Open issues

PPM interim – 2024/10/24

Changes since IETF 120

draft-irtf-cfrg-vdaf-12:

- Various improvements for Prio3 and Poplar1 (change log)
- New **application context** API for defense-in-depth

draft-irtf-cfrg-vdaf-13 (work in progress):

- Editorial work
- New **streaming** aggregation API
- *Ready for RGLC???*

Changes since IETF 120

draft-ietf-ppm-dap-12:

- PR #564: Allow the Helper to handle each step of aggregation asynchronously
- Move to draft-irtf-cfrg-vdaf-12
- Clean up unused features:
  - Remove max batch size from leader-selected batch mode
  - Remove support for per-task HPKE configs
- Various wire breaking changes we've been holding off on (more clean up)

draft-ietf-ppm-dap-13 (work in progress):

- Editorial work, improvements to IANA considerations (still some open questions)

Changes since IETF 120

draft-ietf-ppm-dap-taskprov-00:

- **Adopted** draft-wang-ppm-dap-traskprov-07 plus some minor editorial changes sitting on `main`

draft-ietf-ppm-dap-taskprov-01 (work in progress):

- Various breaking changes we've been holding off on:
  - Add salt to task ID computation
  - Clean up an unused parameters (`max_batch_query_count`, `max_batch_size`)
  - Harmonize extension points: `BatchModeConfig`, `VdafConfig`, and `DpConfig` all have the same structure
- Move to draft-ietf-ppm-dap-12 and draft-irtf-cfrg-vdaf-12
- Improve IANA considerations (one more open question)

Open issues

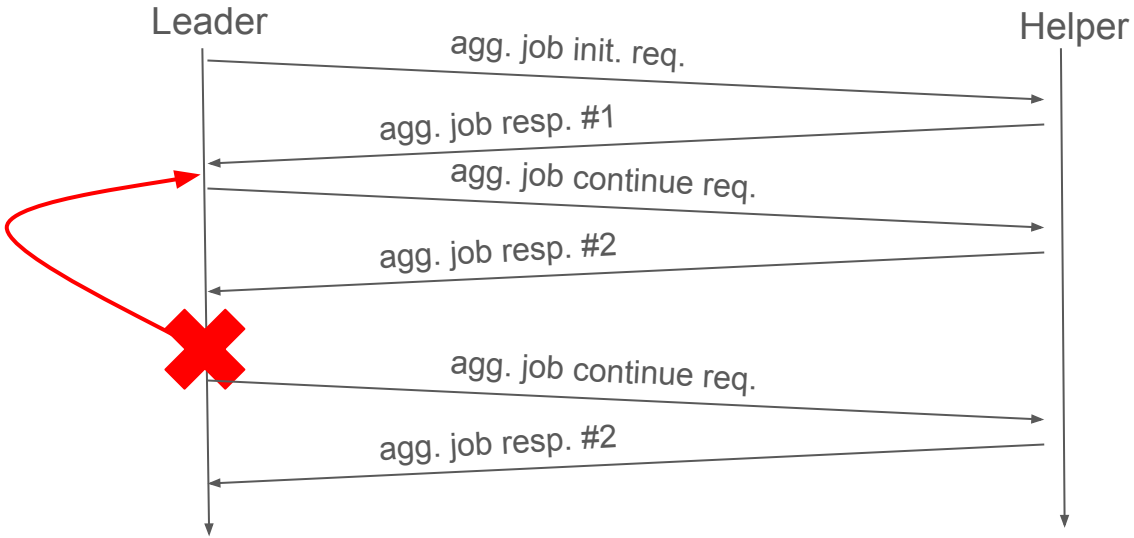
- Aggregation skew recovery
- Extensibility
- Miscellaneous

Aggregation skew recovery

Aggregation skew recovery (issue #604)

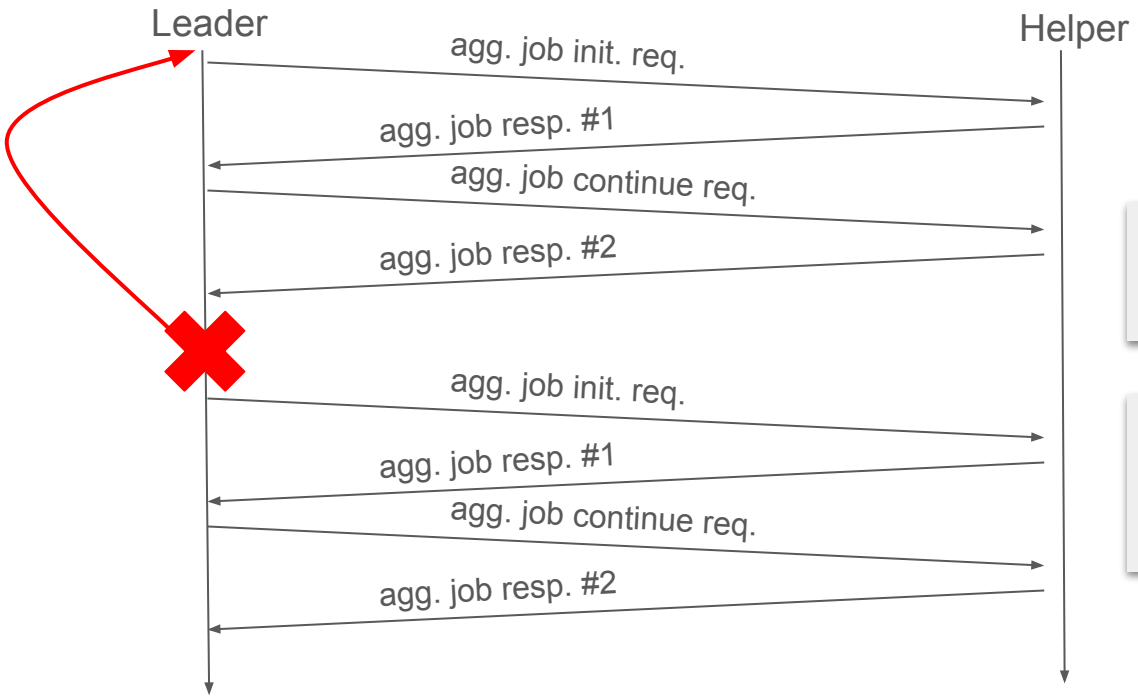
Disaster recovery if Leader crashes or a response is lost in transit during some step of computation during an aggregation job. The Leader MAY retry; the Helper SHOULD help it recover.

Status quo: The Leader can retry the previous step; the Helper remembers last request & response. (Precisely the same requirement as idempotent request handling.)



Aggregation skew recovery (issue #604)

Problem #1 (operational): What if the Leader needs to restart from the beginning?

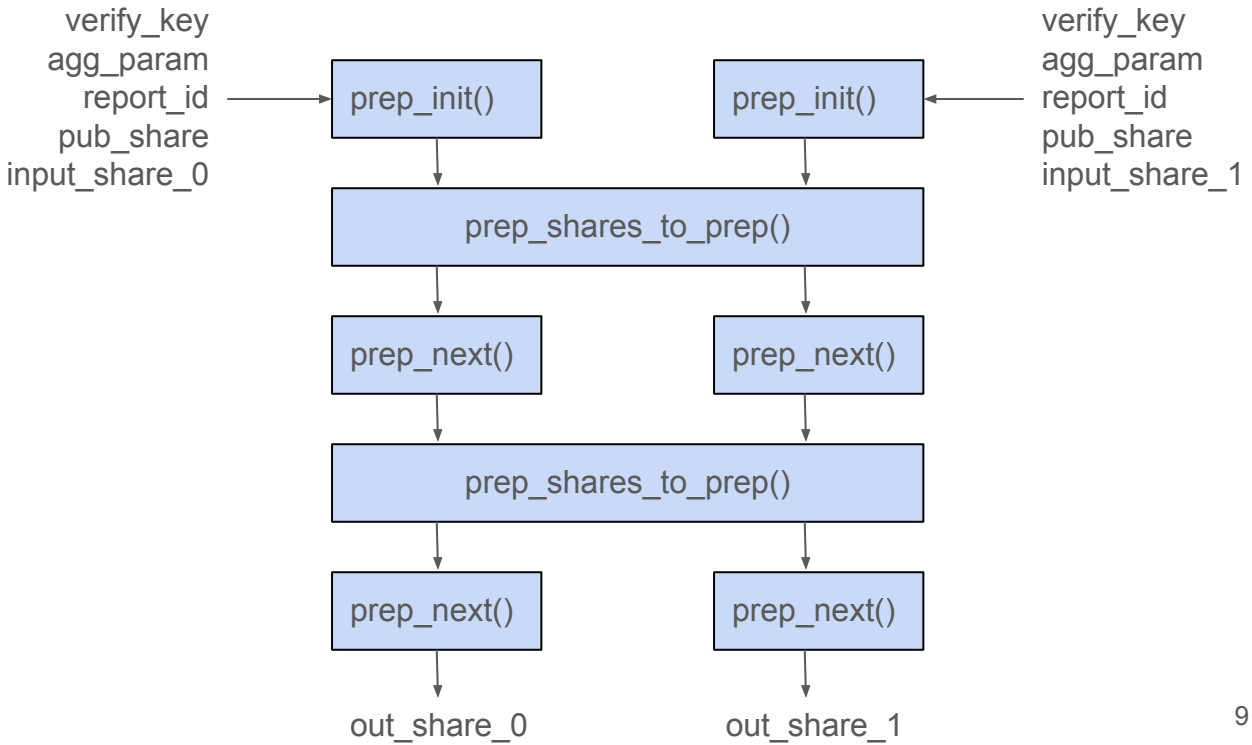


PR #569: Helper retains enough information to rewind from any step

PR #616: Helper retains enough information to rewind from the beginning

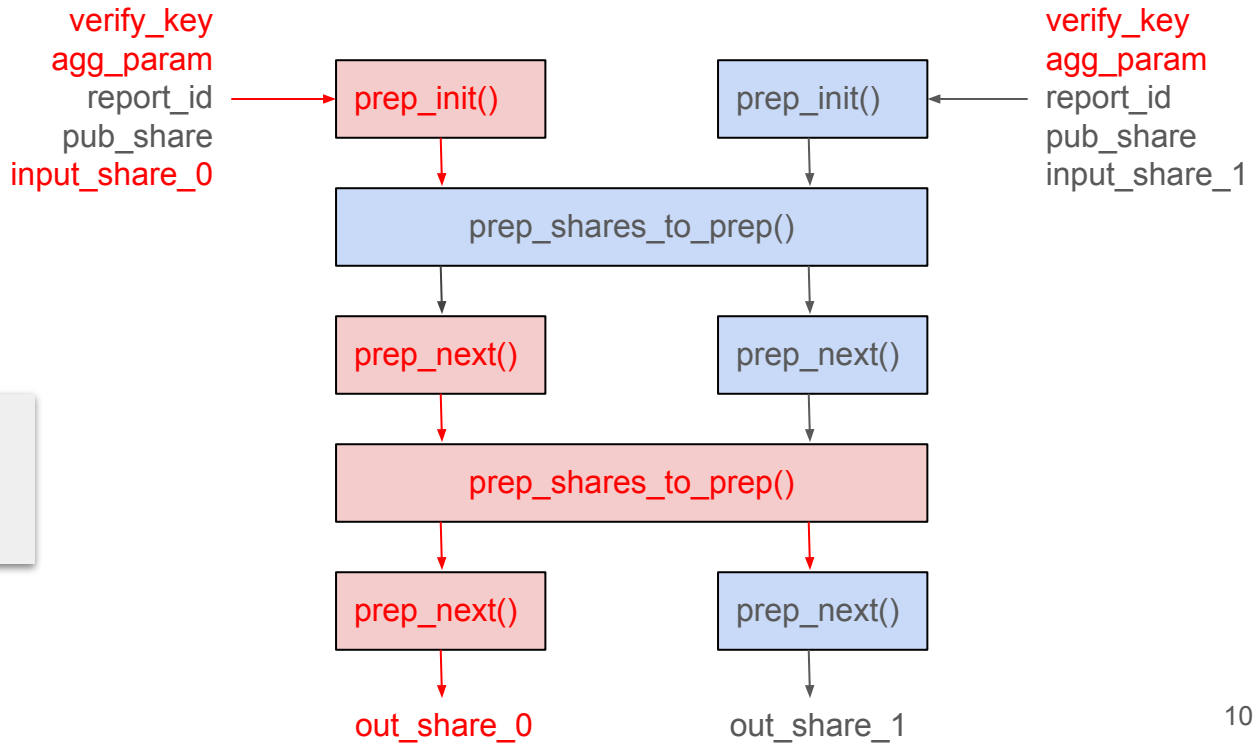
Aggregation skew recovery (issue #604)

Problem #2 (privacy): "Rewind-then-fork" attacks on VDAF preparation



Aggregation skew recovery (issue #604)

Problem #2 (privacy): "Rewind-then-fork" attacks on VDAF preparation

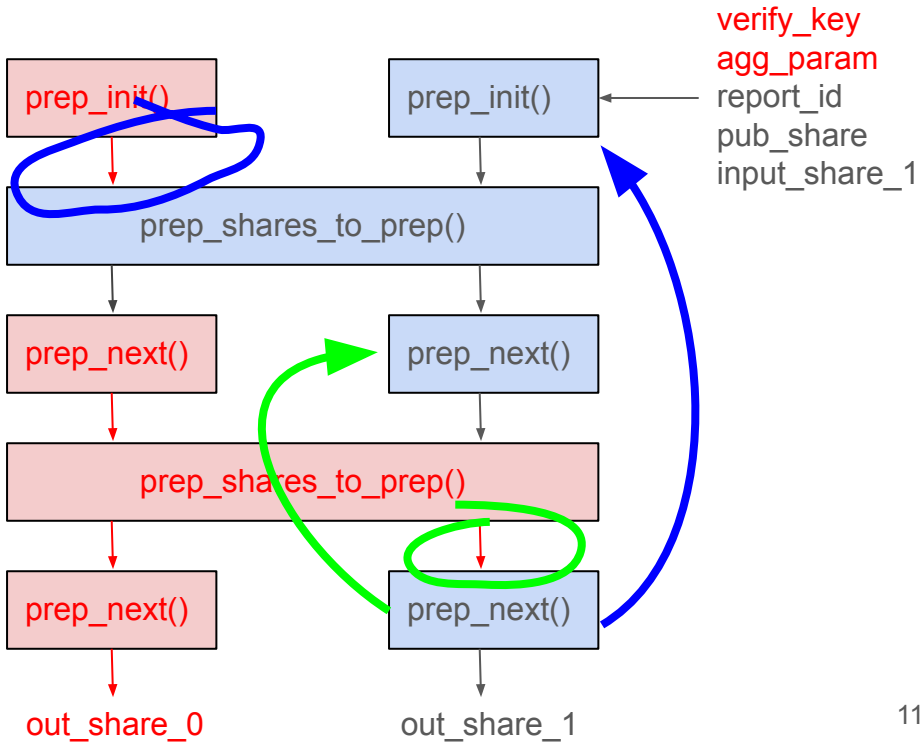


In our threat model, the attacker partially controls the computation and the inputs at each step.

Aggregation skew recovery (issue #604)

Problem #2 (privacy): "Rewind-then-fork" attacks on VDAF preparation

Attack: rewind to **previous step** or to **the beginning** and rerun it with different inputs



Extensibility

Public report extensions (issue #620)

- Add a field to `ReportMetadata` for report extensions: each Aggregator applies current extension processing logic to its private (i.e., encrypted) extensions *and* the public extensions (also need to add public extensions to AAD).
  - Reduce communication cost during upload interaction
  - One less edge case: suppose a Client mistakenly sends an extension to one Aggregator but not the other
  - A bit more protocol complexity
- ***Remove*** private extensions?
  - Reduce complexity
  - Might break existing use cases for private extensions (e.g., draft-priebe-ppm-dap-reportauth)

Probing for extension support (issue #619)

- DAP has no mechanism to advertise which parties support which report extensions: if an Aggregator does not recognize an extension, it rejects the report.
- Leader MAY reject reports with unrecognized extensions during upload interaction, but does not indicate which extensions aren't supported
    - There's no way for the Client to probe the Helper at all
- Considerations:
    - Enumerating extensions that are not standard or you don't want to advertise support for
    - Implicit in the design of DAP: the Client already needs to know "who" the Aggregators are and which features they support

Extending taskprov (taskprov issue #85)

```
# Extending this Document

The behavior of the `taskbind` extension may be extended by future documents
that define:

1. A new DAP batch mode
1. A new VDAF
1. A new DP mechanisms

Such documents SHOULD include a section titled "Taskbind Considerations" that
specifies an encoding of any configuration parameters associated with the DAP
batch mode, VDAF, or DP mechanism. This encoding extends the `BatchConfig`
structure, the `VdafConfig` structure, or the `DpConfig` structure
respectively.

Note that the registry for batch modes is defined by {{!DAP}}; the registry for
VDAFs is defined by {{!VDAF}}; and the registry for DP mechanisms is defined in
{{dp-mechanism-registry}} of this document.
```

```
struct {
    DpMechanism dp_mechanism;
    select (DpConfig.dp_mechanism) {
        case none: Empty;
    };
} DpConfig;
```

General idea: Extension points allow parsing unrecognized variants; new documents "modify" structures in the document by specifying new variants

Extending DAP with new batch modes (issue #622)

DAP has the same problem as taskprov issue #85: Documents that define new batch modes will need to extend the following structs:

- `Query` in the collection request (used to compute the aggregate share)
- `PartialBatchSelector` in the aggregation request (used to map reports to batches)

Could do the same thing as taskprov, **but at the moment we can't handle unrecognized variants**.

```
struct {
  BatchMode batch_mode;
  select (Query.batch_mode) {
    case time_interval: Interval batch_interval;
    case leader_selected: Empty;
  }
} Query;
```

```
struct {
  BatchMode batch_mode;
  select (PartialBatchSelector.batch_mode) {
    case time_interval: Empty;
    case leader_selected: BatchID batch_id;
  };
} PartialBatchSelector;
```

Miscellaneous

Streaming aggregation (issues #385, #581)

PR #623:

- Replace `vdaf.aggregate(out_shares)` → `agg_share` with new streaming API:
  - **initialize** each **batch bucket** `b` with `agg_share[b] = vdaf.agg_init()`
  - aggregation: for each `out_share` for batch bucket `b`:
    - **commit** `agg_share[b] = vdaf.agg_update(agg_share[b], out_share)`
  - collection: the query specifies a **batch bucket span** `b1, …, bN`. Compute the aggregate share for the query as `vdaf.merge(agg_share[b1], …, agg_share[bN])`

Task start time (issue #606, taskprov issue #62)

PR #624 (taskprov PR #76):

- Add start time to task parameters and MAY reject reports with timestamps outside the validity window (smaller than the start time or larger than or equal to the end time)
- Related (issue #625): Aggregators MUST reject reports outside the validity window?

DP mechanisms (taskprov issue #83)

We have only reserved 1 byte for `DpMechanism` variants. Might we need more?

PR #86: Bump to 4 bytes (same as for VDAF codepoints)