Open issues for DAP

IETF 118 - PPM - Christopher Patton
#436, 409, 405, 316, 259 Collecting a batch many times

- We sometimes want to collect a batch multiple times: drill-down (#489); heavy-hitters (Poplar1)
  - **Requirements**: Enforce aggregation parameter validity, per [draft-irtf-cfrg-vdaf, Section 5.3](#489)
  - **Problem**: No one has implemented this (not required for Prio3), so we don't know yet if the spec is correct
    - Sub-optimal communication (#409, 405)
    - Potential bugs (#436, 316, 259)
    - Incomplete definitions ("batch" is ill-defined in the context of multiple collections)
  - **Proposal #1**: Someone implement it and propose a PR to address any issues
  - **Proposal #2**: Remove support for collecting a batch multiple times (i.e., don't support heavy-hitters)
Batch selection as Collector-Leader "business logic"

DAP needs a way for the Aggregators to partition reports into batches

- Different Batching strategies formalized as "query types" (Time-interval, Fixed-size, …?) that give the Collector some in-band control over batch selection
- Problem: Supporting multiple query types adds complexity for implementations
  - Observation: Fixed-size is general enough to support many batching strategies as out-of-band "business logic" implemented Collector and Leader

- Proposal #1: Remove query types and adopt Fixed-size semantics (Leader arbitrarily assigns reports to batches identified by batch IDs)
  - What do implementers think?
- Proposal #2: Do nothing (implementations are free to ignore query types)
#489 Supporting drill-down

- **Use case:** Collector wants to split aggregate result by arbitrary "labels" (user-agent, geolocation, etc.)
  - **Problem:** Currently requires configuring a task for each label ⇒ lacks flexibility, doesn't scale, we miss out on data for "unpopular" labels

  - **Proposal #1:** Add [labels to report metadata](https://draft-mouris-cfrg-mastic), enrich queries to support label sets
    - Problem: Labels are fingerprintable
    - Problem: Still need to enforce the same minimum batch size

  - **Proposal #2** (not mutually exclusive with #1): Do per-label aggregation in MPC ([draft-mouris-cfrg-mastic](https://draft-mouris-cfrg-mastic))
    - Perhaps not as flexible as we need (can do `label1=="value1" && label2=="value2"` but can't do `label1=="value1" || label label2=="value2"`)
Agreement on task parameters

- Desirable property: Honest parties that execute a task agree on the parameters of that task.
  - **Requirement**: Successful completion of the upload, aggregation, or collect sub-protocol should imply agreement on task configuration.

- **Proposal #1**: [draft-wang-ppm-dap-taskprov](#) derives task ID from serialized task config
  \[\Rightarrow\] agreement on task ID implies agreement on task parameters

- **Proposal #2**: Add specific parameters to AAD for HPKE encryption

- **Proposal #3**: "The application MUST implement some mechanism for enforcing agreement on the task configuration."
Recovering after batch mismatch

- Batch mismatch (Leader and Helper don't agree on the set of reports in the batch) is currently fatal.
  - **Proposal #1**: Do nothing, since (1) we can detect batch mismatches and (2) batch mismatch is unlikely
    - Can happen if: one Aggregator's storage gets corrupted; other reasons?
  - **Proposal #2**: Add mechanism allowing the Leader to find the missing reports and retry them
Cheaper checksum

During collection, the Aggregators check for batch mismatch by computing a checksum over the reports.

- **Problem**: The current checksum looks more expensive than necessary. Can't just get rid of it because it has been useful for detecting issues in implementations.

- **Question**: If the attacker controls a subset of Clients and can trigger a network error that causes a batch mismatch, then it can choose report IDs such that the Aggregators compute the same checksum (and thus fail to detect the batch mismatch). **Do we care?**

- **Requirement**: Checksum computation must be independent of the order of reports.

- **Proposal #1**: Make it cheaper
- **Proposal #2**: Make it optional
- **Proposal #3**: Do nothing because it's relatively inexpensive
Deviations from TLS-syntax

- Protocol messages are specified in "TLS-syntax" from RFC 8446, Section 3.
  - Problem: We deviate from a strict interpretation of this spec
    - Proposal #1: Extend TLS-syntax to meet our needs
    - Proposal #2: Fully comply with TLS-syntax as it is (explain things in prose as needed)
    - Proposal #3: Explain deviations when they arise and limit them as much as possible

```c
struct {
    PrepareStepState prepare_step_state = 2; /* reject */
    ReportId report_id;
    ReportShareError report_share_error;
} PrepareStep;
```

draft-ietf-ppm-dap-07, Section 4.5.1.2

draft-irtf-cfrg-vdaf-07, Section 5.8
#459 GET {aggregator}/hpke_config

- **Idea**: Make this endpoint "look like" the others
  - Proposal #1 (PR #510): Add task ID ⇒ {aggregator}/tasks/{task-id}/hpke_config
  - Proposal #2: Do nothing, as this issue is more aesthetic than anything.
#450 PUT or POST {leader}/tasks/{task-id}/reports

- We currently PUT, which contradicts RFC 9110, Section 9.3.4 (we're not "replacing" the resource of the request path)
  - **Question**: Is this an issue for upload only, or is it also an issue for aggregation and collection?
    - If so, then **Proposal #1**: Add the report ID to the request path
Backup slides
# Poplar1 versus Mastic (draft-mouris-cfrg-mastic)

<table>
<thead>
<tr>
<th></th>
<th>Poplar1</th>
<th>Mastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>heavy hitters</td>
<td>yes</td>
<td>yes*</td>
</tr>
<tr>
<td>weighted heavy hitters</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>&quot;Prio with labels&quot;</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>primitives</td>
<td>IDPF + &quot;secure sketch&quot;</td>
<td>&quot;verifiable&quot; IDPF + FLP</td>
</tr>
<tr>
<td>number of aggregators</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>prep rounds</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>overall communication (bits)</td>
<td>–</td>
<td>a little higher*</td>
</tr>
<tr>
<td>overall computation</td>
<td>–</td>
<td>about the same</td>
</tr>
</tbody>
</table>

*VIDPF-proof aggregation