What's new in draft-ietf-ppm-dap-01

PPM - IETF 114
Implementation status

- Implementations of `draft-ietf-ppm-dap-01` are available on GitHub:
  - [Daphne](#), server, Rust
  - [Janus](#), server, Rust
  - [divviup-ts](#), client, TypeScript

- `draft-ietf-ppm-dap-01` depends on `draft-irtf-cfrg-vdaf-02`:
  - Rust implementation of vdaf-01 in `libprio-rs / crate prio`
  - Crate prio still needs a Poplar1 implementation to fully implement vdaf-02

- Interop testing between Daphne and Janus is underway
- Working on designing a DapInteropRunner inspired by `QuicInteropRunner`
Coarse-grained report timestamps

- Nonces must be unique for anti-replay and timestamped for inclusion in a batch interval
- High resolution time leaks information about the client
- Rounding down the timestamp and widening the random component protects privacy while meeting nonce requirements
- Issue [#274](http://example.com/) / PR [#281](http://example.com/) - Thanks to Shan Wang for the great idea!
Aggregation jobs

- Aggregation sub-protocol coordinates *preparation* of each *input share* into an *output share*
- Multiple rounds of stateful communication (2-3, depends on VDAF)
- Preparation means evaluating proofs, possibly transforming inputs somehow

```
struct {
    TaskID  task_id;
    AggregationJobID job_id;
    opaque  agg_param<0..2^16-1>;
    ReportShare report_shares<1..2^16-1>;
} AggregateInitializeReq;
```

- Leader creates mapping of one *aggregation job ID* to many report shares
- Several aggregation jobs may be required to prepare all reports in a batch
Aggregation jobs

- Helper uses job ID to index into its storage to fetch state
- Many helpers can work in parallel provided they share storage
- Job IDs are not secret and don't need anti-replay protections
- Issue #185 / PR #232
Inter-aggregator authentication

- In aggregate sub-protocol, leader is client to helper HTTP server
- This channel must be mutually authenticated
- **PR #328** mandates that leader set a `DAP-Auth-Token` header in its requests with a pre-negotiated secret as the value
- Sufficient for current deployments but:
  - Requires a shared secret between protocol participants
  - Precludes numerous existing authn/authz mechanisms for HTTP APIs
## Survey of channel security in draft-ietf-ppm-dap-01

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Design requirement</th>
<th>Specified mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client ⇒ aggregator</td>
<td>1. Confidentiality</td>
<td>1. HPKE encryption to each aggregator</td>
</tr>
<tr>
<td></td>
<td>2. Server authentication</td>
<td>2. HPKE config fetched over TLS</td>
</tr>
<tr>
<td>Leader⇔Helper</td>
<td>1. Confidentiality</td>
<td>1. TLS?</td>
</tr>
<tr>
<td></td>
<td>2. Mutual authentication</td>
<td>2. Pre-negotiated bearer token (for now) and server TLS certificate</td>
</tr>
<tr>
<td>Collector⇔Leader</td>
<td>1. Confidentiality</td>
<td>1. TLS, HPKE encryption of aggregate share</td>
</tr>
<tr>
<td></td>
<td>2. Mutual authentication</td>
<td>2. Pre-negotiated bearer token (for now) and server TLS certificate</td>
</tr>
<tr>
<td>Collector⇔Helper</td>
<td>1. Confidentiality</td>
<td>1. TLS, HPKE encryption of aggregate share</td>
</tr>
<tr>
<td></td>
<td>2. Mutual authentication</td>
<td>2. Nothing (yet; mutual HPKE?)</td>
</tr>
</tbody>
</table>
What should DAP say about request authentication?

- Straw man: say *nothing*. Stipulate requirements, not solutions.
- DAP is built on HTTP, thus it can rely on existing mechanisms and implementations for:
  - Caching
  - Error handling
  - Authentication
- DAP should aim for composability with existing HTTP authn schemes:
  - AWS request signatures
  - OAuth 2
  - TLS client certificates
- HPKE is used only where we tunnel a secure channel through another participant
Some goals for draft-item-ppm-dap-02

● Rewrite DAP HTTP API to be resource-oriented
  ○ e.g., replace POST [aggregator]/upload with PUT [aggregator]/tasks/<task_id>/reports/<report_id>
● Align with BCP 56, BCP 190 guidance where reasonable
  ○ Better use of HTTP semantics
  ○ Extend hpke_config into an ACME style API directory?
● Revisit request authentication design requirements and prescriptions
● Looking forward to hashing out these ideas in the working group!