Verifiable Distributed Aggregation Functions

draft-patton-cfrg-vdaf
IETF 112
November 2021
<table>
<thead>
<tr>
<th>Protocol</th>
<th>TLS [RFC8446]</th>
<th>MLS [draft-ietf-mls-protocol]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DH, Signature, Hash [RFC7748] [RFC8032]</td>
<td>HPKE [draft-ietf-cfrg-hpke]</td>
</tr>
<tr>
<td>Crypto</td>
<td>PPM [draft-gpew-priv-ppm]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VDAF [draft-patton-cfrg-vdaf]</td>
<td></td>
</tr>
</tbody>
</table>
API

Instantiations:

prio3 - Aggregate statistics
hits - Most common strings
**What does a VDAF do?**

**VD Aggregation** - Compute a statistic over batch of measurements without revealing anything about the individual measurements.

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

**Verifiable** - The aggregators can check the correctness of client’s inputs to prevent malicious or misconfigured clients from corrupting aggregates.
## API

<table>
<thead>
<tr>
<th>Step</th>
<th>Role</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>Client</td>
<td><code>vdaf_setup()</code></td>
</tr>
<tr>
<td>Shard</td>
<td>Client</td>
<td><code>measurement_to_input_shares(public_param, input)</code></td>
</tr>
<tr>
<td>Prepare</td>
<td>Aggregator</td>
<td><code>PrepState(verify_param, agg_param, nonce, input_share)</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>PrepState.next(inbound: Vec[Bytes])</code></td>
</tr>
<tr>
<td>Aggregate</td>
<td>Aggregator</td>
<td><code>output_to_aggregate_shares(agg_param, output_shares)</code></td>
</tr>
<tr>
<td>Unshard</td>
<td>Collector</td>
<td><code>aggregate_shares_to_result(agg_param, agg_shares)</code></td>
</tr>
</tbody>
</table>

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 $P \leq N$
  - 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](https://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](https://github.com/google/distributed_point_functions)
  - IPDF
- **C++** [github.com/google/libprio-cc](https://github.com/google/libprio-cc)
  - "Prio v2" (used in ENPA)
- **C** [github.com/mozilla/libprio](https://github.com/mozilla/libprio)
  - "Prio v1" (used in Origin Telemetry)

<table>
<thead>
<tr>
<th>aggregation function</th>
<th>shard time</th>
<th>communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>8 μs</td>
<td>208 bytes</td>
</tr>
<tr>
<td>histogram (10 buckets)</td>
<td>15 μs</td>
<td>432 bytes</td>
</tr>
<tr>
<td>sum (32 bit integers)</td>
<td>35 μs</td>
<td>960 bytes</td>
</tr>
</tbody>
</table>
FIN
References