Differential Privacy Analysis of Data Processing Workflows

Marlon Dumas[1], Luciano García-Bañuelos[1], and Peeter Laud[2]

[1] University of Tartu, Estonia
{marlon.dumas,luciano.garcia}@ut.ee
[2] Cybernetica, Estonia
peeter.laud@cyber.ee

Abstract. Differential privacy is an established paradigm to measure and control private information leakages occurring as a result of disclosures of derivatives of sensitive data sources. The bulk of differential privacy research has focused on designing mechanisms to ensure that the output of a program or query is $\epsilon-$differentially private with respect to its input. In an enterprise environment however, data processing generally occurs in the context of business processes consisting of chains of tasks performed by multiple IT system components, which disclose outputs to multiple parties along the way. Ensuring privacy in this setting requires us to reason in terms of series of disclosures of intermediate and final outputs, derived from multiple data sources. This paper proposes a method to quantify the amount of private information leakage from each sensitive data source vis-a-vis of each party involved in a business process. The method relies on generalized composition rules for sensitivity and differential privacy, which are applicable to chained compositions of tasks, where each task may have multiple inputs and outputs of different types, and such that a differentially private output of a task may be taken as input by other tasks.

1 Introduction

The broad availability of rich consumer data is driving businesses to become increasingly data-driven in their daily operations. In particular, it is becoming common practice for businesses to exploit private data about their current or potential customers to design, sell and deliver services. As a broader set of organizational stakeholders become involved in processing personal customer data – sometimes across organizational boundaries – it becomes increasingly critical to measure and control private information leakages.

Differential privacy [5] has emerged as a promising foundation to quantify and control private information leakages stemming from access to sensitive data sources. The bulk of research in this field has focused on designing mechanisms to ensure that the output of a given program or query is $\epsilon-$differentially private with respect to a collection of input objects, for a given privacy budget ϵ. In other words, the contribution of each object in the input collection to the output is bounded by a term dependent on the privacy budget.

In an enterprise environment however, data processing generally occurs in the context of business processes consisting of complex chains of tasks performed by a range of IT system components and human actors. Ensuring privacy in this setting requires us to reason not only in terms of an individual disclosure of the output of a program or query to one party, but rather in terms of series of disclosures to a range of parties.

This paper addresses the problem of analyzing differential privacy in the context of business processes that involve multiple tasks, such that the output of one task may be used as input by other tasks, and such that intermediate or final outputs are disclosed to multiple parties. The paper addresses this problem in the setting where business processes are codified using graphical models consisting of processing nodes that extract data from potentially sensitive data sources, transform the extracted data, and disclose derivatives thereof using a differential privacy mechanism. Given such a graphical model, the paper outlines a technique to address the following question: *How much information about individual objects in each input data collection does one execution of a process model disclose to each involved party?*

To illustrate this problem, we consider a simplified process to produce a report about combined (data and call) service usage at a telecommunication services provider. This process is depicted in Figure 1 using the standard Business Process Model and Notation (BPMN) [13]. The telco provider is represented by a pool.[3] There is a separate pool below it, corresponding to a contractor hired by the telco to provide business consultancy services. To provide its services, the contractor needs to access weekly "service summary reports" produced by the telco. Inside the telco's pool, there are two roles represented by the lanes labeled "Data Analyst 1" and "Data Analyst 2". The process starts when a new summary report is created (cf. the start event labelled "summary report required"). First, Data Analyst 1 performs a task wherein a set of call records are accessed in order to prepare a call summary table. We assume that this collection of call records (represented by a data collection – rectangle with a folded corner and three vertical stripes) contains sensitive data. Hence, Data Analyst 1 does not get the actual data collection, but only the result of a differentially private query. As a result of this task, a "Call summary table" is produced. Next, an automated task is executed that combines the previous "Call summary table" with another collection of "Data connection records" in order to produce a "Combined report". Again, since collection "Data connection records" contains sensitive data records, the program executing this latter task incorporates a differential privacy mechanism, which ensures that the combined report is ϵ_2- (resp. ϵ_3-) differentially private with respect to "Data connection records" (resp. "Call summary table"). The combined report is then checked by Data Analyst 2, who may modify it. The process ends with a "message event" denoting the fact that the combined report is sent out to the contractor.

[3] A pool in BPMN (represented by a horizontal rectangle) represents an independent organizational entity that communicates with other entities via *message flows*, represented via dashed arrows.

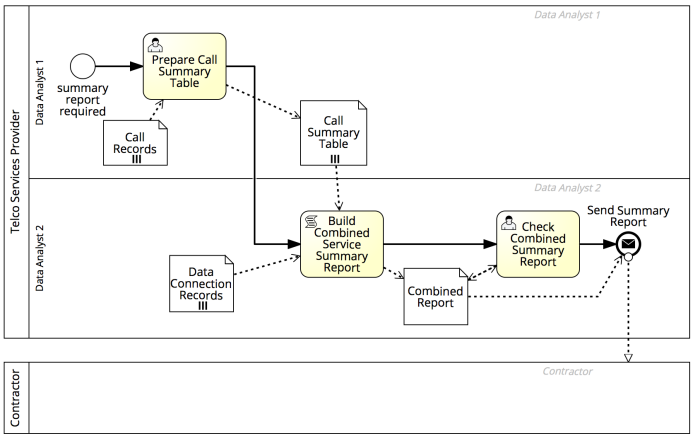

Fig. 1. Model of a report preparation process (in BPMN).

Given a graphical process model annotated with metadata about differentially private data releases, the technique proposed in this paper calculates for each stakeholder in the process (Data Analyst 1, Data Analyst 2 and Contractor) and for each input of the process (here the "Call records" and the "Data connection records"), how much ϵ privacy budget the stakeholder consumes with respect to the data collection in question during one execution of the process. This output can be used by an analyst to fine-tune the process (e.g. by adjusting the ϵ_i privacy budgets) in order to achieve a certain level of privacy vis-a-vis of each stakeholder and each data collection.

The proposed technique relies on a theoretical foundation that provides composition rules to calculate sensitivity and differential privacy of chained compositions of tasks, where these tasks take multiple inputs and produce multiple outputs of different types. The paper outlines an algorithm that applies these composition rules iteratively in order to calculate end-to-end differential privacy for a given process.

The rest of the paper is structured as follows. Section 2 introduces concepts and associated notation used subsequently. Section 3 presents the definitions of differential privacy and sensitivity and associated composition rules. Next, Section 4 introduces a notation for privacy-enhanced process modeling and presents an algorithm for differential privacy analysis of such models. Finally Section 5 discusses related work and Section 6 summarizes the contribution and outlines directions for future work.

2 Notation and preliminaries

We use $\mathbb{R}$ and $\mathbb{N}$ to denote the sets of real and natural numbers, respectively. The sets of non-negative real and extended real numbers are denoted by $\mathbb{R}_+$ and $\mathbb{R}_+^\infty = \mathbb{R}_+ \cup \{\infty\}$. If $a, b \in \mathbb{R}$, then $[a, b]$ denotes the set $\{x \in \mathbb{R} \,|\, a \leq x \leq b\}$.

If X is a set then $\mathcal{D}(X)$ denotes the set of all *countable* probability distributions of X. The elements of X are mappings $\chi : X \to [0,1]$, such that the set $\mathrm{supp}(\chi) = \{x \in X \mid \chi(x) > 0\}$ is (at most) countable.

Given a probability distribution $\psi \in \mathcal{D}(X \times Y)$, we let $\mathsf{proj}_1\psi \in \mathcal{D}(X)$ and $\mathsf{proj}_2\psi \in \mathcal{D}(Y)$ denote its projections to the first and second component, respectively. These are defined by $\mathsf{proj}_1\psi(x) = \sum_{y \in Y} \psi(x,y)$ for all $x \in X$ and similarly for $\mathsf{proj}_2\psi$, where the sum is well-defined due to the support of ψ being countable. For $\chi \in \mathcal{D}(X)$ and $\phi \in \mathcal{D}(Y)$ we let $\chi \otimes \phi \subseteq \mathcal{D}(X \times Y)$ denote the set of all such probability distributions ψ that satisfy $\mathsf{proj}_1\psi = \chi$ and $\mathsf{proj}_2\psi = \phi$.

Let $f : X \to \mathcal{D}(Y)$ and $g : Y \to \mathcal{D}(Z)$. There is an obvious way to "compose" f and g, the result of which we denote with $g \circ_{\mathrm{Kl}} f$ and which is defined by

$$\Pr[(g \circ_{\mathrm{Kl}} f)(x) = z] = \sum_{y \in Y} \Pr[f(x) = y] \cdot \Pr[g(y) = z] \tag{1}$$

for all $x \in X$ and $z \in Z$.

The notion of sensitivity of mappings used in this paper relies on (extended) metric spaces defined as follows.

Definition 1 (Metric space). *A* metric *space is a set X together with a* metric *d_X on it. A mapping $d_X : X \times X \to \mathbb{R}_+$ is a metric if it satisfies the following conditions:*

- *for all $x, y \in X$: $d_X(x,y) = 0$ iff $x = y$;*
- *for all $x, y \in X$, $d_X(x,y) = d_X(y,x)$;*
- *for all $x, y, z \in Z$, $d_X(x,z) \leq d_X(x,y) + d_X(y,z)$.*

An *extended metric* may also take the value ∞. An *extended metric space* is a set X together with an extended metric on it.

3 Differential Privacy

Let $\mathcal{R}$ be the set of possible database records and $X = \mathbb{N}^{\mathcal{R}}$ be the set of databases (i.e. a database is a multiset of records). Let $\mathcal{O}$ be a set of possible outcomes and $\mathcal{M} : X \to \mathcal{O}$ a probabilistic map (an *information release mechanism*). For $r \in \mathcal{R}$ let $x_1 \overset{r}{\sim} x_2$ denote that x_1, x_2 *differ only by* r, i.e. $x_1(r) = x_2(r) \pm 1$ and $x_1(r') = x_2(r')$ for all $r' \in \mathcal{R} \backslash \{r\}$. Two databases $x_1, x_2 \in X$ are *adjacent* if $x_1 \overset{r}{\sim} x_2$ for some $r \in \mathcal{R}$. Let d_X be any (extended) metric on X.

Definition 2 (Differential privacy [5]). *Let $\varepsilon \in \mathbb{R}$. The mechanism $\mathcal{M}$ is* ε-differentially private *if $\Pr[\mathcal{M}(x_1) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(x_2) \in S]$ for all $S \subseteq \mathcal{O}$ and all adjacent databases $x_1, x_2 \in \mathcal{X}$.*

There are a number of ways to make information release mechanisms private, but the most commonly used techniques amount to adding a certain amount of noise to the output of the mechanism. The noise has to be sampled from the correct distribution, in order to obtain the bounds on the ratio of probabilities, as

demanded by Def. 2. The Laplacian distribution has the necessary properties [6]. The required magnitude of the noise depends on the function that is computed by the mechanism. A function that may have very different outputs for databases differing only a little requires more noise to be added than a function that changes only slowly.

Sensitivity is a key tool to reason about the differential privacy of information release mechanisms. It gives upper bounds for the ratio of the change in the value of the function with respect to a change in the argument of the function. For mechanisms that first compute a "useful" function and then add noise to it, the differential privacy of the resulting mechanism is the ratio of the sensitivity of that function and the magnitude of the added noise.

Definition 3 (Sensitivity). *Let X and Y be two metric spaces with distances d_X and d_Y on them. Let $c \in \mathbb{R}_+$. We say that a function $f : X \to Y$ is* c-sensitive*, if for all $x_1, x_2 \in X$, the inequality $d_Y(f(x_1), f(x_2)) \leq c \cdot d_X(x_1, x_2)$ holds.*

Differential privacy itself can also be seen as an instance of sensitivity. Indeed, define the following extended metric d_{dp} over $\mathcal{D}(Y)$:

$$d_{\mathrm{dp}}(\chi, \chi') = \sup_{y \in Y} \left|\ln(\chi(y)/\chi'(y))\right| \ .$$

Then a mechanism $\mathcal{M}$ from X to Y is d_X-private iff it is 1-sensitive with respect to the distances d_X on X and d_{dp} on $\mathcal{D}(Y)$.

The well-known composition theorems of differential privacy are instantiations of more general results on sensitivity of composed mappings. We start with the simplest result for sensitivity.

Proposition 1. *Let $f : X \to Y$ be c-sensitive with respect to the distances d_X on X and d_Y on Y. Let $f' : Y \to Z$ be c'-sensitive with respect to the distances d_Y on Y and d_Z on Z. Then $f' \circ f : X \to Z$ is $c \cdot c'$-sensitive with respect to the distances d_X on X and d_Z on Z.*

Proof. Let $x, x' \in X$. Then $d_Z(f'(f(x)), f'(f(x'))) \leq c' \cdot d_Y(f(x), f(x')) \leq c' \cdot c \cdot d_X(x, x')$.

This proposition can be generalized to multivariate mappings. Let $i \in \{1, \ldots, n\}$. We say that a mapping $f' : Y_1 \times \cdots \times Y_n \to Z$ is c'_i*-sensitive in its i-th argument*, if for all tuples $(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n) \in Y_1 \times \cdots \times Y_{i-1} \times Y_{i+1} \times \cdots \times Y_n$, the univariate mapping $f(y_1, \ldots, y_{i-1}, \cdot, y_{i+1}, \ldots, y_n)$ is c'_i-sensitive.

Proposition 2. *For $i \in \{1, \ldots, n\}$, let $f_i : X \to Y_i$ be c_i-sensitive with respect to the distances d_X on X and d_{Y_i} on Y_i. Let $f' : Y_1 \times \cdots \times Y_n \to Z$ be c'_i-sensitive with respect to the distances d_{Y_i} on Y_i and d_Z on Z (for all $i \in \{1, \ldots, n\}$). Then the mapping $g : X \to Z$, defined by $g(x) = f'(f_1(x), \ldots, f_n(x))$, is $\sum_{i=1}^{n} c_i c'_i$-sensitive with respect to the distances d_X on X and d_Z on Z.*

Proof. Let $x, x' \in X$. Let $z_i = f'(f_1(x), \ldots, f_i(x), f_{i+1}(x'), \ldots, f_n(x'))$. Then $z_0 = g(x')$, $z_n = g(x)$ and by Prop. 1, $d_Z(z_{i-1}, z_i) \leq c_i c_i' \cdot d_X(x, x')$. The claim of the proposition follows from the triangle inequality.

The sequential composition theorem for differential privacy [12, Thm. 3] is really just a special case of Prop. 2. In their setting, there is a dataset $x \in X$ and information release mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$, which are respectively ε_1- and ε_2-differentially private. Let the possible set of outcomes of $\mathcal{M}_i$ be M_i. First $\mathcal{M}_1$ and then $\mathcal{M}_2$ are invoked on x; the exact invocation of $\mathcal{M}_2$ may depend on the result of $\mathcal{M}_1$. Finally, the result of $\mathcal{M}_2$ is published. This result may include the result of $\mathcal{M}_1$, because it affected the invocation of $\mathcal{M}_2$. Such composition of $\mathcal{M}_1$ and $\mathcal{M}_2$ is shown to be $(\varepsilon_1 + \varepsilon_2)$-differentially private.

Prop. 2 applies to this setting in the following manner. We have $\mathcal{M}_1 : X \to \mathcal{D}(M_1)$ and $\mathcal{M}_2 : X \times M_1 \to \mathcal{D}(M_2)$. Let $\overline{\mathcal{M}_2} : X \times \mathcal{D}(M_1) \to \mathcal{D}(M_2)$ be the lifting of $\mathcal{M}_2$ to probability distributions in its second argument:

$$\Pr[\overline{\mathcal{M}_2}(x, \chi) = m_2] = \sum_{m_1 \in M_1} \chi(m_1) \Pr[\mathcal{M}_2(x, m_1) = m_2] \ .$$

Consider the Hamming distance on X (two datasets in X are *adjacent* iff their distance is 1), and the distance d_{dp} on both $\mathcal{D}(M_1)$ and $\mathcal{D}(M_2)$. Let $f_1 \equiv id_X$, $f_2 \equiv \mathcal{M}_1$ and $f' \equiv \overline{\mathcal{M}_2}$. The sensitivity of f_1 is 1, the sensitivity of f_2 is ε_1, and the sensitivities of f' in its first and second argument are ε_2 and 1, respectively. The latter follows the fact that no post-processing of a differentially private query can lower the privacy guarantees it provides. We now apply Prop. 2 and find that the composition of f' with $f_1 \times f_2$ is $(\varepsilon_1 + \varepsilon_2)$-sensitive. This composition corresponds to the invocation of $\mathcal{M}_1$ and $\mathcal{M}_2$ one after another, as described above.

4 Privacy Analysis of Data Processing Workflows

In this section, we introduce a graphical notation for capturing data processing workflows with differential privacy, and we define algorithms to analyze the end-to-end differential privacy of such workflows.

4.1 Data Processing Workflows

In Section 1, we presented a motivating example of a business process using the BPMN notation. While BPMN is a widely used standard, it is also rather complex. It comprises several dozen types of notational elements, covering several flavors of parallel and conditional branching, sequential and parallel repetition, exception handling and transactional constructs among others. For privacy analysis purposes, we propose to reason on a simpler and more abstract graphical notation, herein called *data processing workflow*. This simpler process modeling notation is focused on capturing how data sources taken as input by a business

process are transformed into intermediate and final outputs, each of which is disclosed to one or multiple parties. Below we introduce data processing workflows without considering the notion of "disclosure to a party". The latter notion is added in Section 4.3.

A data processing workflow consists of *data nodes*, *processing nodes* and *data-flow arcs*. A data-flow arc connects a data node to a processing node or vice-versa. A data node without any incoming arc is called a *source data node*. It corresponds to an object or collection of objects that are given as input to the workflow. A data node without any outgoing arc is called an *output node* (i.e. it is data produced by an execution of the workflow). A data node with both incoming and outgoing arcs is called an *intermediate node*.

Figure 2 shows an example of a data processing workflow. Data nodes are represented as rounded rectangles, while data nodes are rectangles with their top-right corner folded over.

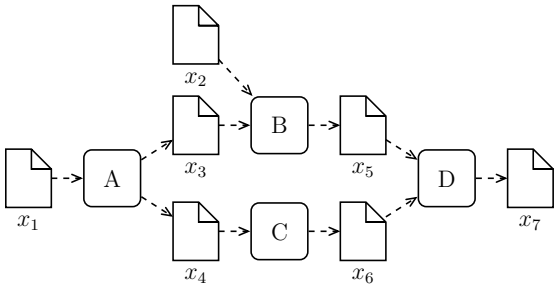

Fig. 2. Example of a data processing workflow

Formally, a *Data Processing Workflow* W is a tuple (D, P, F), where D and P are two finite, disjoint sets, and F is a relation on $(D \times P) \cup (P \times D)$. For convenience, we will refer to $D \cup P$ as the set of nodes N. The elements of D are data nodes and the elements of P are processing nodes, that is, nodes representing computations over some input data.

Given a node $n \in N$, we define $\bullet n = \{m \mid (m, n) \in F\}$ (the predecessors of n) and $n\bullet = \{m \mid (m, n) \in F\}$ (the successors of n). A workflow W is said *well-formed* if it induces an acyclic, weakly connected graph, with the following additional restrictions: every node $d \in D$ has at most one successor and at most one predecessor, i.e. $\mid \bullet d \mid \leq 1$ and $\mid d\bullet \mid \leq 1$, and every node $p \in P$ has at least one predecessor and at least one successor node, i.e. $\mid \bullet p \mid \geq 1$ and $\mid p\bullet \mid \geq 1$. In the following, we consider only well formed workflows.

A privacy-enhanced workflow is a workflow annotated with differential privacy and sensitivity values, which we assume are derived separately via an analysis of a program or query implementing the data processing node (as discussed later in Section 5). Formally, a *Privacy-enhanced workflow* is a tuple $(W, \mathcal{E}, \mathcal{C})$, where $W = (D, P, F)$ is a workflow and $\mathcal{E}$ and $\mathcal{C}$ are mappings of type

$\bigcup_{p \in P} \bullet p \times \{p\} \times p\bullet \to \mathbb{R}_+$, associating a differential privacy and sensitivity value (respectively) to an output produced by a processing node, relative to an input of this processing node.

For example, a privacy-enhanced version of the workflow shown in Figure 2 is shown in Figure 3. In the figure, we use $\epsilon_A[x_1, x_3] = 0.2$ to denote the tuple $(x_1, A, x_3, 0.2) \in \mathcal{E}$, meaning that performing A is ϵ-differential private with $\epsilon = 0.2$, when processing x_1 as input and producing x_3. Similarly, $c_A[x_1, x_3] = 0.4$ is used to denoted the tuple $(x_1, A, x_3, 0.4) \in \mathcal{C}$, which means that A takes as input x_1 and produces x_3 with a sensitivity of 0.4.

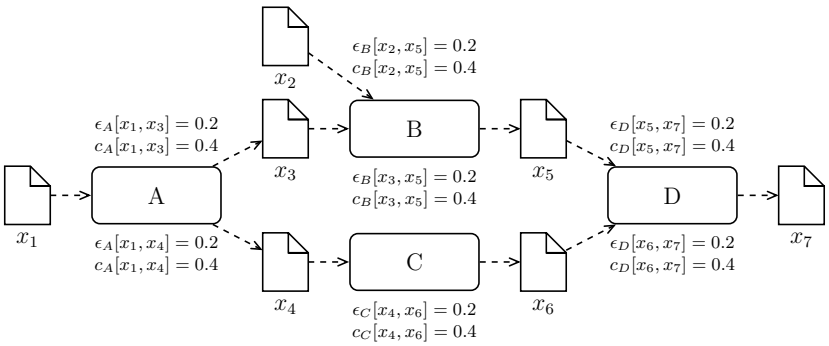

Fig. 3. Example of a privacy-enhanced workflow

A workflow $W = (D, P, F)$ is *interpreted* in the following manner. For each $d \in D$, there is a set X_d and a metric d_d on $\mathcal{D}(X_d)$. For each $p \in P$ and each $d \in p\bullet$, there is a mapping $f_{p \to d} : \prod_{d' \in \bullet p} X_{d'} \to \mathcal{D}(X_d)$, which can be lifted to $\overline{f}_{p \to d} : \prod_{d' \in \bullet p} \mathcal{D}(X_{d'}) \to \mathcal{D}(X_d)$.

Let $S \subseteq D$ be the set of all data nodes d, such that $\bullet d = \emptyset$. For each $d \in D$, an interpretation of W defines a mapping $[\![W]\!]_d : \prod_{d' \in S} \mathcal{D}(X_{d'}) \to \mathcal{D}(X_d)$ as follows. Let $\chi_{d'} \in \mathcal{D}(X_{d'})$ for each $d' \in S$ and let $\mathbf{X}$ be the tuple $(\chi_{d'})_{d' \in S}$. Then

$$[\![W]\!]_d(\mathbf{X}) = \begin{cases} \chi_d, & \text{if } d \in S \\ f_{p \to d}(([\![W]\!]_{d''}(\mathbf{X}))_{d'' \in \bullet p}), & \text{otherwise, where } \{p\} = \bullet d. \end{cases}$$

The mappings $[\![W]\!]_d$ are well-defined due to the acyclicity of W.

The annotations of a privacy-enhanced workflow $(W, \mathcal{E}, \mathcal{C})$ *match* the interpretation of W if for all $p \in P$, $d' \in \bullet p$ and $d \in p\bullet$,

- the sensitivity of $\overline{f}_{p \to d}$ in its argument "d'" is $c_p[d', d]$ with respect to the distances $\mathsf{d}_{d'}$ on $\mathcal{D}(X_{d'})$ and d_d on $\mathcal{D}(X_d)$;
- the sensitivity of $\overline{f}_{p \to d}$ in its argument "d'" is $\epsilon_p[d', d]$ with respect to the distances $\mathsf{d}_{d'}$ on $\mathcal{D}(X_{d'})$ and d_{dp} on $\mathcal{D}(X_d)$.

These requirements talk about metrics over the sets of probability distributions $\mathcal{D}(X_d)$, and sensitivities of lifted mappings in terms of these metrics. In appendix A we discuss, how metrics on sets X_d can be lifted to probability distributions and what should the sensitivities of the original mappings $f_{p\to d}$ be.

4.2 Data Node-Based Analysis of workflows

As stated in Section 1, we are interested in computing upper bounds of the information disclosed when data nodes are accessed by a user playing a given role in the process. In order to do so, we leverage the concepts and definitions of Section 3 to design an algorithm that computes the differential privacy and sensitivity values of every intermediate and output data node in a privacy enhanced workflow, relative to every source data node. Subsequently in Section 4.3, we show how to aggregate the privacy and sensitivity values calculated in this way, in order to compute a bound of the information that a party playing a given role can extract from each source data node, given the data that are disclosed to them during one execution of the workflow.

The proposed algorithm is given in Figure 3. The input of the algorithm is a privacy-enhanced workflow, while the output consists of two matrices, namely d_{dp} and d_c, of size $|S| \times |O|$ where S is the set of source data nodes in the workflow and O is the set of intermediate and output data nodes. A cell in d_{dp} (respectively d_c) gives a differential privacy bound (resp. sensitivity bound) of a given intermediate or output data node o relative to a source data node s. The main idea of the algorithm is to iterate over the processing nodes in the workflow in topological order (which requires that the workflow is well-formed and thus acyclic). At each step, we compute the value of $d_{dp}[s, o]$ and $d_c[s, o]$ for each output o of the current processing node p, using the previously computed values for the input data nodes of p, as well as the formulas for composing sensitivity values given in Propositions 1-3 of Deliverable D1.1 and existing formulas for composition of $\epsilon-$differentially private information release mechanisms.

Example 1. We use the example in Figure 3 to illustrate the algorithm. To this end, we consider the topological order $[A, B, C, D]$ of processing nodes[4].

During the first iteration, in line 1 the algorithm sets p to the processing node A. In line 2, the algorithm iteratively selects a source data node (i.e. $s \in D : |\bullet s| = 0$) and one successor of p such that the latter is reachable from the selected source node. The first iteration of the inner loop then processes the pair $s = x_1$ and $o = x_3$. Since x_1 is a direct predecessor of A the algorithm will perform lines 4-5. As a result, we have that $d_{dp}[x_1, x_3] = \epsilon_A[x_1, x_3] = 0.2$ and $d_c[x_1, x_3] = c_A[x_1, x_3] = 0.4$. The second iteration of the inner loop, in turn, will process the pair $s = x_1$ and $o = x_4$. The latter will result in $d_{dp}[x_1, x_4] = \epsilon_A[x_1, x_4] = 0.2$ and $d_c[x_1, x_4] = c_A[x_1, x_4] = 0.4$. This will complete the first iteration of the outer loop because none of the successors of A is reachable from x_2. The following matrices summarize the outcome of the first iteration:

[4] Note that there exists another topological order of the processing nodes of the example, namely $[A, C, B, D]$. Either one would produce the same output matrices.

Algorithm 1: Differential privacy of a workflow

Data: A well-formed *workflow* (W, S), with $W = (D, P, F)$
Result: The matrices d_{dp} and d_c

1 **foreach** processing node $p \in P$ in topological order **do**
2 **foreach** $s \in D, o \in p\bullet : |\bullet s| = 0 \wedge (s, o) \in F^+$ **do**
3 **if** $s \in \bullet p$ **then**
4 $d_{dp}[s, o] = \epsilon_p[s, o]$
5 $d_c[s, o] = c_p[s, o]$
6 **else**
7 $d_{dp}[s, o] = \sum_{i \in \bullet p : (s,i) \in F^+} \min\left(d_{dp}[s, i],\ d_c[s, i] \cdot \epsilon_p[i, o]\right)$
8 $d_c[s, o] = \sum_{i \in \bullet p : (s,i) \in F^+} \left(d_c[s, i] \cdot c_p[i, o]\right)$
9 **end**
10 **end**
11 **end**
12 **return** d_{dp}, d_c

	x_3	x_4	x_5	x_6	x_7
x_1	$\epsilon_A[x_1, x_3] = 0.2$	$\epsilon_A[x_1, x_4] = 0.2$			
x_2					

d_{dp}

	x_3	x_4	x_5	x_6	x_7
x_1	$c_A[x_1, x_3] = 0.4$	$c_A[x_1, x_4] = 0.4$			
x_2					

d_c

In the second iteration, the algorithm sets p to the processing node B (line 1). The inner loop first computes the values for source node x_1 and the only successor of B, that is x_5. This time, the algorithm executes lines 7-8, because x_1 is not a direct predecessor of B. Note that x_3 is the only direct predecessor of B which is reachable from x_1 and, as a result, there is only one term in the summation of line 7. Therefore, in line 7 we have that $d_{dp}[x_1, x_5] = \min(d_{dp}[x_1, x_3], d_c[x_1, x_3] \cdot \epsilon_B[x_3, x_5]) = \min(0.2, 0.4 \cdot 0.2) = 0.08$ and in line 8 $d_c[x_1, x_5] = d_c[x_1, x_3] \cdot c_B[x_3, x_5] = 0.4 \cdot 0.4 = 0.16$. In the second iteration of the inner loop, the algorithm computes the values associated to the source node x_2 and the only successor of b, that is x_5. Since x_2 is direct predecessor of B, the algorithm sets $d_{dp}[x_2, x_5] = \epsilon_B[x_2, x_5] = 0.2$ and $d_c[x_2, x_5] = c_B[x_2, x_5] = 0.4$.

The third iteration selects $p = C$ and proceeds in a similar way as for the second iteration. The following matrices summarize the values computed at the end of this iteration.

	x_3	x_4	x_5	x_6	x_7
x_1	0.2	0.2	0.08	0.08	
x_2			0.2		

d_{dp}

	x_3	x_4	x_5	x_6	x_7
x_1	0.4	0.4	0.16	0.16	
x_2			0.4		

d_c

In the final iteration, the algorithm computes the values by selecting p to be the processing node D. In the inner loop, the algorithm will first select the source node x_1. Note that D has x_7 as its only successor. However, there are two

direct predecessors of D, namely x_5 and x_6. Therefore the computation of d_{dp} involves the summation of the values that come from x_5 and x_6. Thus, we have that:

$$\begin{aligned} d_{dp}[x_1, x_7] &= \min\left(d_{dp}[x_1, x_5], d_c[x_1, x_5] \cdot \epsilon_D[x_5, x_7]\right) + \\ &\quad \min\left(d_{dp}[x_1, x_6], d_c[x_1, x_6] \cdot \epsilon_D[x_6, x_7]\right) \\ &= \min\left(0.08, 0.16 \cdot 0.2\right) + \min\left(0.08, 0.16 \cdot 0.2\right) \\ &= 0.064 \end{aligned}$$

and

$$\begin{aligned} d_c[x_1, x_7] &= \left(d_c[x_1, x_5] \cdot c_D[x_5, x_7]\right) + \left(d_c[x_1, x_6] \cdot c_D[x_6, x_7]\right) \\ &= (0.16 \cdot 0.4) + (0.16 \cdot 0.4) \\ &= 0.128 \end{aligned}$$

In the final iteration of the inner loop, the algorithm computes the values for $s = x_2$ and $o = x_5$. In this case however, there is only one term in the summation. Therefore, $d_{dp}[x_2, x_7] = \min\left(d_{dp}[x_2, x_5], d_c[x_2, x_5] \cdot \epsilon_D[x_5, x_7]\right) = \min\left(0.2, 0.4 \cdot 0.2\right) = 0.08$. Finally, $d_c[x_2, x_7] = d_c[x_2, x_5] \cdot c_D[x_5, x_7] = 0.4 \cdot 0.4 = 0.16$.

The following matrices summarize the outcome of the algorithm.

	x_3	x_4	x_5	x_6	x_7
x_1	0.2	0.2	0.08	0.08	0.064
x_2			0.2		0.16

d_{dp}

	x_3	x_4	x_5	x_6	x_7
x_1	0.4	0.4	0.16	0.16	0.128
x_2			0.4		0.08

d_c

The correctness of Algorithm 1 is established by the following theorem.

Theorem 1. *Let $(W, \mathcal{E}, \mathcal{C})$ be a privacy-enhanced workflow with $W = (D, P, F)$. Let W have an interpretation that matches the annotations $\mathcal{E}$ and $\mathcal{C}$. Let $x \in S$ and $y \in O$. Let the matrices d_{dp} and d_c be computed by Alg. 1 from $(W, \mathcal{E}, \mathcal{C})$. Then $[\![W]\!]_y$ is $d_{dp}[x, y]$-differentially private and $d_c[x, y]$-sensitive in its argument "x" according to the distances d_x on $\mathcal{D}(X_x)$ and d_y on $\mathcal{D}(X_y)$.*

Proof. The theorem is proved by induction over the data nodes of W, taken in topological order. First we amend d_{dp} and d_c with columns corresponding to data nodes in S, defining $d_c(x, x) = 1$ and $d_{dp}(x, x) = \infty$ for all $x \in S$, as well as $d_c(x, y) = d_{dp}(x, y) = 0$ for all $x, y \in S$ with $x \neq y$. We now proceed with the induction.

Base case: $y \in S$. Then $[\![W]\!]_y$ takes the component "y" from its tuple of arguments. The values $d_{dp}(x, y)$ and $d_c(x, y)$ describe the sensitivity and differential privacy of the protection.

Induction step: let $\{p\} = \bullet y$ and let the differential privacy and sensitivity claims hold for all $[\![W]\!]_{y'}$, where $y' \in \bullet p$. We note that $f_{p \to y}$ is 1-sensitive for all its inputs, if the distance on both the input and the output is d_{dp}. The reason for this is, that no post-processing can degrade the privacy level of a differentially private mapping.

Let $C_{y'}$ be the sensitivity of $[\![W]\!]_{y'}$ in its argument "x". By induction hypothesis, $C_{y'} \leq d_c[x, y']$. Prop. 2 now gives us that the sensitivity of $[\![W]\!]_y$ in its argument "x" is at most $\sum_{y' \in \bullet p} C_{y'} \cdot c_p[y', y] \leq \sum_{y' \in \bullet p} d_c[x, y'] \cdot c_p[y', y] = d_c[x, y]$ by Alg. 1.

Similarly, let $E_{y'}$ be differential privacy level of $[\![W]\!]_{y'}$ in its argument "x"; by induction hypothesis, $E_{y'} \leq d_{dp}[x, y']$. On each $\mathcal{D}(X_{y'})$, we may consider either the distance $\mathsf{d}_{y'}$ or d_{dp}. The sensitivity of $[\![W]\!]_{y'}$ (in argument "x") is $\mathbf{c}^1_{y'} = C_{y'}$ according to $\mathsf{d}_{y'}$ or $\mathbf{c}^0_{y'} = E_{y'}$ according to d_{dp}. The sensitivity of $f_{p \to y}$ in argument "y'" according to the distance $\mathsf{d}_{y'}$ [resp. d_{dp}] on $\mathcal{D}(X_{y'})$ and the distance d_{dp} on $\mathcal{D}(X_y)$ is $\boldsymbol{\epsilon}^1_{y'} = \epsilon_p[y', y]$ [resp $\boldsymbol{\epsilon}^0_{y'} = 1$]. Let $b[y'] \in \{0, 1\}$ for each $y' \in \bullet p$. According to Prop. 2, the differential privacy of $[\![W]\!]_y$ is $\sum_{y' \in \bullet p} \mathbf{c}^{b[y']}_{y'} \cdot \boldsymbol{\epsilon}^{b[y']}_{y'}$, obtained by considering the distance $\mathsf{d}_{y'}$ (if $b[y'] = 1$) or d_{dp} (if $b[y'] = 0$) on $\mathcal{D}(X_{y'})$. This bound for differential privacy holds for any choice of bits $b[y']$. Hence the differential privacy of $[\![W]\!]_y$ is

$$\min_{\forall y' \in \bullet p : b[y'] \in \{0,1\}} \sum_{y' \in \bullet p} \mathbf{c}^{b[y']}_{y'} \cdot \boldsymbol{\epsilon}^{b[y']}_{y'} = \sum_{y' \in \bullet p} \min(\mathbf{c}^0_{y'} \cdot \boldsymbol{\epsilon}^0_{y'}, \mathbf{c}^1_{y'} \cdot \boldsymbol{\epsilon}^1_{y'}) =$$

$$\sum_{y' \in \bullet p} \min(E_{y'} \cdot 1, C_{y'} \cdot c_p[y', y]) \leq \sum_{y' \in \bullet p} \min(d_{dp}[x, y'], d_c[x, y'] \cdot c_p[y', y]) = d_{dp}[x, y]$$

by Alg. 1.

4.3 Role-based privacy analysis of workflows

So far, we have considered workflows without a notion of *parties* to whom data is disclosed. To capture this latter aspect, we extend the notion of privacy-enhanced workflow with a disclosure relation $Disc \subseteq D \times R$, such that $Disc(n, r)$ denotes the fact that data node n is disclosed to role r. We assume here a classical role-based access control model, entailing that all users who play role r are able to access all data nodes disclosed to role r.

Given the matrices d_{dp} and d_c computed from a privacy-enhanced workflow W and given the relation $Disc$ capturing the disclosure of data nodes to roles, we can now compute a differential privacy bound $\epsilon_r(s)$ of the information that a given role r can extract from a given source data node s – i.e. how much a party playing a given role can learn about individual records of a given input s of W:

$$\epsilon_r(s) = \sum_{(n,r) \in Disc \,:\, (s,n) \in F^+} d_{dp}[s, n] \tag{2}$$

Example 2. Given the matrices computed in the previous example:

	x_3	x_4	x_5	x_6	x_7
x_1	0.2	0.2	0.08	0.08	0.064
x_2			0.2		0.16

d_{dp}

	x_3	x_4	x_5	x_6	x_7
x_1	0.4	0.4	0.16	0.16	0.128
x_2			0.4		0.08

d_c

we can compute the differential privacy guarantee with respect to data node x_1 that can be made for a party playing a role r that has access to both data nodes x_5 and x_6 in the workflow shown in Figure 3:

$$\begin{aligned}\epsilon_r(x_1) &= d_{dp}[x_1, x_5] + d_{dp}[x_1, x_6] \\ &= 0.08 + 0.08 = 0.16\end{aligned}$$

In Equation 2, we sum up the ϵ values calculated for each intermediate/output data node that is disclosed to role r. This is a worst-case bound, applicable in the case of "sequential composition" of differentially private release mechanisms. The underpinning assumption is that if a role has access to two data nodes n_1 and n_2 produced from a given source data node s via two different paths in the workflow, these two paths use the same or overlapping parts of data source s. If this is not the case, meaning that n_1 and n_2 derive from independent parts of s, a tighter bound may be applied – in the best case $\max(d_{dp}[s, n_1], d_{dp}[s, n_2])$ instead of $d_{dp}[s, n_1] + d_{dp}[s, n_2]$, based on existing results for so-called "parallel composition" of differentially private release mechanisms. Hence, if we annotated a privacy-enhanced workflow with additional metadata capturing independence relations between multiple accesses to the same data source, we could refine the calculation of the differential privacy budgets. Investigating this refinement is a direction for future work.

5 Related work

Differential privacy has been widely studied in the context of program analysis, using e.g. types [10] or theorem proving [2]. These techniques allow one to reason about the differential privacy of the output of a program relative to its input. In a similar vein, techniques have been proposed to analyze sensitivity and differential privacy for database queries expressed in SQL-like languages (e.g. PINQ) [12]. In this latter work, the aim is to ensure that the output of a given query is differentially private with respect to the input tables, for a given privacy budget. Again, these techniques are geared at analyzing that the output of a processing node is differentially private with respect to its input. In this respect, these proposals are complementary to ours: They can be used to analyze the sensitivity and differential privacy of each data processing node in a workflow.

Perhaps the closest work to ours is Featherweight PINQ [7]. This latter work defines a calculus that can be used to determine the sensitivity of parallel and sequential compositions of queries defined in a workflow-like notation. However, it does not provide a framework that combines sensitivity and differential privacy under the same roof as we do in our proposal.

Our work is also related to *d-privacy* [14]: a generalization of differential privacy that turns any metric on the set of possible datasets into a composable privacy metric, of which differentially privacy is a special case. In this paper, we build upon these and related ideas in [4, 8] in order to define new composition

rules for processing nodes with differentially-private release mechanism, specifically rules that combine sensitivity and differential privacy in a way that allows us to calculate differentially private bounds when a differentially-private output of a node is fed as input to another differentially-private node.

Previous work on privacy analysis of business processes [1, 9] relies on Petri net reachability analysis and model checking to detect data objects that are declassified to unauthorized parties either in full or in part. These approaches adopt a multi-level security model, wherein the objects and subjects of the system are divided in security levels. The goal of these techniques is to identify cases where information from an object of a higher security level is copied to an object with lower security level. However such techniques are boolean: they detect potential leakages but fail to quantify them, which is the goal of the present paper.

The workflow notation employed in this paper is conceptually similar to graphical workflow notations used in data warehousing [11] – where they are referred to as Extract-Transform-Load (ETL) workflows – and also bears resemblances with data analytics workflow notations such as the one embodied in the popular KNIME toolset [3]. The results presented in this paper can potentially be applied to analyze workflows in these related notations.

6 Conclusion

To summarize, the main contributions of this paper are:

1. Theoretical results on sensitivity of composed mappings that generalize well-known composition theorems of differential privacy and allow us to calculate differential privacy bounds in the case where the differentially-private output of a function is used as input to another differentially-private function.
2. A notion of privacy-enhanced workflow where tasks (processing nodes) transform input objects into output objects using differentially private mechanisms and each intermediate or output object is disclosed to one or more roles.
3. An algorithm that given a data processing workflow and given the sensitivity and differential privacy leakage of each processing node in the workflow, estimates the differential privacy leakage generated by the disclosure of data to each role involved in the workflow.

The proposed analysis technique has been implemented in a tool called Pleak.[5] Pleak allows one to: (i) model a process using the standard Business Process Model and Notation (BPMN); (ii) annotate the elements of the process model with sensitivity and differential privacy metadata; and (iii) obtain a table stating the differential privacy budget consumed by each role (lane or pool) in the process relative to each input data collection. Internally, the tool extracts a data processing workflow from the BPMN model, and applies the technique presented above. At present, only a subset of BPMN is supported, comprising

[5] The tool is available at `http://pleak.io/` for research purposes.

tasks, sequence flows, parallel gateways, data objects, lanes and pools. Also, the input process models are assumed to be acyclic. These restrictions are meant to ensure the BPMN model can be transformed into a data processing workflow.

In future, we will extend the notion of data processing workflow in order to lift some of the restrictions imposed so far, particularly the restriction that data processing workflows do not contain conditional branching nor cycles. To this end, we need to extend the theoretical foundation to reason about conditional branches in a differentially private computation. Also, as stated in Section 4.3, we plan to enhance the workflow notation to capture independence relations between multiple accesses to the same data source, so as to calculate tighter bounds for differential privacy budgets when multiple computations access independent parts of the same data collection (e.g. distinct sets of attributes).

Acknowledgments. This work is funded by DARPA's "Brandeis" programme.

References

1. Rafael Accorsi, Andreas Lehmann, and Niels Lohmann. Information leak detection in business process models: Theory, application, and tool support. *Inf. Syst.*, 47:244–257, 2015.
2. Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. Probabilistic relational reasoning for differential privacy. *ACM Trans. Program. Lang. Syst.*, 35(3):9, 2013.
3. Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. *SIGKDD Explorations*, 11(1):26–31, November 2009.
4. Konstantinos Chatzikokolakis, Miguel E. Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. Broadening the scope of differential privacy using metrics. In Emiliano De Cristofaro and Matthew Wright, editors, *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*, volume 7981 of *Lecture Notes in Computer Science*, pages 82–102. Springer, 2013.
5. Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.
6. Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.
7. Hamid Ebadi and David Sands. Featherweight PINQ. *CoRR*, abs/1505.02642, 2015.
8. Ehab ElSalamouny, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Generalized differential privacy: Regions of priors that admit robust optimal mechanisms. In Franck van Breugel, Elham Kashefi, Catuscia Palamidessi, and Jan

Rutten, editors, *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*, volume 8464 of *Lecture Notes in Computer Science*, pages 292–318. Springer, 2014.

9. Simone Frau, Roberto Gorrieri, and Carlo Ferigato. Petri net security checker: Structural non-interference at work. In *5th International Workshop on Formal Aspects in Security and Trust (FAST), October 9-10, Malaga, Spain*, pages 210–225. Springer, 2008.
10. Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. Linear dependent types for differential privacy. In Roberto Giacobazzi and Radhia Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 357–370. ACM, 2013.
11. Ralph Kimball, Laura Reeves, Warren Thornthwaite, Margy Ross, and Warren Thornwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1998.
12. Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 19–30. ACM, 2009.
13. Object Management Group. Business Process Model and Notation (BPMN) Version 2.0, 2011.
14. Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In Paul Hudak and Stephanie Weirich, editors, *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010*, pages 157–168. ACM, 2010.

A Lifting distances to probability distributions

To interpret a privacy-enhanced DP-workflow $(W, \mathcal{E}, \mathcal{C})$ (where $W = (D, P, F)$), we have to give metrics d_d on $\mathcal{D}(X_d)$ for each $d \in D$. Moreover, for the interpretation to be matched by the annotations, the mappings $\overline{f}_{p\to d}$ between these probability distributions must have the sensitivities given by $\mathcal{E}$ and $\mathcal{C}$. It may be more natural to assume that the interpretation gives us metrics on X_d, not on $\mathcal{D}(X_d)$. It is also more natural to require the mappings $f_{p\to d}$ to have a certain sensitivity.

We thus define that a *pre-interpretation* consists of sets X_d for each $d \in D$ together with a metric $\mathsf{d}_d^\flat$ on it, as well as the mappings $f_{p\to d}$ for each $p \in P$ and $d \in p\bullet$. We have to specify what kind of interpretation it generates, and when to the annotations $\mathcal{E}, \mathcal{C}$ match the pre-interpretation. The key for this is to specify the metric d_d on $\mathcal{D}(X_d)$.

Let X be a set and d_X a metric on it. It turns out that the following definition of a metric $d_X^\#$ on $\mathcal{D}(X)$ is a suitable one. Let $\chi, \chi' \in \mathcal{D}(X)$. Then

$$d_X^\#(\chi, \chi') = \inf_{\psi \in \chi \otimes \chi'} \sup_{(x,x') \in \mathrm{supp}(\psi)} d_X(x, x') \ . \tag{3}$$

The proposed metric $d_X^{\#}$ can be seen as a kind of "worst-case" earth mover's distance (or Wasserstein metric). In the "usual" earth mover's distance, one would take the average over ψ, not the supremum over $\mathrm{supp}(\psi)$.

The suitability of the construction (3) is given by the following two propositions. Note that the first of them would not hold for the "usual" earth mover's distance.

Proposition 3. *Let $f : X \to \mathcal{D}(Y)$ be ε-sensitive according to the distance d_X on X and distance d_{dp} on $\mathcal{D}(Y)$. Then the lifting $\overline{f} : \mathcal{D}(X) \to \mathcal{D}(Y)$ is ε-sensitive according to the distance $d_X^{\#}$ on $\mathcal{D}(X)$ and d_{dp} on $\mathcal{D}(Y)$.*

Proof. Let $\chi, \chi' \in \mathcal{D}(X)$, $\psi \in \chi \otimes \chi'$ and $y \in Y$. Then

$$\begin{aligned}
\Pr[\overline{f}(\chi) = y] = \sum_{x \in X} \chi(x) \cdot \Pr[f(x) = y] = \sum_{x,x' \in X} \psi(x,x') \cdot \Pr[f(x) = y] \leq \\
\sum_{x,x' \in X} \psi(x,x') \cdot e^{\varepsilon \cdot d_X(x,x')} \Pr[f(x') = y] \leq \\
\sum_{x,x' \in X} \psi(x,x') \cdot e^{\sup_{x \in \mathrm{supp}(\psi(\cdot,x'))} \varepsilon \cdot d_X(x,x')} \Pr[f(x') = y] = \\
\sum_{x' \in X} \chi'(x') \cdot e^{\sup_{x \in \mathrm{supp}(\psi(\cdot,x'))} \varepsilon \cdot d_X(x,x')} \Pr[f(x') = y] \leq \\
e^{\sup_{x,x' \in \mathrm{supp}(\psi)} \varepsilon \cdot d_X(x,x')} \cdot \sum_{x' \in X} \chi'(x') \cdot \Pr[f(x') = y] = \\
e^{\sup_{x,x' \in \mathrm{supp}(\psi)} \varepsilon \cdot d_X(x,x')} \cdot \Pr[\overline{f}(\chi') = y],
\end{aligned}$$

where $\mathrm{supp}(\psi(\cdot, x'))$ denotes the set of all $x \in X$, such that $\psi(x,x') > 0$. We obtain

$$\begin{aligned}
d_{\mathrm{dp}}(\overline{f}(\chi), \overline{f}(\chi')) = \sup_{y \in Y} \left| \ln \frac{\Pr[\overline{f}(\chi') = y]}{\Pr[\overline{f}(\chi) = y]} \right| \leq \\
\inf_{\psi \in \chi \otimes \chi'} \sup_{x,x' \in \mathrm{supp}(\psi)} \varepsilon \cdot d_X(x,x') = \varepsilon \cdot d_X^{\#}(\chi, \chi') \ .
\end{aligned}$$

Proposition 4. *Let $f : X \to \mathcal{D}(Y)$ be c-sensitive according to the distance d_X on X and distance $d_Y^{\#}$ on $\mathcal{D}(Y)$, where $d_Y^{\#}$ is constructed from some distance d_Y on Y according to (3). Then $\overline{f} : \mathcal{D}(X) \to \mathcal{D}(Y)$ is c-sensitive according to the distance $d_X^{\#}$ on $\mathcal{D}(X)$ and $d_Y^{\#}$ on $\mathcal{D}(Y)$.*

Proof. Let $\chi, \chi' \in \mathcal{D}(X)$. Define $\mathbf{F}$ as the following set of mappings of type $X \times X \to \mathcal{D}(Y \times Y)$:

$$\mathbf{F} = \{\xi \,|\, \forall x, x' \in X : \xi(x,x') \in f(x) \otimes f(x')\} \ .$$

Also consider the set $\Phi \subseteq \mathcal{D}(Y \times Y)$, defined as follows:

$$\Phi = \{ \sum_{x,x' \in X} \psi(x,x') \cdot \xi(x,x') \,|\, \psi \in \chi \otimes \chi', \xi \in \mathbf{F}\} \ .$$

In the definition of Φ, we take the averages over $\xi(x,x')$ with the weights given by $\psi(x,x')$. We have $\Phi \subseteq \overline{f}(\chi) \otimes \overline{f}(\chi')$ because the first [resp. second] projection of any element of Φ is $\overline{f}(\chi)$ [resp. $\overline{f}(\chi')$]. We now have

$$d_Y^{\#}(\overline{f}(\chi),\overline{f}(\chi')) = \inf_{\phi \in \overline{f}(\chi)\otimes\overline{f}(\chi')} \sup_{(y,y')\in\mathrm{supp}(\phi)} d_Y(y,y') \leq$$

$$\inf_{\phi\in\Phi} \sup_{(y,y')\in\mathrm{supp}(\phi)} d_Y(y,y') = \inf_{\psi\in\chi\otimes\chi'} \inf_{\xi\in\mathbf{F}} \sup_{(x,x')\in\mathrm{supp}(\psi)} \sup_{(y,y')\in\mathrm{supp}(\xi(x,x'))} d_Y(y,y') =$$

$$\inf_{\psi\in\chi\otimes\chi'} \sup_{(x,x')\in\mathrm{supp}(\psi)} \inf_{\phi\in f(x)\otimes f(x')} \sup_{(y,y')\in\mathrm{supp}(\phi)} d_Y(y,y') =$$

$$\inf_{\psi\in\chi\otimes\chi'} \sup_{(x,x')\in\mathrm{supp}(\psi)} d_Y^{\#}(f(x),f(x')) \leq \inf_{\psi\in\chi\otimes\chi'} \sup_{(x,x')\in\mathrm{supp}(\psi)} c{\cdot}d_X(x,x') = c{\cdot}d_X^{\#}(\chi,\chi')$$

These two propositions tells us how to turn a pre-interpretation of a privacy-enhanced DP-workflow into an interpretation. We define $\mathsf{d}_d = (\mathsf{d}_d^\flat)^\#$ for each $d \in D$. The annotations $\mathcal{E}, \mathcal{C}$ match the pre-interpretation if for all $p \in P$, $d' \in {\bullet}p$ and $d \in p{\bullet}$:

- the sensitivity of $f_{p\to d}$ in its argument "d'" is $c_p[d',d]$ with respect to the distances $\mathsf{d}_{d'}^\flat$ on $X_{d'}$ and d_d on $\mathcal{D}(X_d)$;
- the sensitivity of $f_{p\to d}$ in its argument "d'" is $\epsilon_p[d',d]$ with respect to the distances $\mathsf{d}_{d'}^\flat$ on $X_{d'}$ and d_{dp} on $\mathcal{D}(X_d)$.

In this way, the corresponding interpretation is also matched by the annotations.