Verifiable Distributed Aggregation Functions

draft-patton-cfrg-vdaf-00

Presented at IETF 112 (CFRG)
Authors: Christopher Patton (speaker), Richard Barnes, Phillipp Schoppmann

Motivation

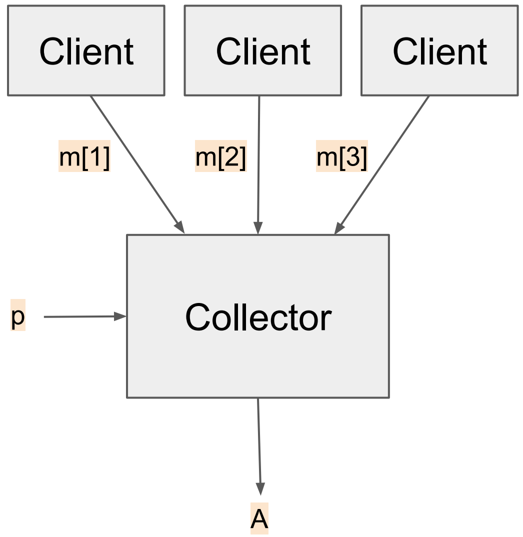
- PRIV BoF (Wednesday): Privacy preserving aggregation of user measurements
  - Privacy made possible by distributing the computation across multiple servers
  - Coordination required to ensure correctness of the computation
- Lots of recent work in the literature, but...
  - There is no "one-size-fits-all" solution
    - Each protocol is tailored to a particular (class of) aggregation functions
  - Protocols vary in their security and operational considerations
    - Lack consistent abstraction boundary for PRIV (and other standardization efforts) to build upon

Objective of this draft

- Provide an abstraction boundary (VDAF) that:
  - addresses the security/operational considerations of real-world deployments (ENPA, Origin Telemetry)
  - provides design criteria for cryptographers to build new and improved schemes
- Standardize a few VDAFs from the literature
  - in particular, those discussed in PRIV so far

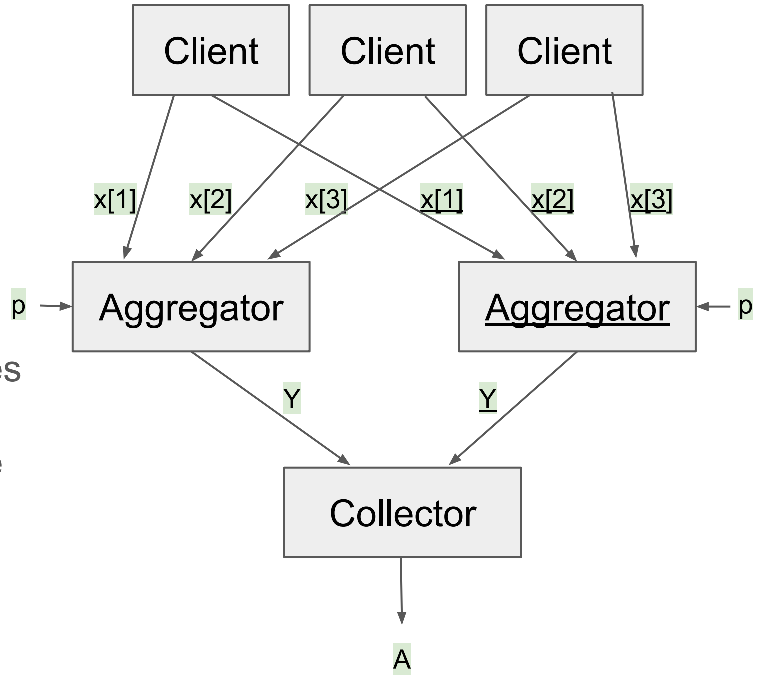
Verifiable Distributed Aggregation Function (VDAF)

- Want to compute A := F(p, m[1], …, m[n])
  - m[1], …, m[n] are the client measurements
  - p is the aggregation parameter
  - A is the aggregate result
- Examples:
  - A is arithmetic mean
  - A is a histogram estimating the distribution
  - A counts how many times p occurs in m[1], …, m[n]



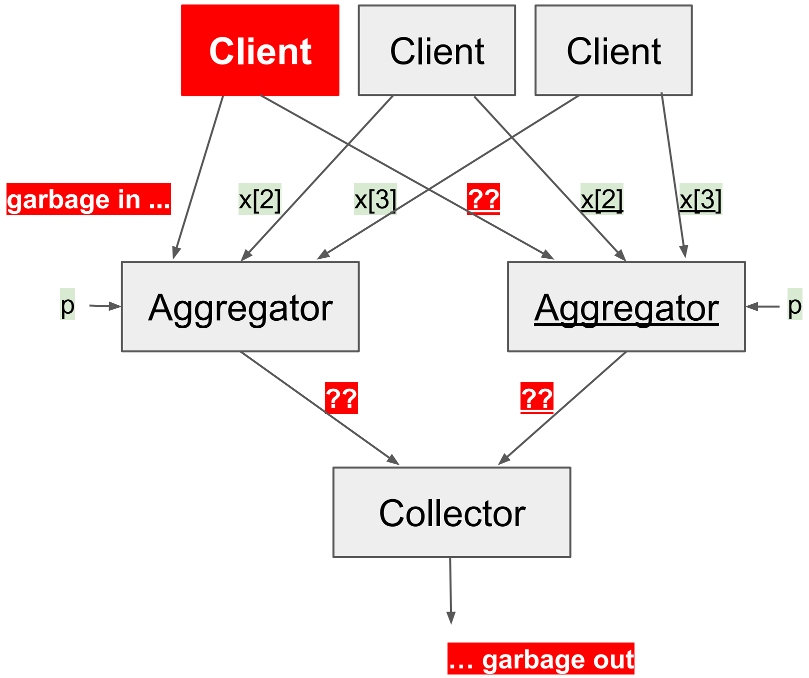
Verifiable Distributed Aggregation Function (VDAF)

- Privacy via secret sharing
  - **Shard(m)→(x, x):** Client shards its measurement into input shares and distributes them among the Aggregators.
  - **Prepare(p, x)→y:** Aggregator maps the aggregation parameter and its input share to its output share (e.g., DPFs).
  - **Aggregate(y[1], …, y[n])→Y:** Aggregator combines output shares to get its aggregate share.
  - **Unshard(Y, Y)→A:** Collector combines aggregate shares to get aggregate.
- Correctness
  - (y[1] + y[2] + y[3]) + (y[1] + y[2] + y[3]) = Y + Y = A



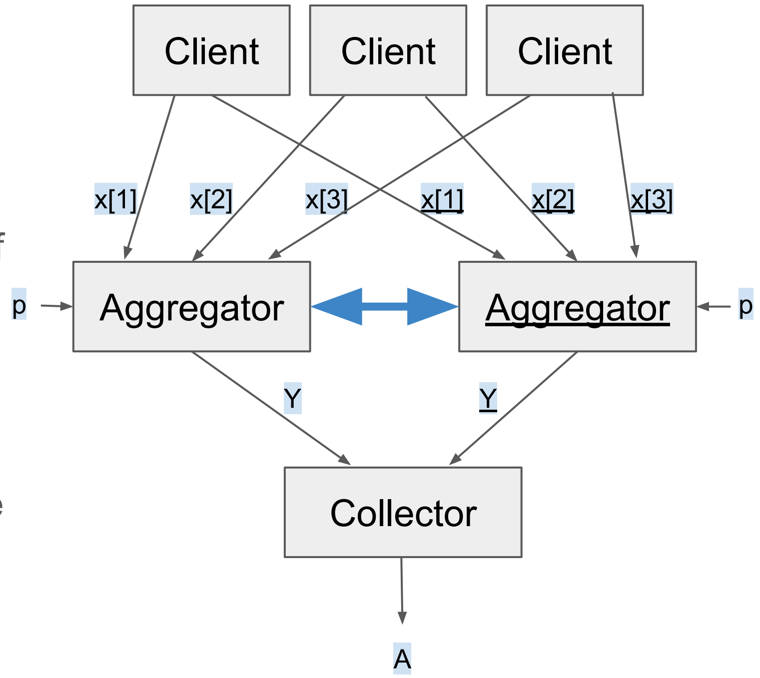
Verifiable Distributed Aggregation Function (VDAF)

- What about **malicious** (or merely misconfigured) clients?
  - (**??** + y[2] + y[3]) + (**??** + y[2] + y[3]) = **??**



Verifiable Distributed Aggregation Function (VDAF)

- Robustness via multi-party computation
  - **Shard(m)→(x, $\underline{x}$):** Client shards its measurement into input shares and distributes them among the Aggregators.
  - **Prepare:** Aggregators engage in a secure MPC of (y, $\underline{y}$) := **Dist-Prepare**(p, x, $\underline{x}$).
  - **Aggregate(y[1], …, y[n])→Y:** Aggregator combines output shares to get its aggregate share.
  - **Unshard(Y, $\underline{Y}$)→A:** Collector combines aggregate shares to get aggregate.
- *Not* general-purpose MPC!



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 $f(n, x[1] + \ldots + x[n])$
  - 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 $P \le N$
  - 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

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.