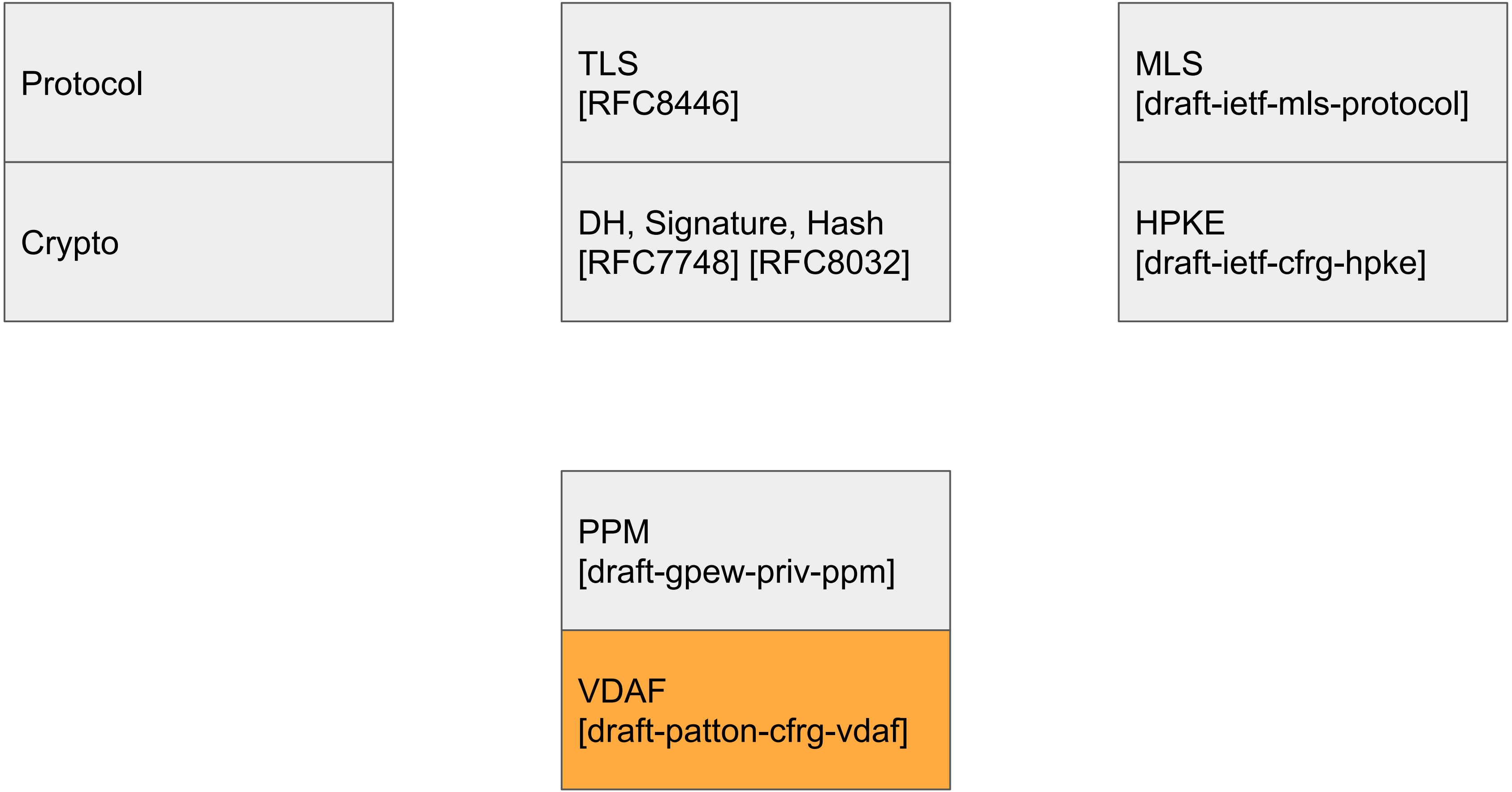
Verifiable Distributed Aggregation Functions

draft-patton-cfrg-vdaf
IETF 112
November 2021



Protocol

Crypto

TLS
[RFC8446]

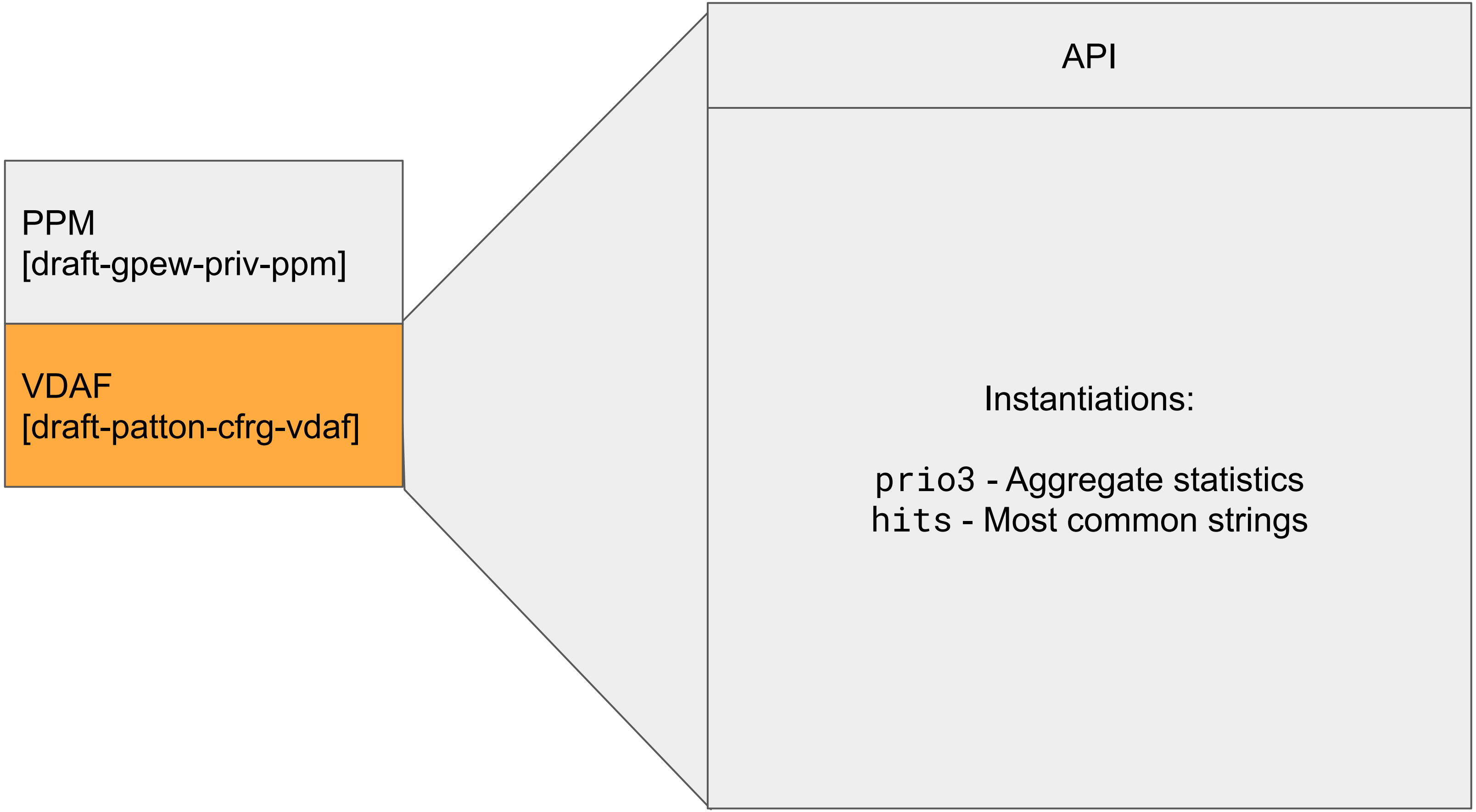
DH, Signature, Hash
[RFC7748] [RFC8032]

MLS
[draft-ietf-mls-protocol]

HPKE
[draft-ietf-cfrg-hpke]

PPM
[draft-gpew-priv-ppm]

VDAF
[draft-patton-cfrg-vdaf]



PPM
[draft-gpew-priv-ppm]

VDAF
[draft-patton-cfrg-vdaf]

API

Instantiations:

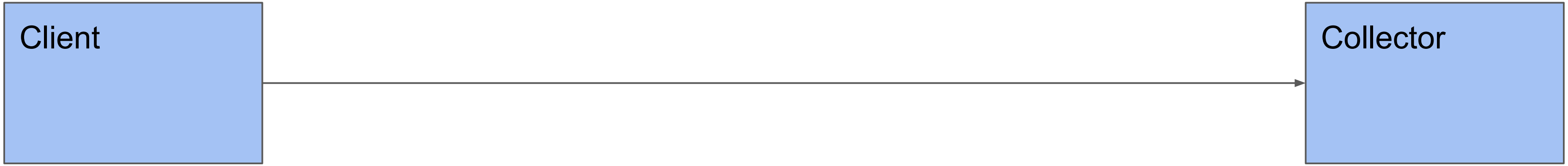
`prio3` - Aggregate statistics
`hits` - Most common strings

What does a VDAF do?

VDAggregationF - Compute a statistic over batch of measurements without revealing anything about the individual measurements

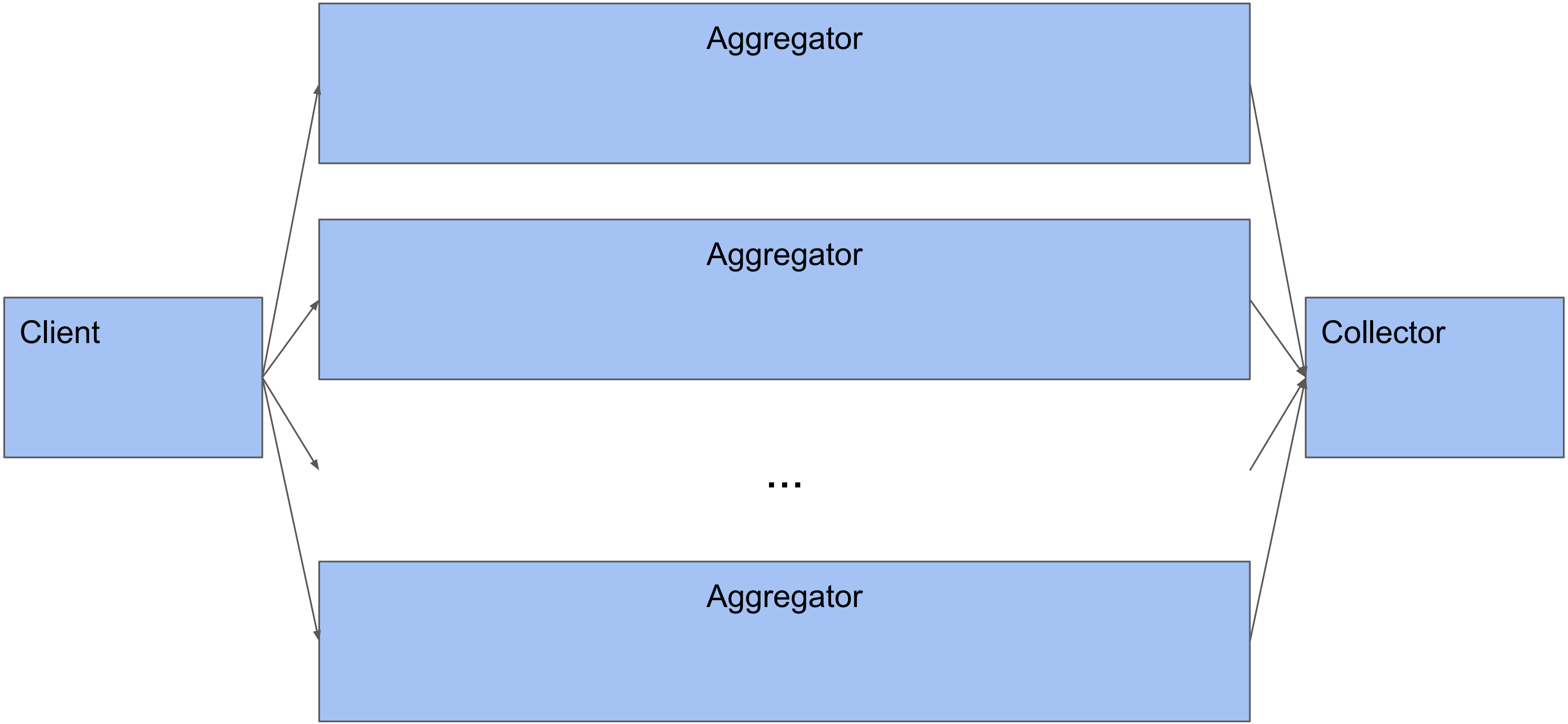
VDistributedAF - The privacy of individual measurements is assured by spreading the computation over **non-colluding** servers (“aggregators”)

VerifiableDAF - The aggregators can check the correctness of client’s inputs to prevent malicious or misconfigured clients from corrupting aggregates

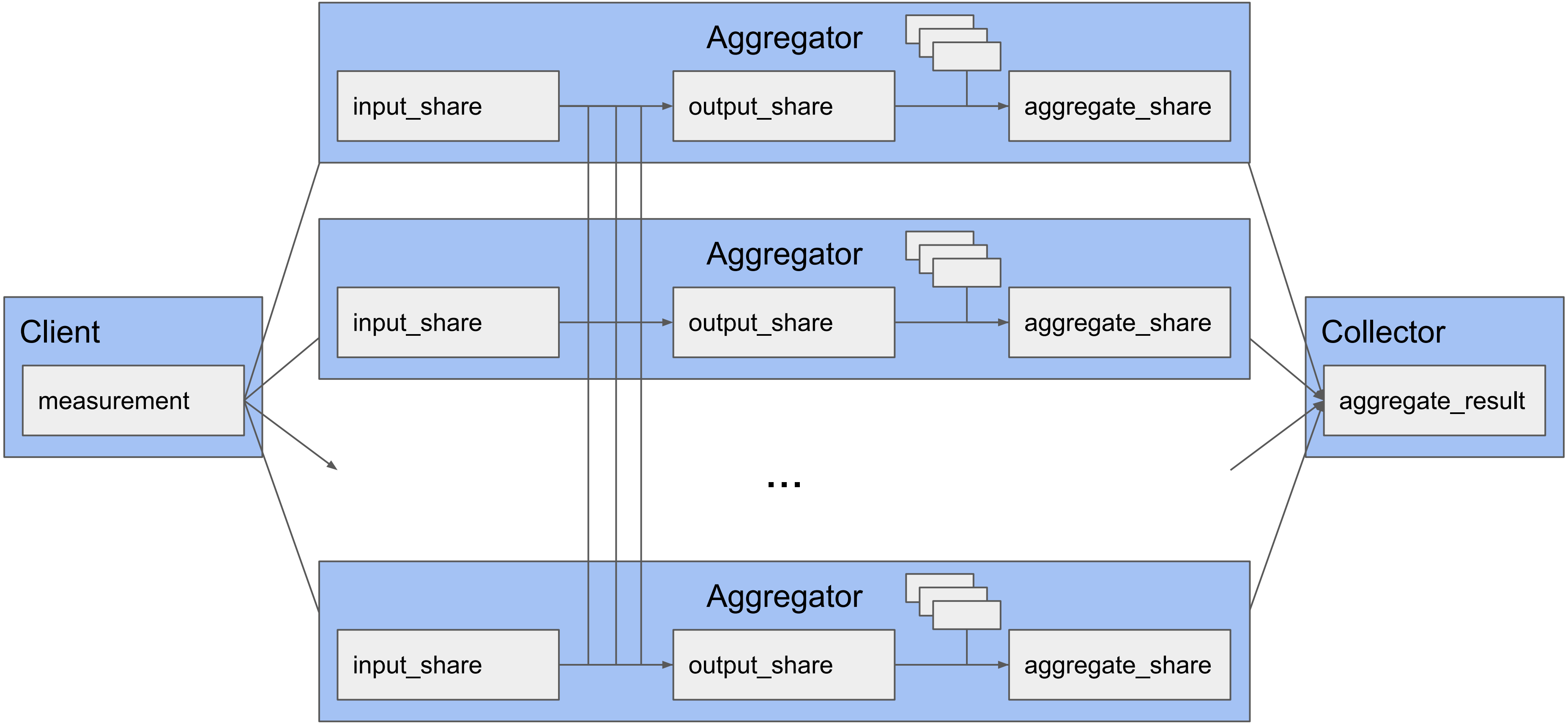


Client

Collector



Shard → **Prepare** → **Aggregate** → **Unshard**



Shard → Prepare → Aggregate → Unshard

API

Setup		`vdaf_setup()`
Shard	Client	`measurement_to_input_shares(public_param, input)`
Prepare	Aggregator	`PrepState(verify_param, agg_param, nonce, input_share)`
		`PrepState.next(inbound: Vec[Bytes])`
Aggregate	Aggregator	`output_to_aggregate_shares(agg_param, output_shares)`
Unshard	Collector	`aggregate_shares_to_result(agg_param, agg_shares)`

PPM'S JOB IS TO GET THE RIGHT DATA TO THE RIGHT PLACES AT THE RIGHT TIMES!

Constructions of VDAFs

- prio3 [CBG17, BBCG+19]
  - Encode each measurement m as vector x of elements of a finite field
  - Aggregation parameter: number of measurements n
  - Any aggregation function of the form f(n, x[1] + … + x[n])
  - Any number of aggregators
  - **Dist-Prepare**: C(x)=0 for arithmetic circuit C that defines validity
- hits [BBCG+21]
  - Measurement: N-bit string (encoded as IDPF shares)
  - Aggregation parameter: sequence of P-bit strings (the "candidate prefixes") where P <= N
  - Aggregation function: how many inputs are prefixed by each candidate
  - Two aggregators
  - **Dist-Prepare**: input is prefixed by at most one candidate
- … and many more!

Implementations (so far)

- Rust github.com/abetterinternet/libprio-rs
  - prio3
  - hits (proof-of-concept only, missing efficient IDPF)
  - "Prio v2" (used in ENPA)
- C++ github.com/google/distributed_point_functions
  - IPDF
- C++ github.com/google/libprio-cc
  - "Prio v2" (used in ENPA)
- C github.com/mozilla/libprio
  - "Prio v1" (used in Origin Telemetry)

prio3 client perf (two aggregators)		
aggregation function	shard time	communication
count	8 μs	208 bytes
histogram (10 buckets)	15 μs	432 bytes
sum (32 bit integers)	35 μs	960 bytes

Fin

References

- [CGB17] Corrigan-Gibbs-Boneh. "Prio: Private, Robust, and Scalable Computation of Aggregate Statistics". NSDI 2017.
- [BBCG+19] Boneh et al. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs". CRYPTO 2019.
- [BBCG+21] Boneh et al. "Lightweight Techniques for Private Heavy Hitters". IEEE S&P 2021.