Privacy Proofs for OpenDP: Partition Map (Measurement)

Summer 2022

Contents

1 Algorithm Implementation

1.1 Code in Rust

The current OpenDP library contains the `make_map_partition_meas` function constructing a `Measurement` for partitioned data based on a list of `Measurements` (parallel composition). This is defined in lines 76-136 of the file `mod.rs` in the Git repository https://github.com/opendp/opendp/blob/449222066a006a19d15bb68e62a87105ce49eb15/rust/src/comb/partition_map/mod.rs#L76-L136.

1.2 Pseudo Code in Python

Preconditions

To ensure the correctness of the output, we require the following preconditions:

- `User-specified types:`
  - Variable `measurements` must be a vector of elements of class `Measurement`. [1]
  - `output_measure::Distance` must have trait `TotalOrd`.
  - `input_domain`: must have trait `Domain`.
  - `output_domain`: must have trait `Domain`.
  - `input_metric`: must have trait `Metric`.
  - `output_measure`: must have trait `Measure`.

Postconditions

- A `Measurement` is returned (i.e., if a `Measurement` cannot be returned successfully, then an error should be returned).

[1] To be defined in the pseudocode defs doc.

Pseudo Code

```python
def make_map_partition_meas(measurements: List[Measurement]) -> Measurement:
    input_domain = ProductDomain(m.input_domain for m in measurements)
    output_domain = ProductDomain(m.output_domain for m in measurements)
    def function(data: Vec<DI::Carrier>) -> Vec<DO::Carrier>:
        output = []
        for part, m in zip(data, measurements):
            output.append(m.function(part))
        return output
    input_metric = ProductMetric(measurements[0].input_metric)
    output_measure = measurements[0].output_measure
    def privacy_map(d_in: input_metric::Distance) -> output_measure::Distance:
        return max(m.map(d_in) for m in measurements)
    return Measurement(input_domain, output_domain, function, input_metric, output_measure, privacy_map)
```

2 Proof

The necessary definitions for the proof can be found at "List of definitions used in the proofs".

Theorem 2.1. *For every setting of the input parameter `measurements` to `make_map_partition_meas` such that the given preconditions hold, `make_map_partition_meas` raises an exception (at compile time or runtime) or returns a valid `Measurement` with the following privacy guarantee:*

1. **(Domain-metric compatibility.)** *The domain `input_domain` matches one of the possible domains listed in the definition of `input_metric`.*

2. **(Privacy guarantee.)** *For every pair of elements v, w in `input_domain` and for any `d_in`, where `d_in` has the associated type for `input_metric`, if v, w are `d_in`-close under `input_metric`, then `function(v)`, `function(w)` are `privacy_map(d_in)`-close under `output_measure`.*

Proof.

1. **(Domain-metric compatibility.)** The `input_domain` of `make_map_partition_meas` is `ProductDomain` and the `input_metric` is `ProductMetric`. Since each component `m` of the variable `measurements` is a `Measurement`, `m.input_domain` matches one of the possible domains listed in the definition of `m.input_metric`. Therefore, the product of `m.input_domain` is compatible with the product of `m.input_metric`.

2. **(Privacy guarantee.)** Let $\mathtt{m}_i$ be the ith element of the variable `measurements`, thus $\mathtt{m}_i$ is a `Measurement`. Let `MI` and `MO` denote the `input_metric` and the `output_measure` of $\mathtt{m}_i$, respectively. For any i and v_i, w_i in the input domain of $\mathtt{m}_i$,

$$d_{\mathtt{MI}}(v_i, w_i) \leq \mathtt{d\_in}$$

implies

$$d_{\mathtt{MO}}(f_i(v_i), f_i(w_i)) \leq \mathtt{m}_i.\mathtt{privacy\_map(d\_in)},$$

where f_i denotes the function in $\texttt{m}_i$, and v_i and w_i are the ith component of v and w, respectively. Let $d_{\texttt{PM,MI}}$ denote the distance under `ProductMetric(MI)`. Then by Definition 1(i),

$$d_{\texttt{PM,MI}}(v, w) = \sum_i d_{\texttt{MI}}(v_i, w_i).$$

Note that, given $d_{\texttt{PM,MI}}(v, w) \leq \texttt{d\_in}$, by the nature of partitioned data, all $d_{\texttt{MI}}(v_i, w_i)$ are zeros except for at most one. We then have

$$\begin{aligned} d_{\texttt{MO}}(g(v), g(w)) &\leq max_i(d_{\texttt{MO}}(f_i(v_i), f_i(w_i))) \\ &\leq max_i(\texttt{m}_i.\texttt{privacy\_map}(\texttt{d\_in})), \end{aligned}$$

where g is the `function` in `make_map_partition_meas`.

Therefore, we have shown that $g(v)$ and $g(w)$ are `privacy_map`(`d_in`)-close if v and w are `d_in`-close.

□

Definition 1 (Distance under `ProductMetric`)**.** *Let $d_{\texttt{PM,M}}$ denote the distance under `ProductMetric(M)` where `M` is a valid metric. Then $d_{\texttt{PM,M}}$ is defined as the sum of distance under each `M`. Specifically, for any v, w in the input domain and v_i, w_i denote their ith entry, respectively,*

(i) for input metric `MI`,

$$d_{\texttt{PM,MI}}(v, w) = \sum_i d_{\texttt{MI}}(v_i, w_i).$$

(ii) for output metric `MO`,

$$d_{\texttt{PM,MO}}(g(v), g(w)) = \sum_i d_{\texttt{MO}}(f_i(v_i), f_i(w_i)),$$

where g and f_i denote the `function` in their corresponding `Transformation`.