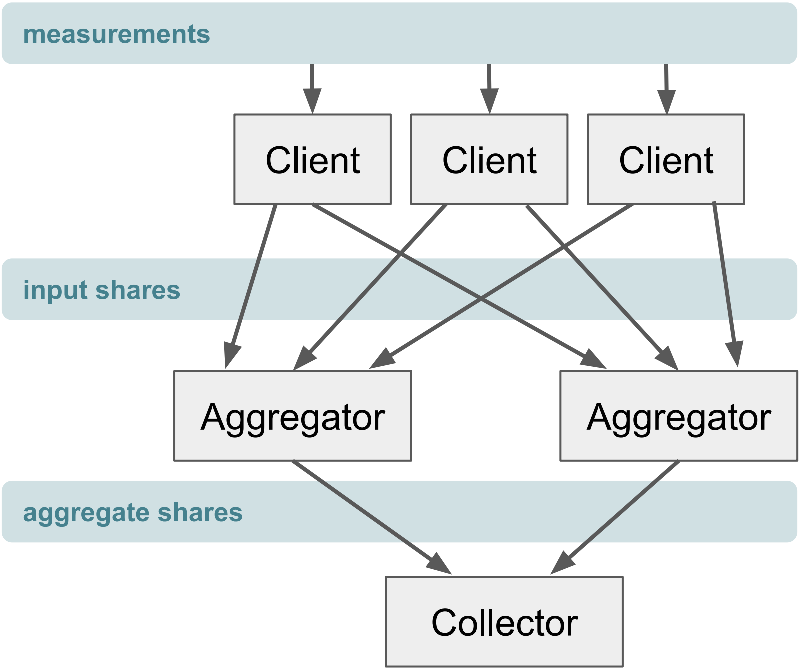
Data plane

- Each client shards its measurement into input shares and sends one share to each aggregator



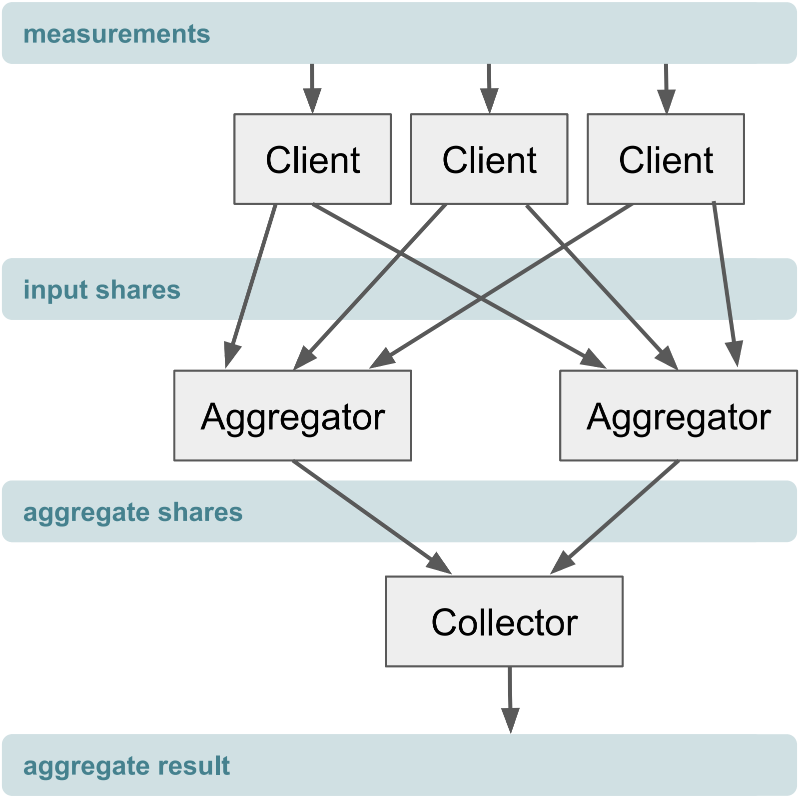
Data plane

- Each client shards its measurement into input shares and sends one share to each aggregator
- Aggregators compute aggregate shares, then send their share to the collector



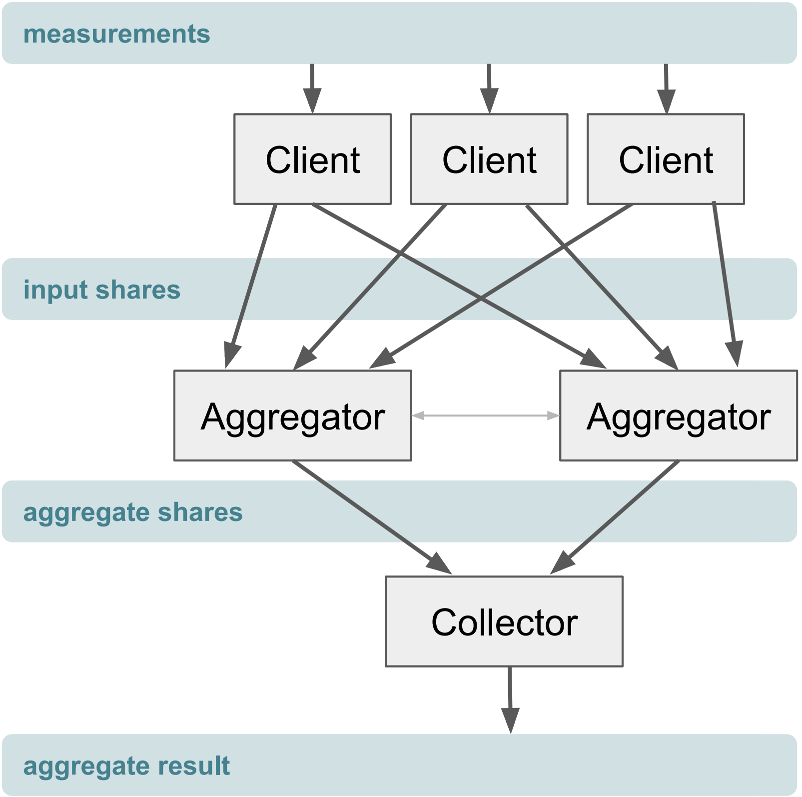
Data plane

- Each client shards its measurement into input shares and sends one share to each aggregator
- Aggregators compute aggregate shares, then send their share to the collector
- Collector unshards the aggregate result



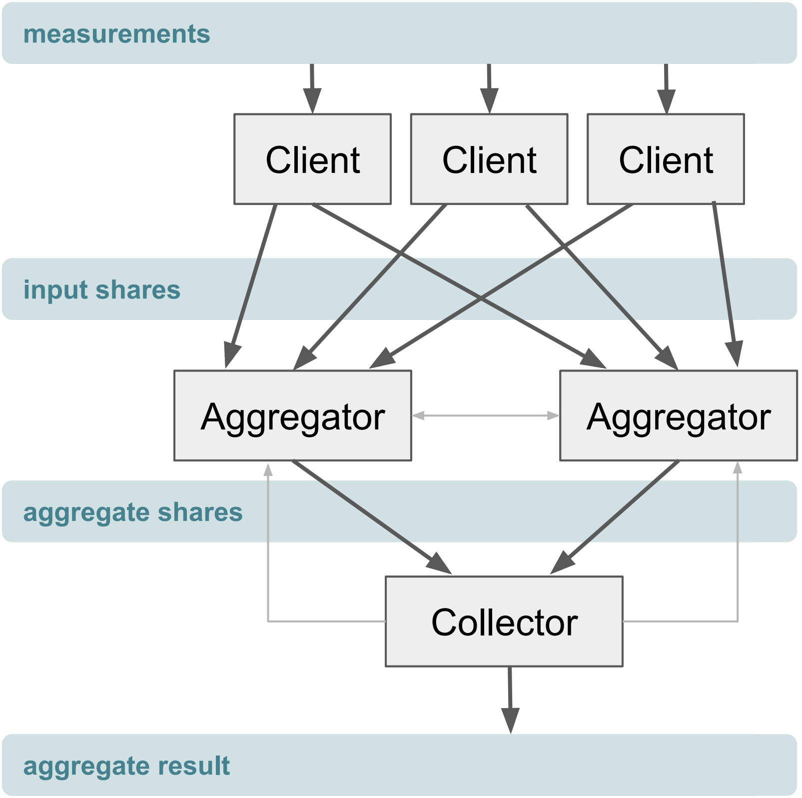
Control plane

- Aggregators interact during aggregation (input validation)



Control plane

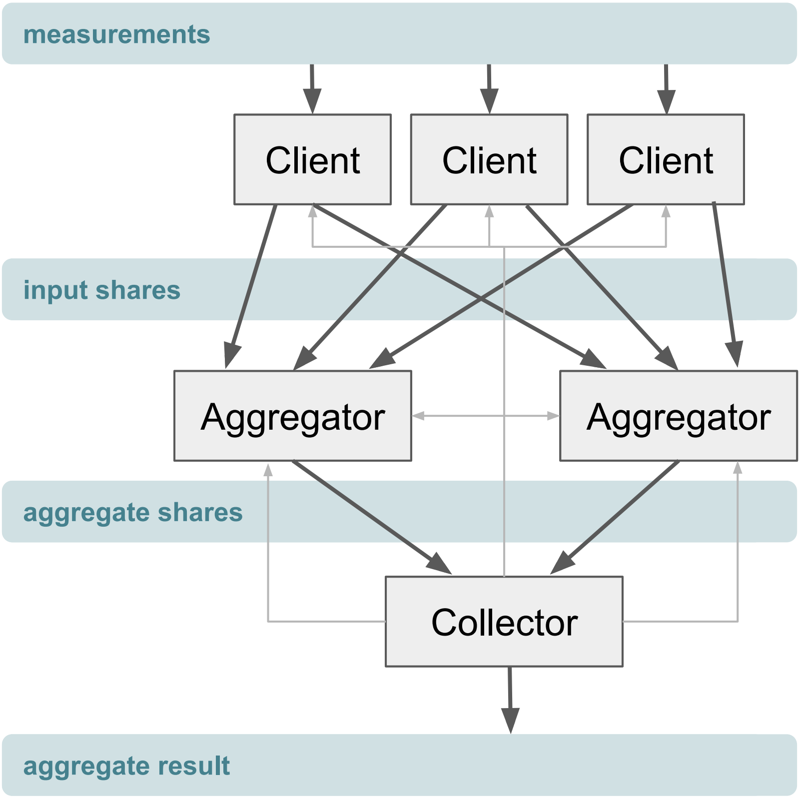
- Aggregators interact during aggregation (input validation)
- Collector might push information to aggregators (heavy hitters with Poplar)



CLOUDFLARE

Control plane

- Aggregators interact during aggregation (input validation)
- Collector might push information to aggregators (heavy hitters with Poplar)
- Collector might push information to clients (federated learning with PINE)



CLOUDFLARE

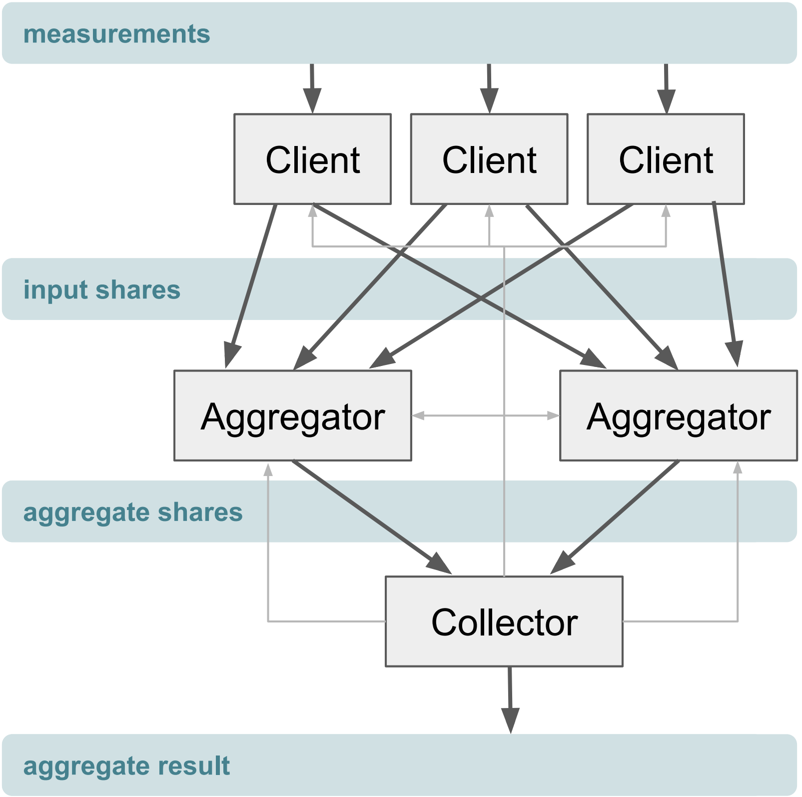
Verifiable Distributed Aggregation Function (VDAF)

- Used to compute functions of the form

$$f(\sigma, m_1, \ldots, m_N) = g(\sigma, m_1) + \ldots + g(\sigma, m_N)$$

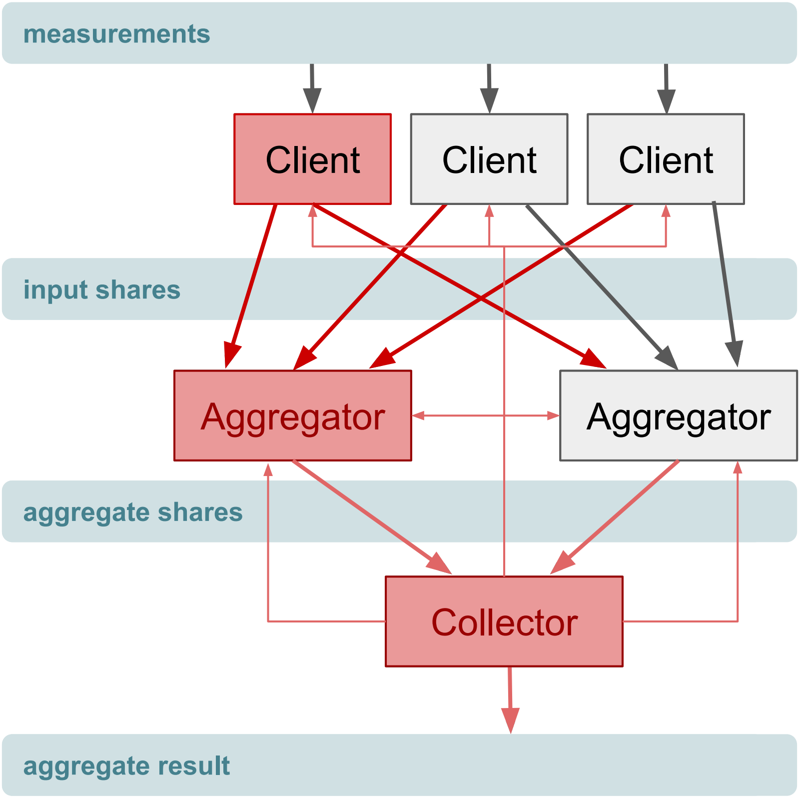
$m_1, \ldots, m_N \in$ **Measurements** (chosen by clients)

$\sigma \in$ **Aggregation Parameters** (chosen by collector)



Privacy

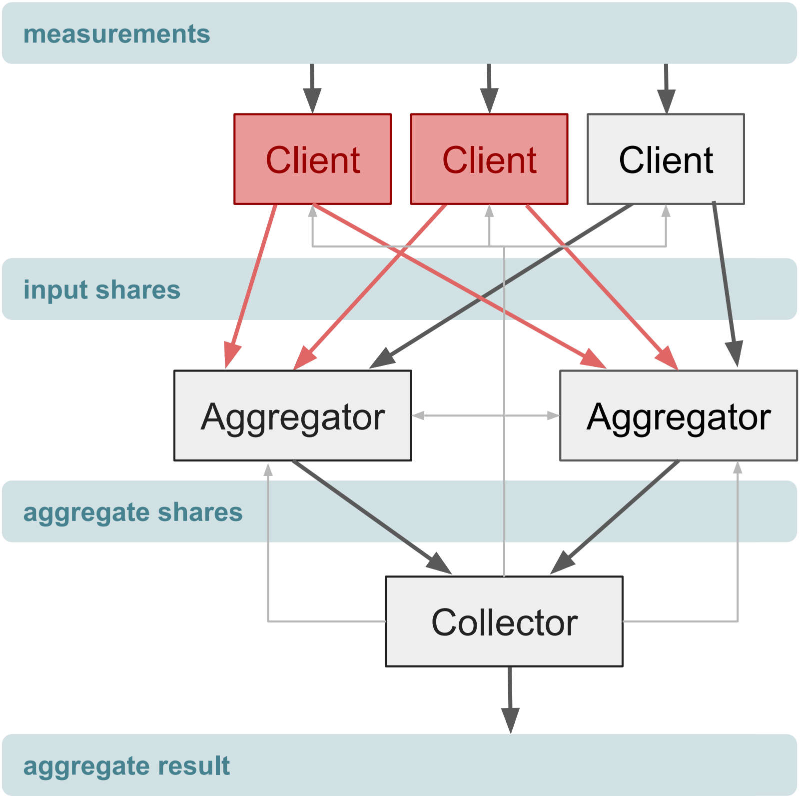
- Threat model: one aggregator is honest
- Security goal: data minimization
    - Attacker's view of the protocol execution is efficiently simulatable given the aggregate result*



*There may be additional leakage, depending on the VDAF.

Robustness

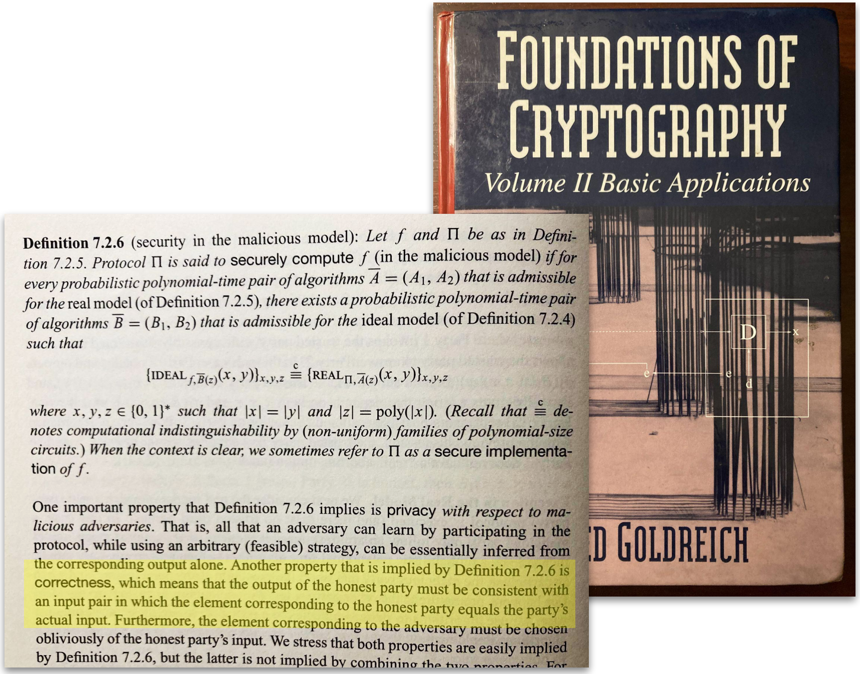
- Threat model: aggregators are honest
- Security goal: collector correctly aggregates honest clients' measurements
    - Aggregate result is efficiently extractable from the attacker's execution (i.e., its random oracle queries)



CLOUDFLARE

Malicious versus semi-honest security

- We ***must have*** privacy against malicious aggregators
- We ***don't always need*** robustness against malicious aggregators
    - Malicious robustness is nice to have, but not any cost (more parties, more rounds, more bandwidth, etc.)



Definition 7.2.6 (security in the malicious model): *Let f and Π be as in Definition 7.2.5. Protocol Π is said to* securely compute f (in the malicious model) *if for every probabilistic polynomial-time pair of algorithms $\overline{A} = (A_1, A_2)$ that is admissible for the* real model (of Definition 7.2.5), *there exists a probabilistic polynomial-time pair of algorithms $\overline{B} = (B_1, B_2)$ that is admissible for the* ideal model (of Definition 7.2.4) *such that*

$$\{\text{IDEAL}_{f,\overline{B}(z)}(x, y)\}_{x,y,z} \stackrel{c}{\equiv} \{\text{REAL}_{\Pi,\overline{A}(z)}(x, y)\}_{x,y,z}$$

where $x, y, z \in \{0,1\}^*$ such that $|x| = |y|$ and $|z| = \text{poly}(|x|)$. (Recall that $\stackrel{c}{\equiv}$ denotes computational indistinguishability by (non-uniform) families of polynomial-size 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 (feasible) 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 chosen obliviously of the honest party's input. We stress that both properties are easily implied by Definition 7.2.6, but the latter is not implied by combining the two properties. For

- VDAF: building a box around MPC
- **Building VDAFs**
- Beyond VDAFs
- MPC hot takes 🌶️

Fully linear proofs [BBCG+19]

Syntax:

Π := Prove(X) // proof generation
V := Query(X, Π; qr) // query generation
d := Decide(V) // decision

Full linearity: Query(X, Π; qr) is equivalent to:

- Split X, Π into shares $[X]_i$, $[\Pi]_i$ for all i
- $[V]_i$:= Query($[X]_i$, $[\Pi]_i$; qr) for all i
- Return $[V]_1 + \ldots + [V]_s$

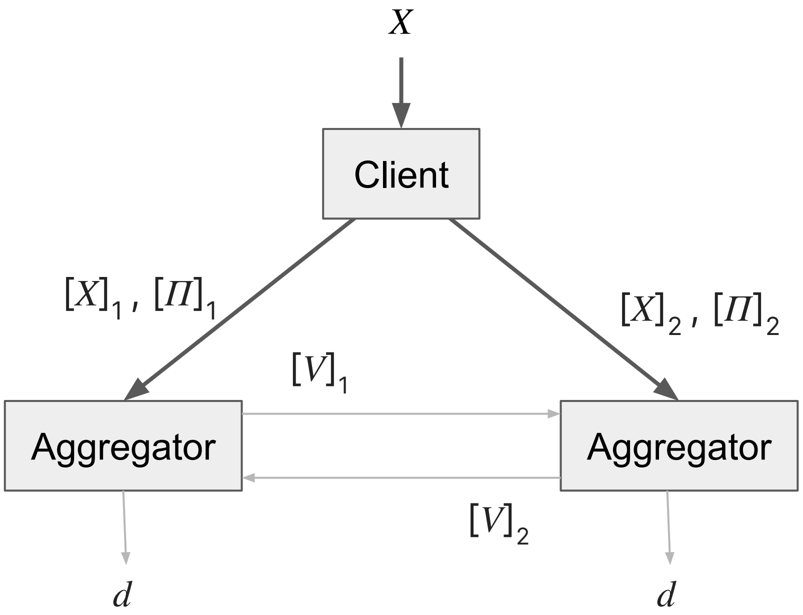
[BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs." CRYPTO 2019.

Syntax:

Π := Prove(X) // proof generation
V := Query(X, Π; qr) // query generation
d := Decide(V) // decision

Full linearity: Query(X, Π; qr) is equivalent to:

- Split X, Π into shares $[X]_i$, $[\Pi]_i$ for all i
- $[V]_i$:= Query($[X]_i$, $[\Pi]_i$; qr) for all i
- Return $[V]_1 + \ldots + [V]_s$



[draft-irtf-cfrg-vdaf] Barnes et al. "Verifiable Distributed Aggregation Functions." IRTF draft 09.

Application of FLPs: federated learning with PINE [ROCT23]

[draft-chen-cfrg-vdaf-pine]

```
   data
     |
     v
+---------+          gradients                 +--------+
| Clients |-+   ---------------------------->  | Server |
+---------+ |-+                                +--------+
  +---------+ |                                    |
    +---------+                                    |
     ^                                             |
     |              updated model                  |
     +---------------------------------------------+

            Figure 1: Federated learning
```

```
   data
     |
     v        gradient                  aggregate
+---------+   shares    +-------------+  shares    +-----------+
| Clients |-+   ---->   | Aggregators |-+  ----->  | Collector |
+---------+ |-+         +-------------+ |          +-----------+
  +---------+ |           +-------------+                |
    +---------+                                          |
     ^                                                   |
     |               updated model                       |
     +---------------------------------------------------+

        Figure 2: Federated learning with a VDAF
```

- PINE: A VDAF for federated learning
  - Aggregating real-valued vectors (***gradients***) with bounded ***L2-norm***
  - FLP for L2 norm computation; new techniques for checking correctness of computation
  - More practical than Prio for larger models

L2 norm: $||\mathbf{x}||_2 = ((x_1)^2 + \ldots + (x_d)^2)^{1/2}$

[ROCT23] Rothblum et al. "PINE: Efficient Norm-Bound Verification for Secret-Shared Vectors." USENIX 2024.

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)≠0

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

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)≠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.

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)≠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 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.

CLOUDFLARE

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)≠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
```

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

```
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
```

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

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)≠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 p(0)

# 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 * p(i) for i in range(len(x)))
    sum_chk = p(len(x))
    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
```

Observation:
Polynomial evaluation is linear!

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

CLOUDFLARE

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)≠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 qr)

```
def counter(x: F) -> F:
    return p(0)

# 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.

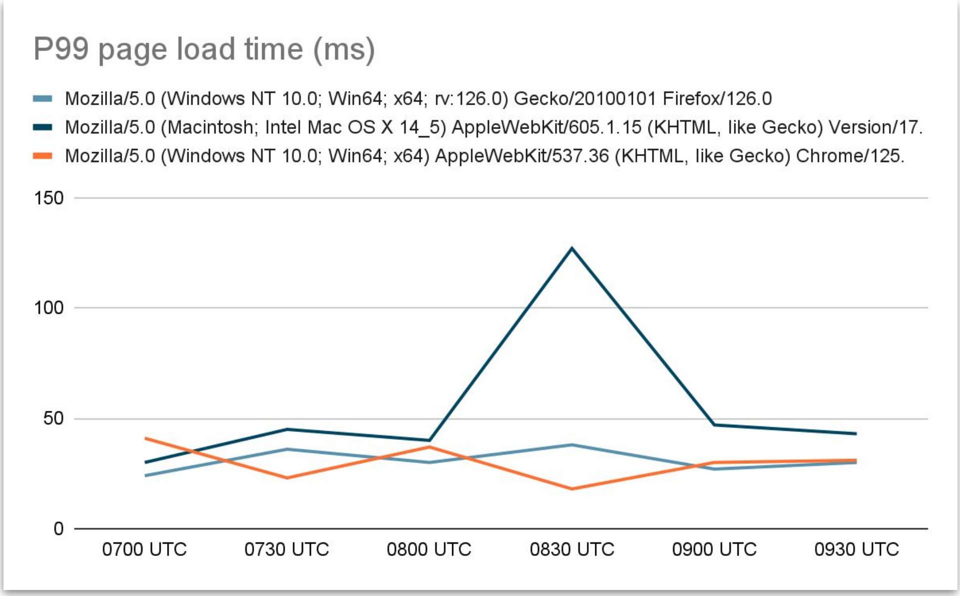
Distributed point functions [GI14]

- Point function: $f(\alpha)=\beta$ and $f(X)=0$ for all $X\neq\alpha$
  - DPF: secret sharing of a point function
    - (P, K_1, K_2) := Gen(α,β)
    - $[f(X)]_i$ = Eval(P, K_i, X) for all X, i

index	value
0	0
1	0
2	0
…	…
α	β
…	…

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

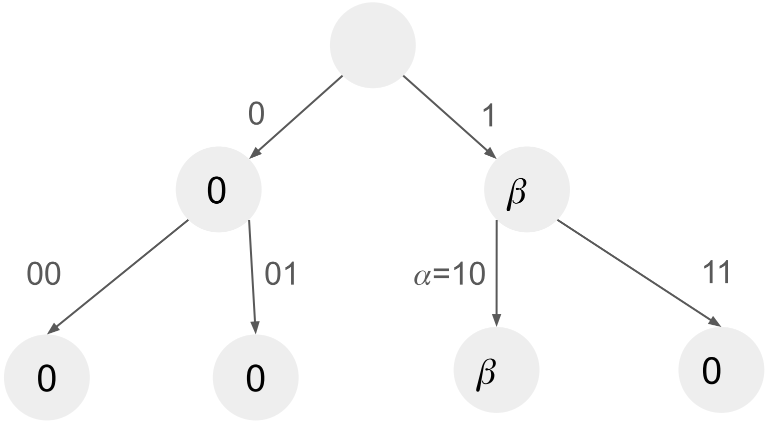
- Prio-style metrics, grouped by ***attributes*** (user agent, software version, geolocation, etc.) without reducing anonymity set



[MPD+24] Mouris et al. "Mastic: Private Weighted Heavy-Hitters and Attribute-Based Metrics." ***In submission.***.

Incremental distributed point functions [BBCG+21]

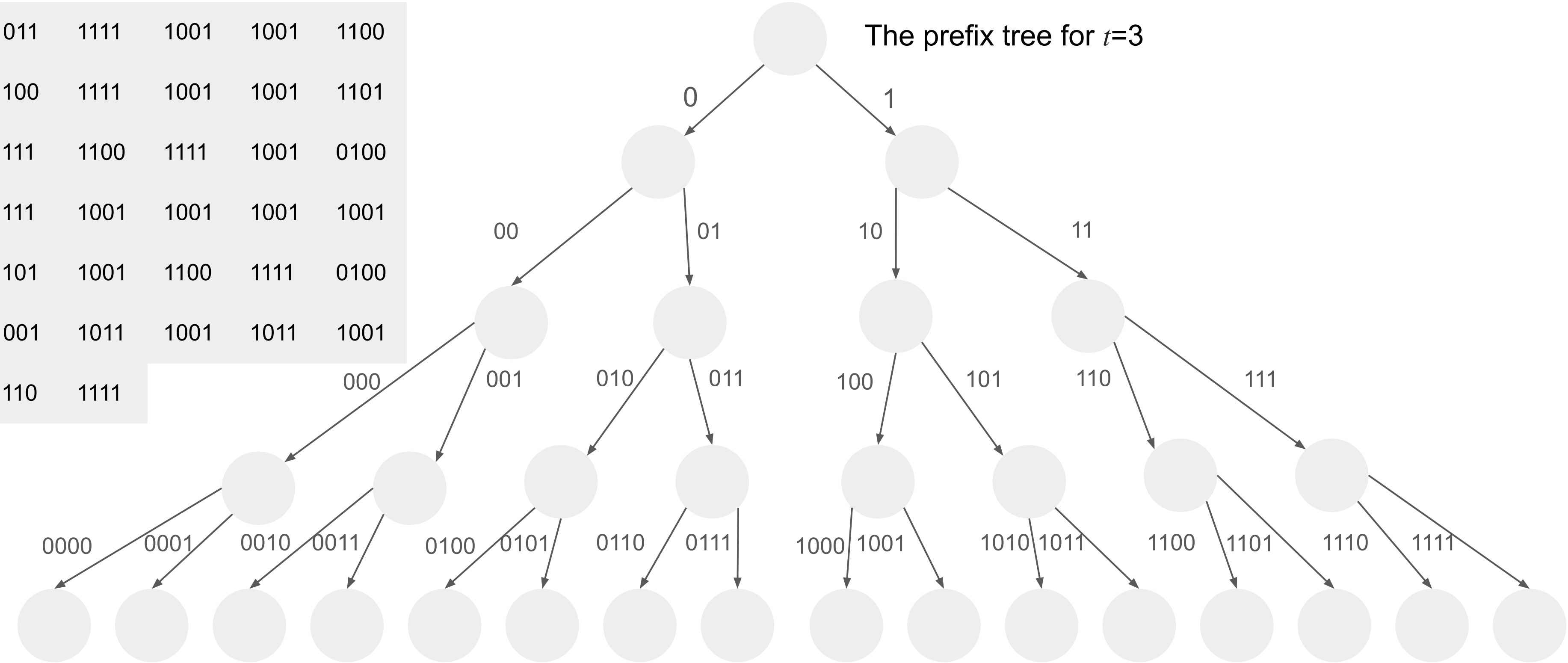
- Incremental point function: $f(X)=\beta$ for any ***prefix*** X of $\alpha\in\{0,1\}^n$ and $f(X)=0$ otherwise
  - IDPF: secret-sharing of an incremental point function
    - (P, K_1, K_2) := Gen(α,β)
    - $[f(X)]_i$ = Eval(P, K_i, X) for all X, i



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

Application of IDPFs: Computing t-heavy-hitters with Poplar1 [draft-irtf-cfrg-vdaf]

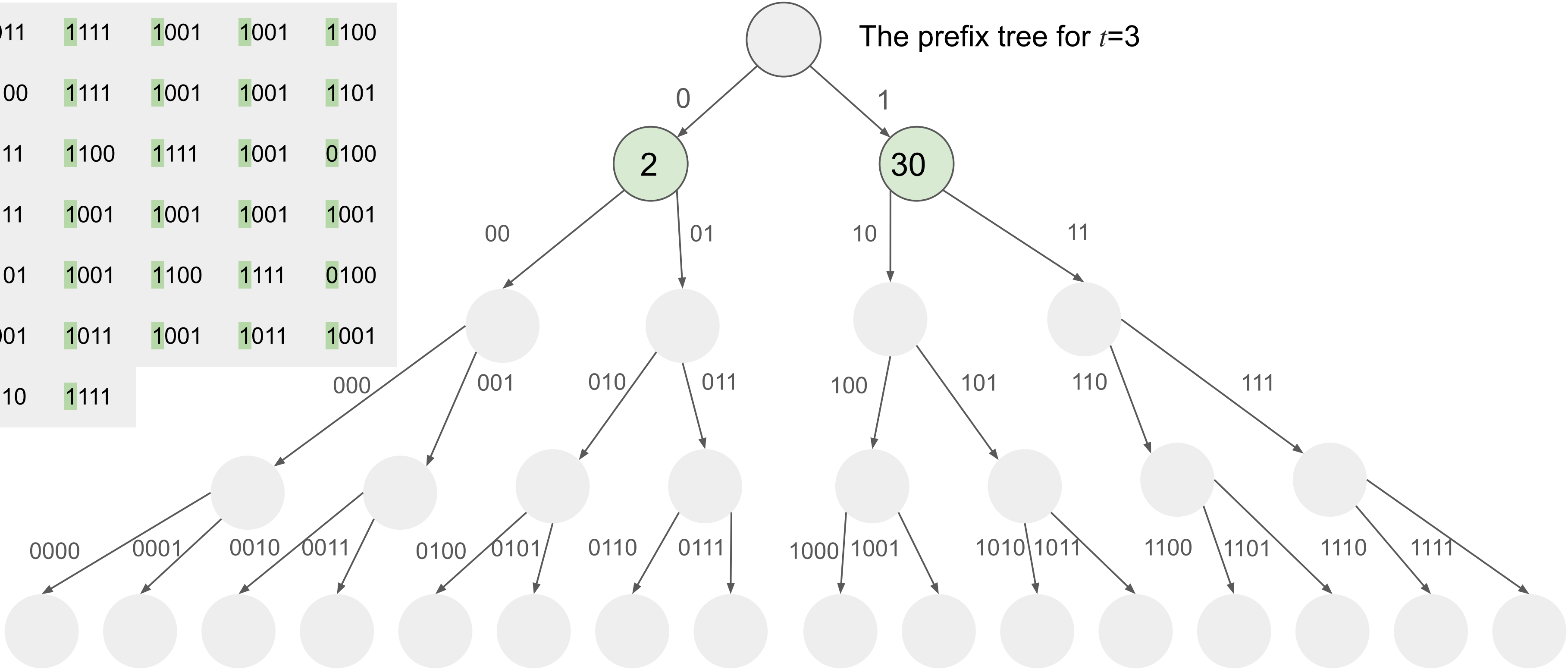
1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



[draft-irtf-cfrg-vdaf] Barnes et al. "Verifiable Distributed Aggregation Functions." IRTF draft 09.

Application of IDPFs: Computing t-heavy-hitters with Poplar1 [draft-irtf-cfrg-vdaf]

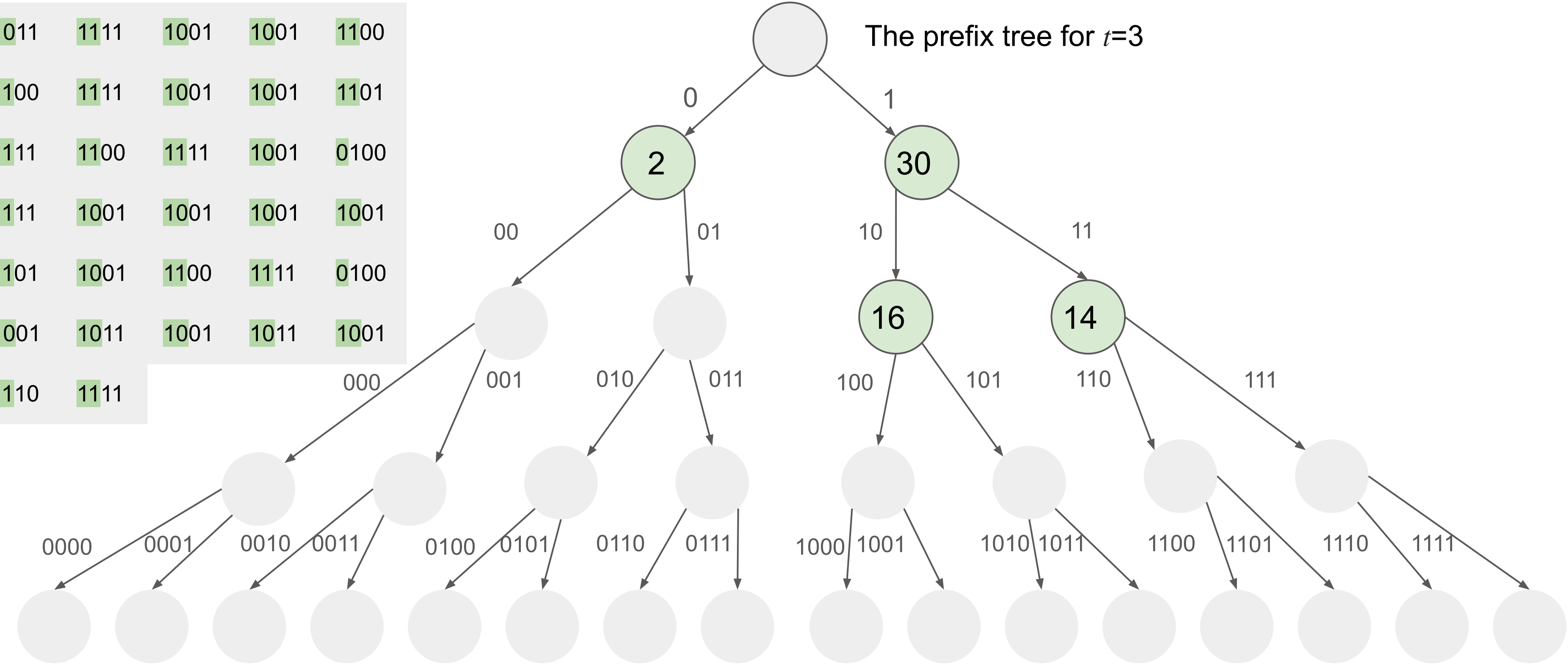
1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



[draft-irtf-cfrg-vdaf] Barnes et al. "Verifiable Distributed Aggregation Functions." IRTF draft 09.

Application of IDPFs: Computing t-heavy-hitters with Poplar1 [draft-irtf-cfrg-vdaf]

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			

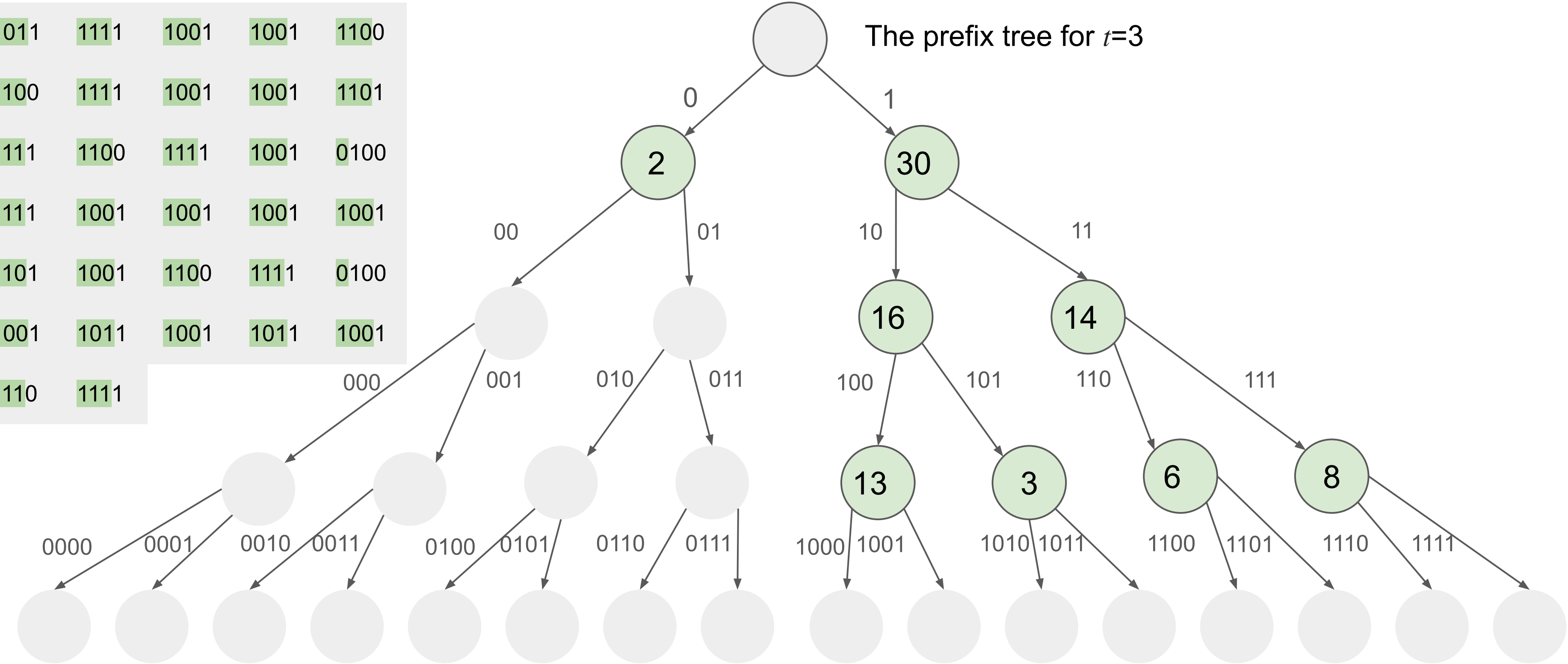


[draft-irtf-cfrg-vdaf] Barnes et al. "Verifiable Distributed Aggregation Functions." IRTF draft 09.

Application of IDPFs: Computing t-heavy-hitters with Poplar1 [draft-irtf-cfrg-vdaf]

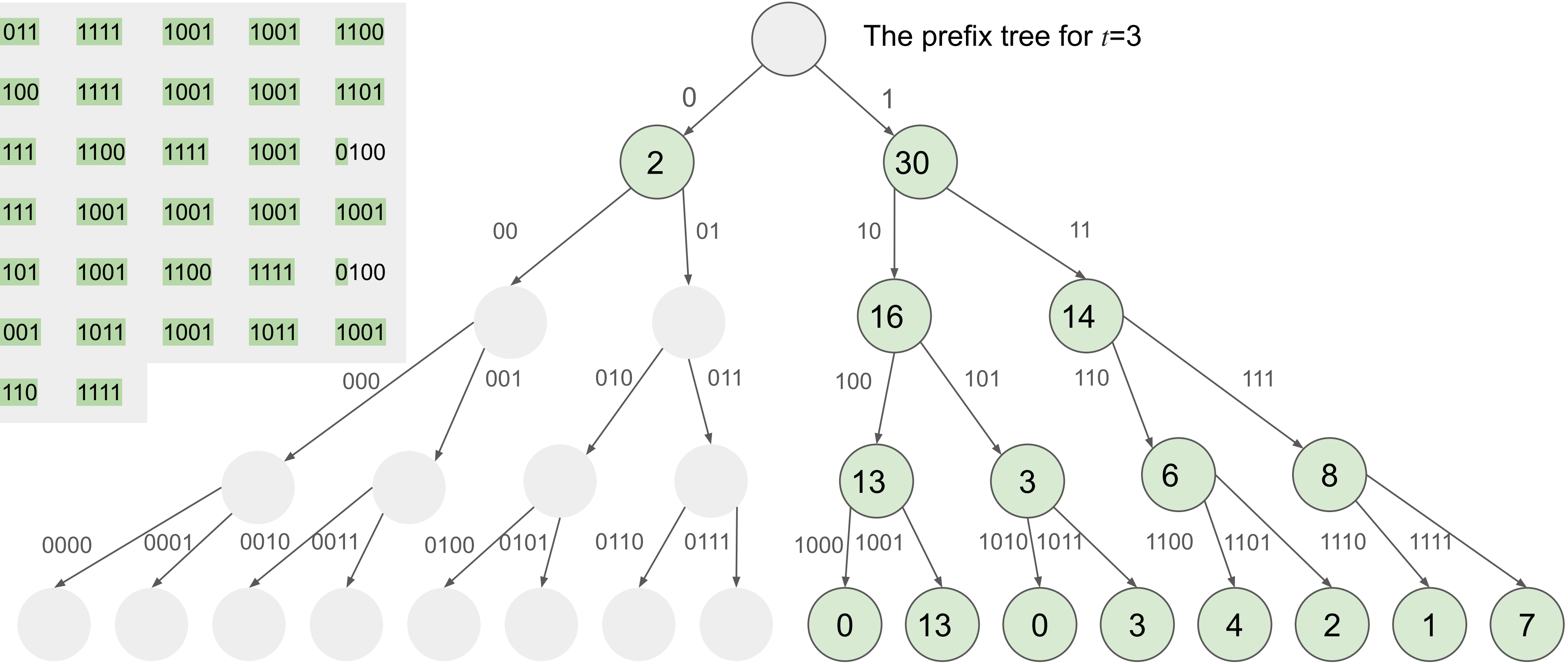
CLOUDFLARE

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



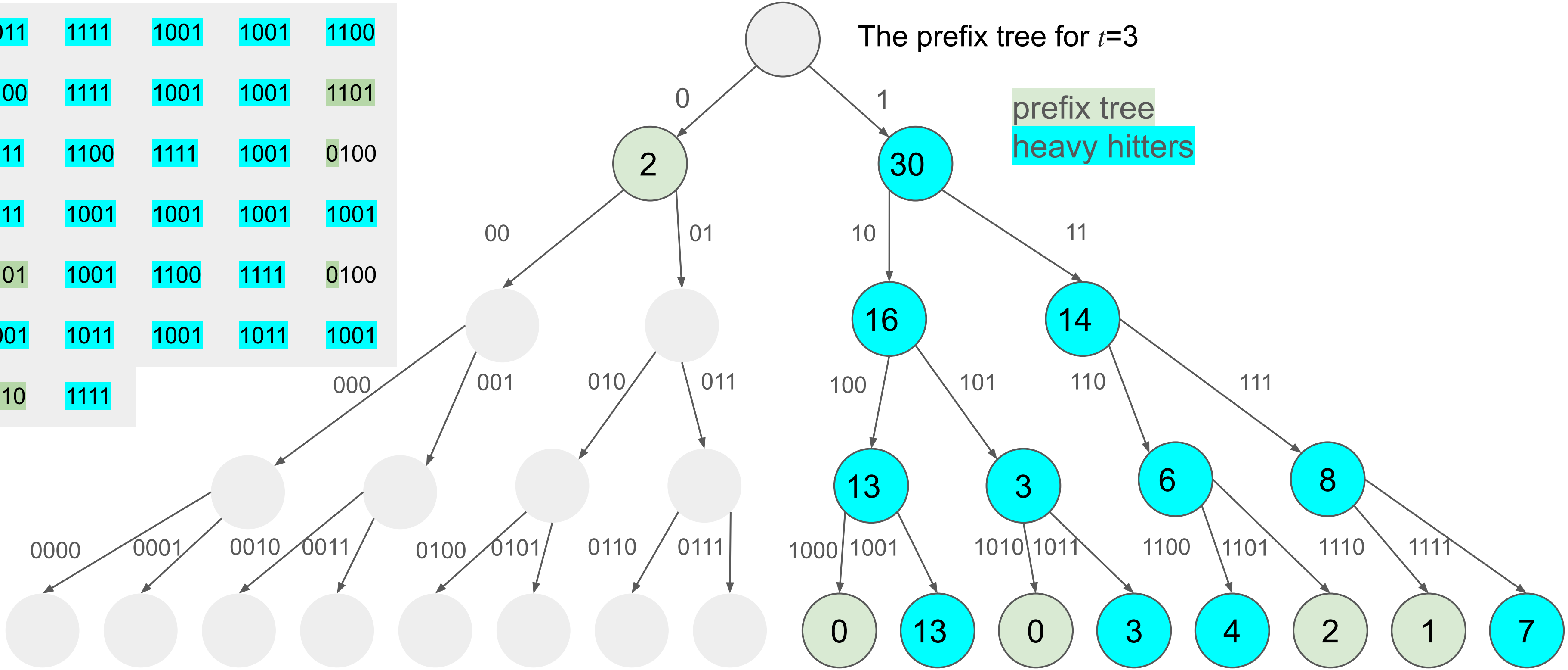
[draft-irtf-cfrg-vdaf] Barnes et al. "Verifiable Distributed Aggregation Functions." IRTF draft 09.

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



Application of IDPFs: Computing t-heavy-hitters with Poplar1 [draft-irtf-cfrg-vdaf]

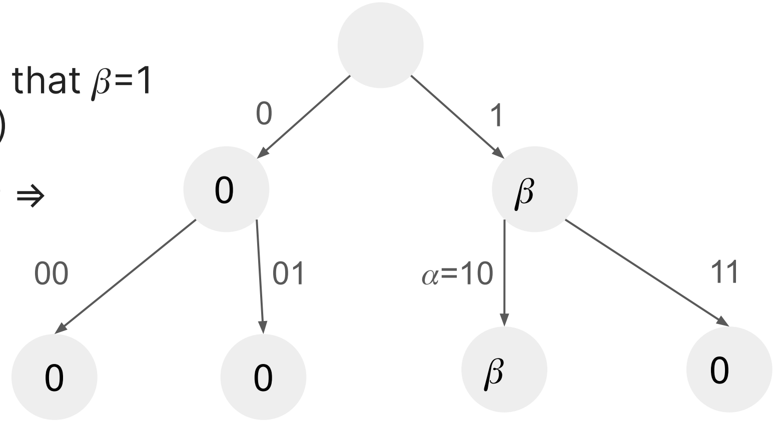
1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



[draft-irtf-cfrg-vdaf] Barnes et al. "Verifiable Distributed Aggregation Functions." IRTF draft 09.

Verifiable IDPF [MST24]

- IDPF with verifiability of ***one-hotness***
  - (P, K_1, K_2) := Gen(α,β)
  - $([f(X_1), \ldots, f(X_p)]_i, \pi_i)$ = Eval$(P, K_i, \mathbf{X})$ for all $\mathbf{X}=(X_1, \ldots, X_p)$, i
    - $\pi_1 = \pi_2$ implies $f(X_1), \ldots, f(X_p)$ is a one-hot vector
- Also need to verify that the non-zero value is ***in-range***
  - PLASMA [MST24] solves this for the special case that β=1 (same as Poplar, but with lower round complexity)
  - Mastic [MPD+24] solves the general case via FLP ⇒ *weighted* heavy hitters, *attribute-based* metrics



[MST24] Mouris et al. "PLASMA: Private, Lightweight Aggregated Statistics against Malicious Adversaries." PETS 2024.
[MPD+24] Mouris et al. "Mastic: Private Weighted Heavy-Hitters and Attribute-Based Metrics." ***In submission.***

Boolean-to-arithmetic conversion [ABJ+22]

- Use case: aggregating vectors of counters
- Clients send XOR shares of each counter; aggregators convert to shares in a field suitable for aggregation
  - Much more efficient for clients
  - ***Open question:*** 2-party conversion that is private in the presence of a malicious aggregator. (Easy in the 3-party, honest-majority setting.)

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

Silently verifiable proofs [RZCGP24]

- Extends FLP such that proof verification can be batched across multiple reports
  - Much lower aggregator⇔aggregator bandwidth cost
  - ***Open question:*** Denial-of-Service (DoS) risk costs increases as the fraction of invalid reports increases
    - Fine for many deployments, but too risky for others
  - Also possible for VIDPF [MST24]

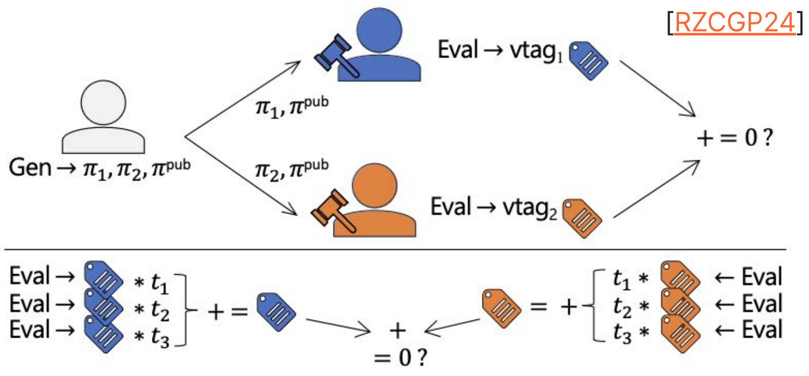


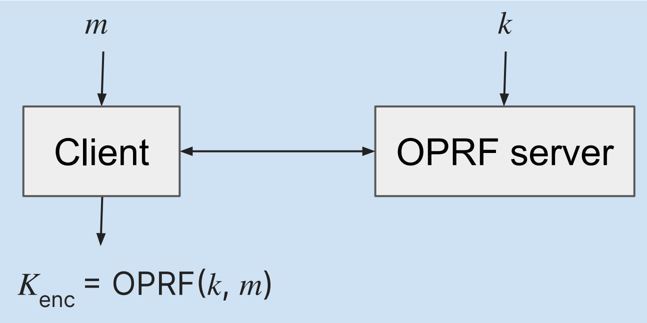
Figure 3: Silently verifiable proofs with batch verification.

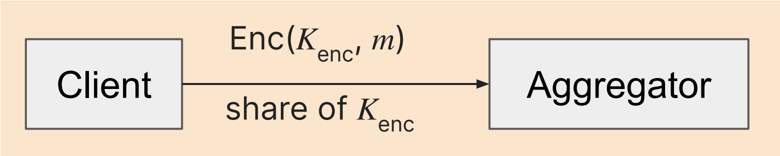
[RZCGP24] Rathee et al. "Private Analytics via Streaming, Sketching, and Silently Verifiable Proofs." IEEE S&P 2024.
[MST24] Mouris et al. "PLASMA: Private, Lightweight Aggregated Statistics against Malicious Adversaries." PETS 2024.

- VDAF: building a box around MPC
- Security goals for VDAFs
- Building VDAFs
- **Beyond VDAFs**
- MPC hot takes 🌶️

Heavy hitters via threshold secret sharing [DSG+22]

- Basic idea:
  - Each client generates a t-of-n secret share of a key to encrypt its measurement
  - After receiving t shares, aggregator can recover the key and decrypt
- Based on an Oblivious PRF (OPRF) [RFC 9497]
- With more recent techniques [LNT24], achieves the same level of privacy as Poplar (in a slightly different threat model)
- Achieving robustness is expensive





[DSG+22] Davidson et al. "STAR: Secret Sharing for Private Threshold Aggregation Reporting." CCS 2022.
[RFC 9497] Davidson et al. "Oblivious Pseudorandom Functions (OPRFs) Using Prime-Order Groups."
[LNT24] Li et al. "POPSTAR: Lightweight Threshold Reporting with Reduced Leakage." ***In submission.***

CLOUDFLARE

Sparse histograms

- Clients hold pairs (α, β): for each index α held by at least one client, compute the sum of the values β
- Protocol of [BGR+24]
  - Based on OPRF and multiplicative homomorphic encryption
  - Differentially privacy baked in by default ⇒ much better utility than Poplar with differential privacy

Differential privacy: aggregate should not depend "too much" on any one measurement

Credit: Paille // CC BY-SA 2.0.

[BGR+24] Braun et al. "Malicious Security for Sparse Private Histograms." ePrint 2024/469.

Joining data sources

- Last-touch attribution: count the number of purchases attributable to an ad
  - Put purchases and ad impressions in a database: for each purchase, find the most recent ad impression
  - IPA ("Interoperable Private Attribution"): Sorting via the 3-party, honest majority computation [CHI+19]

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.

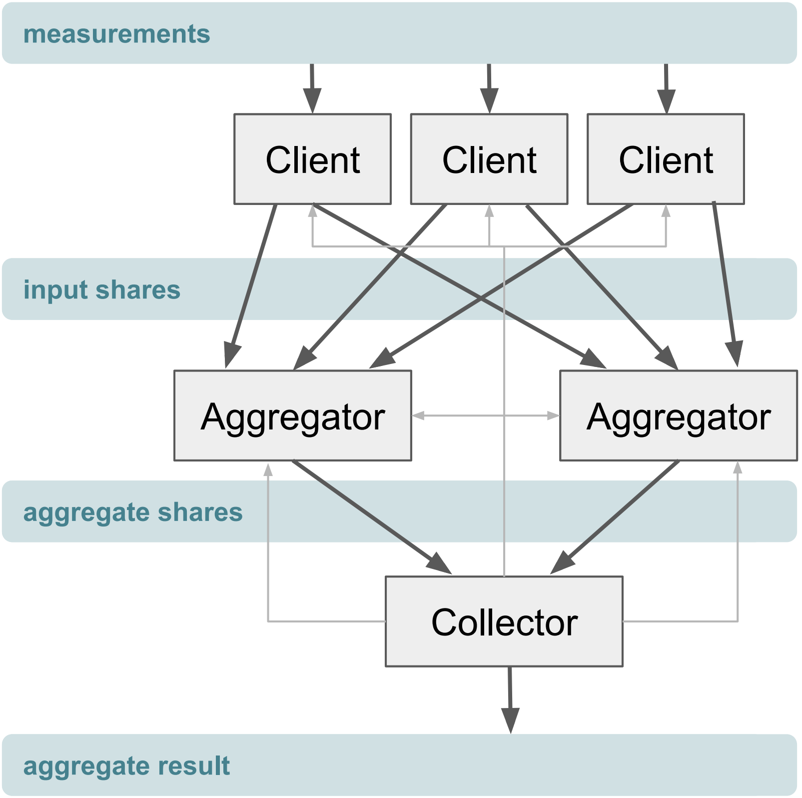
- VDAF: building a box around MPC
- Building VDAFs
- Beyond VDAFs
- **MPC hot takes 🌶️**

MPC is crypto + distributed systems

- While some requirements like memory, CPU, and bandwidth are well-documented in the literature, many other requirements are not well understood
  - Strong versus eventual consistency
  - Load balancing across machines
  - Moving workloads between machines

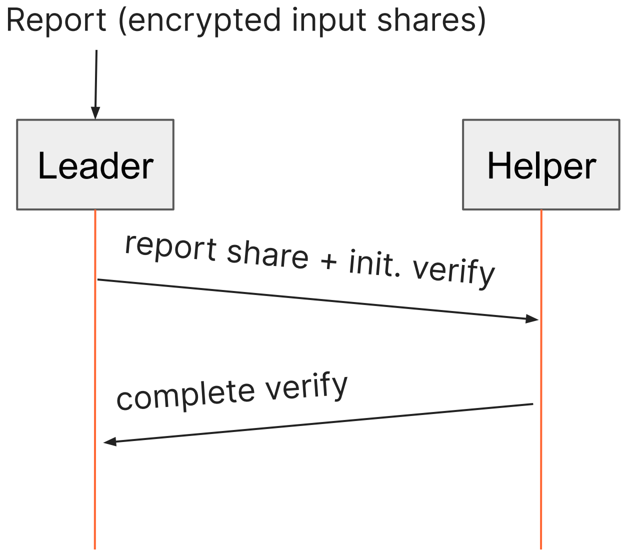
Number of parties

- 2 parties fits neatly into client-server architecture of HTTP
- 3 parties is more complex, but workable
- ≥4 parties is probably too much coordination



Number of rounds

- 1 round: Complete aggregation in a single HTTP request
- ≥2 rounds: Have to keep state across HTTP requests; less flexibility in the server architecture



Differential privacy should be baked in

- Especially important when the protocol has leakage (e.g., heavy hitters)
- Generic composition of VDAF with some DP mechanism usually has sub-optimal utility
- ***Challenge:*** Securely sample shares of discrete Gaussian or Laplace with low communication cost, in the 2-party setting [KKL+23]

THEOREM 1 (Π IS ϵ-SIM-CDP). *Let S be the simulator defined in* FIGURE 1. *Then:*

(i) *S is ϵ-DP.*

(ii) *For every Π-attacker A there exists a* FOO*-attacker B such that* $\mathbf{Adv}_{\Pi}^{\text{sim-cdp}}(A, S) \leq \mathbf{Adv}_{\text{Foo}}^{\text{bar}}(B)$.

Differential privacy: aggregate should not depend "too much" on any one measurement

Credit: Paille // CC BY-SA 2.0.

[KKL+23] Keeler et al. "DPrio: Efficient Differential Privacy with High Utility for Prio." PETS 2023.

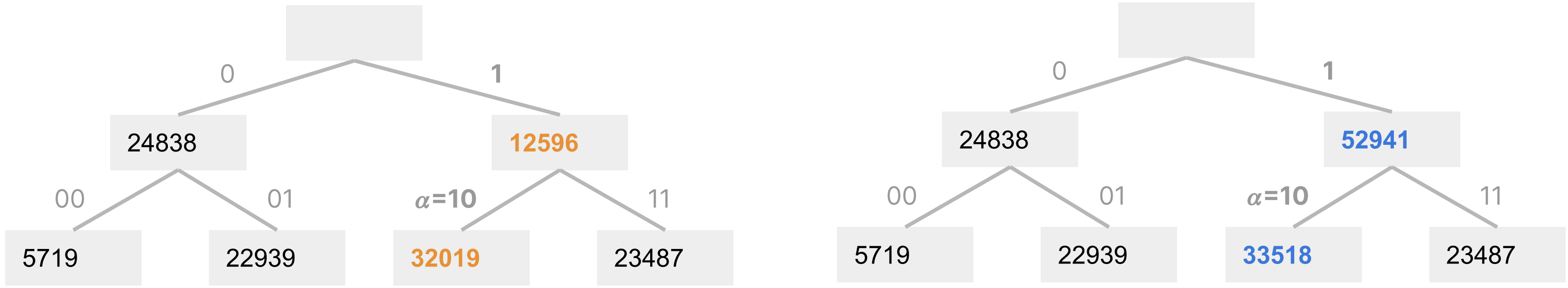
Thanks!

- Join the mailing list: https://www.ietf.org/mailman/listinfo/Ppm
- Join #ppm in the IETF slack: https://ietf.slack.com/
- Base 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 new VDAFs, differential privacy, dealing with Sybil attacks, and more!

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***



Function secret sharing

- FSS [BGI16]: split f into shares such that $[f(X)]_1, \ldots, [f(X)]_s$ can be evaluated for any X
  - Possible to construct efficient schemes for specific classes of functions
  - Transforming privacy-only FSS to verifiable FSS
    - Arithmetic sketching [BBCG+23] generalizes sketching scheme from Poplar for achieving robustness with IDPFs

[BGI16] Boyle et al. "Function Secret Sharing: Improvements and Extensions." CCS 2016.
[BBCG+23] Boneh et al. "Arithmetic Sketching." CRYPTO 2023.

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.