Combining Differential Privacy and Mutual Information for Analyzing Leakages in Workflows

Martin Pettai(✉) and Peeter Laud

Cybernetica AS, Tartu, Estonia
{martin.pettai,peeter.laud}@cyber.ee

Abstract. *Workflows* are a notation for business processes, focusing on tasks and data flows between them. We have designed and implemented a method for analyzing leakages in workflows by combining differential privacy and mutual information. The input of the method is a description of leakages for each workflow component, using either differential-privacy- or mutual-information-based quantification (whichever is known for the component). The differential-privacy-based bounds are combined using the triangle inequality and are then converted to mutual-information-based bounds. Then the bounds for the components are combined using a maximum-flow algorithm. The output of the method is a mutual-information-based quantification of leakages of the whole workflow.

1 Introduction

As the businesses' capabilities of collecting and analysing data increase, so do the privacy issues around such collection and processing. Existing research on finding trade-offs between the privacy of individuals and the utility of collected data has concentrated on constructing or analysing mechanisms for making the outputs of data analysis methods or programs in general privacy-preserving for their inputs, while providing as much accuracy as possible. In enterprise environments, however, processes are compositions of complex tasks performed by humans and computers. To analyze them, we need methods to compose the guarantees given by individual tasks, and reason about a series of information releases to different parties, or the applications of privacy enhancing technologies.

Different privacy-enhancing technologies are best characterized using different measures, and these measures may behave differently and provide different guarantees when tasks are composed. In this paper, we consider the analysis of workflows, i.e. a sequential/parallel composition of tasks, where the outputs of one task may serve as inputs of another. A workflow conveniently captures the data flows in a business process. The inputs to the workflow are privacy-sensitive pieces or collections of data, while the outputs are disclosed to some entity. We want to measure the amount of information flowing from the inputs of the workflow to the outputs, given such measures for single tasks. The information flows of tasks are characterized either in terms of sensitivity of the function

© Springer-Verlag GmbH Germany 2017
M. Maffei and M. Ryan (Eds.): POST 2017, LNCS 10204, pp. 298–319, 2017.
DOI: 10.1007/978-3-662-54455-6_14

computed by the task, or in terms of a privacy measure, which may be either differential privacy, or mutual information between the inputs and outputs of the task. Depending on the order of application of the rules of composition, we may obtain results of very different quality.

As an example, consider a theoretical early warning system (EWS) to locate major incidents in a large city. An incident is deemed to have taken place in a point that most people are moving away from. To find such points, the EWS collects location and movement information from the mobile phones in the city. To enhance the privacy of such collection, noise is added. From the noised data, we can find the epicenter that most phones are moving away from, and report its location with the precision of a city block to rapid responders. At the same time, the law enforcement is interested in determining the outliers among the phones close to the epicenter, given e.g. by their recent call logs, and found using techniques of secure computation [13]. Here the effect of adding noise is best described by differential privacy, while the projections of the location to a city block, and the call log to a classifier are better characterized through Shannon entropy and related notions.

Our analysis receives the descriptions of tasks with respect to the information flows from their inputs to their outputs, as well as the graph describing how the tasks have been composed. The information flows may be characterized through various means named above. The analysis will derive mutual information based information flow constraints for all tasks, and then apply an algorithm based on maximum flow to derive an upper bound on the global leakage. The correctness proof of the proposed maximum flow based algorithm (which by itself is not too complex) is one of the main contributions of this paper.

We begin by describing related work in Sect. 2. In Sect. 3, we define the workflows that can be analyzed using our method. In Sects. 4 and 5, we describe and mutual information and differential privacy, respectively, and study how they are used to quantify leakages in the workflow and its components. In Sect. 6, we list and describe the different kinds of information flow descriptions about tasks handled by our analyser. These descriptions are used by the analyser that is given in Sect. 7 together with its correctness proof. In Sect. 8 we show that the analyser is also in some sense complete — for a given set of information flow descriptions, there exists a workflow matching them, such that the adversary can actually leak as many bits from the source to the target wires as are reported by the analysis. In Sect. 9 we give some examples of component types that can be expressed in our system. In Sect. 10, we describe our implementation of the method. In Sect. 11, we give an example of using the method. Finally, we conclude in Sect. 12.

2 Related Work

Differential privacy [9] (DP) has emerged as a popular metric for privacy preservation in computational mechanisms. There exist a fair number of (semi-)automatic approaches to determine or certify the DP level of a task from its language-based description, including typing [12,17] or automated reasoning [4]. DP has served as the basis of privacy-preserving querying systems [16], where the privacy level of a query is automatically computed [10]. These approaches are not directly applicable to the analysis of workflows, because here several privacy-enhancing tasks may be sequentially composed.

Conditional entropy and mutual information (channel capacity) have also been used as privacy metrics; automated ways to determine or approximate it include static program analysis [6] and statistical sampling [5]. Instead of Shannon entropy, min-entropy has been argued to be more suitable to characterize the resilience of a system against an adversary trying to guess a secret [18]. The automated approaches based on finding the kernel of the leakage function and computing the sizes of its equivalence classes [2], as well as measuring the bit-width of the paths of flow of information through a program [15] can be seen as instantiations of this idea. In [15], the measurement is performed by finding the maximal flow in a certain network, where the arcs correspond to values computed in the program, with their bit-widths being the capacities. This is somewhat similar to Algorithm 1 in this paper, but the capacities correspond to different quantities. Shannon entropy has found its use to describe the total throughput of a communication network with noisy channels [3]. In their setting, the finding of the optimal communication rate also reduces to the computation of maximum flow in a certain network, but their setting is more restricted than ours (e.g. secret sharing cannot be expressed), with greater independence required between various messages.

Recently, a DP analysis for workflows has been proposed [8], composing the sensitivity and DP of individual tasks to the DP of entire workflow. Our approach extends it with the possibility to characterize the privacy preservation in tasks using mutual information and allows converting from differential privacy to mutual information. Also, a leakage description in our approach connects a subset of inputs with a subset of outputs of a task, whereas in [8], the link is between a single input and a single output. Earlier works on privacy analysis of business processes [11] are qualitative in nature.

3 Workflows

A *workflow* consists of information processing *components*, composed sequentially and/or in parallel. The components are connected by *wires*.

Let *Ports* be a fixed infinite set, the elements of which are called *ports*. For each $p \in \mathit{Ports}$ let $\mathrm{V}(p)$ be the set of values that can be input or output through port p. For a set X, et $\mathcal{D}(X)$ denote the set of probability distributions over X.

Definition 1. *A* component *is a tuple* $M = (\mathsf{ip}_M, \mathsf{op}_M, f_M)$*, where* $\mathsf{ip}_M, \mathsf{op}_M \subset \mathit{Ports}$ *are finite,* $\mathsf{ip}_M \cap \mathsf{op}_M = \emptyset$*, and* $f_M : \prod_{p\in\mathsf{ip}_M} \mathrm{V}(p) \to \mathcal{D}(\prod_{p\in\mathsf{op}_M} \mathrm{V}(p))$*.*

Definition 2. *A* workflow *is a tuple* $WF = (\mathcal{M}, \mathcal{W}, \mathsf{s}, \mathsf{t})$*, where* $\mathcal{M}$ *is a finite set of components,* $\mathcal{W}$ *is the finite set of* wires*,* $\mathsf{s} : \sum_{M\in\mathcal{M}} \mathsf{op}_M \to \mathcal{W}$ *and* $\mathsf{t} : \sum_{M\in\mathcal{M}} \mathsf{ip}_M \to \mathcal{W}$*, satisfying the following constraints:*

- *The mapping* s *is injective.*
- *For any two ports* p_1, p_2 *of the components of WF, if* $\mathsf{s}(p_1) = \mathsf{t}(p_2) = w$ *or* $\mathsf{t}(p_1) = \mathsf{t}(p_2) = w$*, then* $\mathrm{V}(p_1) = \mathrm{V}(p_2)$*. We denote this set by* $\mathrm{V}(w)$*.*
- *There are no cycles in the directed graph having the tasks in* $\mathcal{M}$ *as vertices, where an arc from* M_1 *to* M_2 *exists iff there exist* $p_1 \in \mathsf{op}_{M_1}$ *and* $p_2 \in \mathsf{ip}_{M_2}$*, such that* $\mathsf{s}(p_1) = \mathsf{t}(p_2)$*.*

We introduce the following workflow-related notions:

- The *inputs* or *input wires* of a component M are the wires in the set $\mathcal{I}_M = \mathsf{t}(\mathsf{ip}_M)$. Similarly, the *outputs* of M are the wires in $\mathcal{O}_M = \mathsf{s}(\mathsf{op}_M)$.
- The *listeners* of a wire w are the components M satisfying $w \in \mathcal{I}_M$.
- A *path* in the workflow is an alternating list of wires and components, each wire followed by one of its listeners and each component by one of its output wires.
- A wire w is a global input of the workflow, if $\mathsf{s}(p) \neq w$ for all output ports of all components in the workflow. Denote the set of all global inputs by $\mathcal{G}$.

Definition 3. *Let WF* $= (\mathcal{M}, \mathcal{W}, \mathsf{s}, \mathsf{t})$ *be a workflow with global inputs* $\mathcal{G}$*. Let InpDist* $\in \mathcal{D}(\mathcal{G} \to \mathbf{V})$*. The* run *of WF starting from InpDist is a random variable of type* $\mathcal{W} \to \mathbf{V}$*, sampled as follows:*

- *The values for all* $w \in \mathcal{G}$ *are sampled from the distribution InpDist;*
- *Each component* $\mathcal{M}$ *for which all of its input wires are already mapped to values, applies* f_M *to the tuple of values at its input ports, probabilistically producing a tuple of values for its output ports. These values are added to the mapping for the output wires of* M*.*
- *The previous item is repeated until all wires are mapped (this terminates because there are no cycles in the workflow).*

4 Information Flow

Now suppose that a subset S of the global inputs contains sensitive information and a subset T of all wires is eavesdropped by an adversary. We would like to estimate how much sensitive information the adversary can learn during each run of the workflow, i.e. how much information is leaked from S to T. One way to quantify this leakage is using mutual information.

Definition 4. *Let* X*,* Y *and* Z *be discrete random variables. Then the* mutual information of X and Y conditioned over Z *(in bits) is*

$$I(X;Y|Z) = \sum_x \sum_y \sum_z p_{X,Y,Z}(x,y,z) \log \frac{p_Z(z) p_{X,Y,Z}(x,y,z)}{p_{X,Z}(x,z) p_{Y,Z}(y,z)}$$

where the logarithm is base-2 and $p_{X,Y}$*, etc. are probability mass functions, e.g.* $p_{X,Z}(x,z) = \Pr(X = x, Z = z)$*. The* mutual information $I(X;Y)$ *of* X *and* Y *is defined as* $I(X;Y|Z)$ *for a constant* Z*.*

Fact. If X, Y, Z, W are random variables, then $I(X; Y|Z, W) \leq I(X, W; Y|Z)$. This follows easily from the relationships between mutual information and (conditional) entropy [14, Sect. 2.4].

We identify wires and their corresponding random variables. Also each set of wires is identified with a tuple of random variables (in some order of wires, fixed for the workflow) considered as a single composite random variable. Thus we can write $I(\mathcal{A}; \mathcal{C})$ as the mutual information between the sets of wires $\mathcal{A}$ and $\mathcal{C}$.

Lemma 1. *Let $\mathcal{A}$ be the set of all input wires of a component M. Let $\mathcal{B}$ be subset of the output wires of M. Let $\mathcal{C}$ be a subset of wires into which there is no path from M. Then $I(\mathcal{B}; \mathcal{C}|\mathcal{A}) = 0$.*

Proof. Follows from the definition of the run. □

For each set of wires $\mathcal{X}$, let

- $\mathrm{V}(\mathcal{X})$ be the set of possible values of the wires $\mathcal{X}$,
- $\mathrm{d}(\mathcal{X})$ be the distribution of the values on the wires $\mathcal{X}$,
- $\mathcal{D}(\mathcal{X})$ be the set of all distributions over $\mathrm{V}(\mathcal{X})$,
- $\mathrm{Const}(\mathcal{X})$ be the set of all constant distributions (also called degenerate distributions or deterministic distributions) over $\mathrm{V}(\mathcal{X})$.

For each value v, let $\mathrm{Const}(v)$ be the constant distribution of the value v.

As inputs to the analysis, each component may have a description about the known bounds on the information flow from some subsets of its inputs to some subsets of its outputs. If there are no known bounds then the flows can be infinite. Each wire may also have a bound on the size of the values sent over that wire.

We are interested in $I(S; T)$. This quantity is uniquely determined by the distribution $\mathrm{d}(G)$ of the values of the global inputs G and the conditional distributions $P_{\mathcal{O}_M|\mathcal{I}_M=\boldsymbol{a}}$ for all M and $\boldsymbol{a}$ (which together induce the distribution of the values of all wires). We do not necessarily know these distributions exactly. Instead, the input of our analyzer includes declarations that restrict the distribution $\mathrm{d}(G)$ to a subset of $\mathcal{D}(G)$ and for all M, restrict the function f where $f(\boldsymbol{a}) = P_{\mathcal{O}_M|\mathcal{I}_M=\boldsymbol{a}}$, to a subset of $\mathcal{D}(\mathcal{O}_M)^{\mathrm{V}(\mathcal{I}_M)}$.

S may be a proper subset of $\mathcal{G}$. In this case, the global inputs $\mathcal{G}\backslash S$ are not considered sensitive. Thus, we may assume that the adversary already knows the value of $\mathcal{G}\backslash S$, i.e. its distribution may be considered constant. Thus $\mathrm{d}(\mathcal{G}\backslash S) \in \mathrm{Const}(\mathcal{G}\backslash S)$. Therefore, our goal is to compute an upper bound on the value

$$\max_{\substack{\mathrm{d}(S)\in\mathcal{D}_S \\ \mathrm{d}(\mathcal{G}\backslash S)\in\mathrm{Const}(\mathcal{G}\backslash S)}} I(S; T) \tag{1}$$

where $\mathcal{D}_S$ is the set of distributions into which the distribution of S is known to belong, according to the sensitivity declarations that will be described in Sect. 6.1.

We compute an upper bound on $I(S; T)$ using similar bounds for the individual components, i.e. $I(\mathcal{A}, \mathcal{C})$ for each component M, for all $\mathcal{A} \subseteq \mathcal{I}_M, \mathcal{C} \subseteq \mathcal{O}_M$.

Let $\mathcal{I} = \mathcal{I}_M$ and $\mathcal{C} \subseteq \mathcal{O}_M$. For each $\mathcal{A} \subseteq \mathcal{I}$ and $\mathcal{D} \subseteq \mathcal{D}(\mathcal{A})$, let

$$q_M^{\mathcal{D}}(\mathcal{A};\mathcal{C}) = \max_{\substack{\mathrm{d}(\mathcal{A}) \in \mathcal{D} \\ \mathrm{d}(\mathcal{I}\setminus\mathcal{A}) \in \mathrm{Const}(\mathcal{I}\setminus\mathcal{A})}} I(\mathcal{A};\mathcal{C}) \tag{2}$$

We take the maximum over all distributions that $\mathcal{A}$ may belong to because we do not know what the actual distribution on $\mathcal{A}$ is and we want $q_M^{\mathcal{D}}(\mathcal{A};\mathcal{C})$ to be an upper bound on $I(\mathcal{A};\mathcal{C})$. If we do not have any knowledge about the distribution of $\mathcal{A}$ then $\mathcal{D} = \mathcal{D}(A)$. If we have already determined the possible distributions of the inputs $\mathcal{I}_M$ (as will be described in Sect. 6.1) then we can write $q_M(\mathcal{A};\mathcal{C})$ instead of $q_M^{\mathcal{D}}(\mathcal{A};\mathcal{C})$.

The description of a component M should ideally contain the values $q_M(\mathcal{A};\mathcal{C})$ for all subsets of $\mathcal{A}$ of the inputs of M and all subsets $\mathcal{C}$ of the outputs of M. Because it may be difficult to determine the values $q_M(\mathcal{A};\mathcal{C})$, we may instead have upper bounds on these values. Also, we may not have the values for all $\mathcal{A}$ and $\mathcal{C}$.

The triangle equality does not hold for q_M. Thus it is possible that

$$q_M(\mathcal{A}_1;\mathcal{C}) + q_M(\mathcal{A}_2;\mathcal{C}) < q_M(\mathcal{A}_1 \cup \mathcal{A}_2;\mathcal{C})$$

or

$$q_M(\mathcal{A};\mathcal{C}_1) + q_M(\mathcal{A};\mathcal{C}_2) < q_M(\mathcal{A};\mathcal{C}_1 \cup \mathcal{C}_2)$$

Thus it does not in general suffice to give $q_M(\mathcal{A};\mathcal{C})$ only for one-element sets $\mathcal{A}$ and $\mathcal{C}$, because no bounds for larger sets of wires can be deduced from these.

Monotonicity does hold:

$$\mathcal{A}' \subseteq \mathcal{A} \wedge \mathcal{C}' \subseteq \mathcal{C} \Rightarrow q_M(\mathcal{A}';\mathcal{C}') \leq q_M(\mathcal{A};\mathcal{C})$$

but it may not give the best upper bound on $q_M(\mathcal{A}';\mathcal{C}')$.

5 Differential Privacy

Because q_M does not satisfy the triangle inequality, we may instead use a different quantity that does satisfy the triangle inequality and that implies a bound on q_M.

Definition 5. *Let P_1 and P_2 be discrete probability distributions over a set X. Then P_1 and P_2 are* ε-close *iff for all $x \in X$, $P_1(x) \cdot e^{-\varepsilon} \leq P_2(x) \leq P_1(x) \cdot e^{\varepsilon}$. Let the* differential-privacy distance *between P_1 and P_2 be the smallest value ε (which may be ∞) such that P_1 and P_2 are ε-close.*

Definition 6. *Let P be a probability distribution over $\mathcal{A}$. Denote by $d_M^P(\mathcal{A};\mathcal{C})$ the least value ε (which may also be ∞) such that for all value tuples $\boldsymbol{a}$ and $\boldsymbol{a}'$ of the inputs $\mathcal{A}$ for which $P(\boldsymbol{a}) > 0$ and $P(\boldsymbol{a}') > 0$, for all value tuples $\boldsymbol{b}$ of the inputs $\mathcal{I}_M\setminus\mathcal{A}$, the probability distributions $P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a},\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}$ and $P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a}',\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}$ are ε-close. For any set $\mathcal{D}$ of probability distributions over $\mathcal{A}$, let*

$$d_M^{\mathcal{D}}(\mathcal{A};\mathcal{C}) = \max_{P \in \mathcal{D}} d_M^P(\mathcal{A};\mathcal{C})$$

We have the following connection between differential privacy and information flow.

Lemma 2. *Let $d^{\mathcal{D}}_M(\mathcal{A};\mathcal{C}) = \varepsilon$. Then $q^{\mathcal{D}}_M(\mathcal{A};\mathcal{C}) \le q$ bits, where*

$$q = \varepsilon \frac{(e^{\varepsilon} - 1)(1 - e^{-\varepsilon})}{(e^{\varepsilon} - 1) + (1 - e^{-\varepsilon})} \cdot \frac{1}{\ln 2} \tag{3}$$

Proof. The proof is similar to [7]. Table 1 lists correspondences between some notions in our paper and in [7]. Let $D(P \parallel Q) = \sum_{\mathbf{a}} P(\mathbf{a}) \log(P(\mathbf{a})/Q(\mathbf{a}))$ be the Kullback-Leibler divergence from Q to P.

Table 1. Correspondence between the notions in our paper and [7]

Our paper	[7]
$q^{\mathcal{D}}_M(\mathcal{A};\mathcal{C}) = \max\limits_{\substack{\mathrm{d}(\mathcal{A})\in\mathcal{D} \\ \mathrm{d}(\mathcal{I}\setminus\mathcal{A})\in\mathrm{Const}(\mathcal{I}\setminus\mathcal{A})}} I(\mathcal{A};\mathcal{C})$	$\sup\limits_{i,P_{X^n}} I(X_i;Y\|X^{-i})$
$\exists P \in \mathcal{D}.\ P(\boldsymbol{a}) > 0 \wedge P(\boldsymbol{a}') > 0$	Databases D and $\tilde{D}$ are neighbors (they differ in at most one entry)
$d^{\mathcal{D}}_M(\mathcal{A};\mathcal{C}) = \epsilon$	ϵ is the least value such that for all neighboring databases x^n and $\tilde{x}^n$, $P_{Y\|X^n=x^n} \approx^{(\epsilon,0)} P_{Y\|X^n=\tilde{x}^n}$

We have $\forall P \in \mathcal{D} :\ P(\boldsymbol{a}) > 0 \wedge P(\boldsymbol{a}') > 0.\ \forall \boldsymbol{b}.\ P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a},\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}$ and $P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a}',\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}$ are ε-close. This is analogous to the statement [7] that for all neighboring databases x^n and $\tilde{x}^n$, $P_{Y|X^n=x^n}$ and $P_{Y|X^n=\tilde{x}^n}$ are ε-close. Both of these statements characterize ε-differential privacy.

Cuff and Yu [7] show that if P and Q are ε-close, then $D(P \parallel Q) \le q$ bits and $D(Q \parallel P) \le q$ bits where q is as in (3). I.e. we have $\forall P \in \mathcal{D} :\ P(\boldsymbol{a}) > 0 \wedge P(\boldsymbol{a}') > 0.\ \forall \boldsymbol{b}.\ D(P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a},\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}} \parallel P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a}',\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}) \le q$ bits.

Consider any case where $P \in \mathcal{D}$, $\mathcal{A} \sim P$, and $\mathcal{I}_M\setminus\mathcal{A} = \boldsymbol{b}$. Then, analogously to [7],

$$\begin{aligned} I(\mathcal{A};\mathcal{C}) &= \mathbb{E}_{\mathcal{A}} D(P(\mathcal{C}|\mathcal{A}) \parallel P(\mathcal{C})) \\ &= \mathbb{E}_{\mathcal{A}} D(P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a},\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}} \parallel \mathbb{E}_{P(\boldsymbol{a}')>0} P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a}',\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}) \\ &\le \mathbb{E}_{\mathcal{A}} \mathbb{E}_{P(\boldsymbol{a}')>0} D(P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a},\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}} \parallel P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a}',\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}) \le q \text{ bits} \end{aligned}$$

Because this holds for all considered cases, we have $q^{\mathcal{D}}_M(\mathcal{A};\mathcal{C}) \le q$ bits. □

If we have already determined the possible distributions of the inputs $\mathcal{I}_M$ (as will be described in Sect. 6.1) then we can write $d_M(\mathcal{A};\mathcal{C})$ instead of $d^{\mathcal{D}}_M(\mathcal{A};\mathcal{C})$. Then d_M satisfies triangle inequality for inputs:

$$d_M(\mathcal{A}_1;\mathcal{C}) + d_M(\mathcal{A}_2;\mathcal{C}) \ge d_M(\mathcal{A}_1 \cup \mathcal{A}_2;\mathcal{C})$$

Thus the description of a component may give $d_M(\mathcal{A};\mathcal{C})$ only for the cases where $\mathcal{A}$ is a one-element set, then we can use the triangle inequality to find an upper bound on $d_M(\mathcal{A};\mathcal{C})$ for the cases where $\mathcal{A}$ is a larger set, and then convert this to an upper bound on $q_M(\mathcal{A};\mathcal{C})$.

Note that d_M may not satisfy triangle inequality for outputs. If the outputs C_1 and C_2 are calculated from the input A (which is in some bounded range) by adding r and $-r$ to them, respectively, where r is a Laplace random value, then $d_M(A;C_1) = d_M(A;C_2)$ is finite but $d_M(A;C_1,C_2) = \infty$ because the randomness in C_1 and C_2 can be canceled out, revealing the exact value of A.

Differential privacy is useful for bounding leakages of information from a certain provenance but it may not always give the best bounds. For example, if we make in parallel 100 queries, each 0.1-differentially private, then the combination is 10-differentially private. When converted to mutual information (using (3)), this gives 14.4 bits of leakage. On the other hand, each 0.1-differentially private query separately, when converted to mutual information, leaks 0.0072 bits. Because results of the queries are conditionally independent (conditioned on the inputs), the triangle inequality holds here for mutual information, thus the 100 queries together leak only 0.72 bits, not 14.4 bits. Thus we get a much better bound on the leakage. This gives motivation for combining differential privacy and mutual information when bounding leakages.

Note that, in (3), $q \approx \frac{\varepsilon^2}{2\ln 2}$ when ε is small. This is one of the reasons that we use Shannon entropy instead of min-entropy. If we used min-entropy, we would get the bound $q = \frac{\varepsilon}{\ln 2}$ [1], even when ε is small. When the output $\mathcal{C}$ can have only 2 possible values then [1] gives an improved bound $q \approx \frac{\varepsilon}{2\ln 2}$ when ε is small. Now consider the example in the previous paragraph. Each 0.1-differentially private query, when converted to min-entropy, leaks at most 0.0703 bits of min-entropy if the output is binary, and 0.144 bits in the general case. The 100 queries together leak either 7.03 or 14.4 bits. Thus, combining differential privacy with min-entropy during the whole analysis, we would get no or only a small improvement over the bound (14.4 bits) that we get when using only differential privacy in the analysis and converting the final result to min-entropy. On the other hand, as described in the previous paragraph, combining differential privacy with Shannon entropy during the whole analysis improves the bound 20 times compared to using only differential privacy in the analysis and converting the final result to mutual information.

6 Inputs to the Analysis

Our information-flow analysis takes as input the graphical description of the workflow — the names of tasks and ports, as well as the wires from one port to another. It takes as input the subsets S and T of wires, stating which global inputs contain sensitive information, and which wires are read by the adversary. It also takes as input the information flow behaviour of tasks. The latter may be expressed in many different kinds, which we describe below.

6.1 Sensitivity

For each wire w, let dist_w be a distance (metric) on $\mathrm{V}(w)$. Let

$$\beta_0(w) = \max_{a,a' \in \mathrm{supp}\, \mathrm{d}(w)} \mathrm{dist}_w(a, a') \tag{4}$$

This is the diameter (according to dist_w) of the support of the distribution of w.

Our analysis can make use of declarations that the support of the distribution of a global input w has diameter (according to dist_w) at most s. In this case, let $\beta(w) = s$. For those global inputs w for which there is no such declaration, let $\beta(w) = \infty$. Then $\beta(w) \geq \beta_0(w)$ for all global inputs w.

Our analysis can also make use of declarations that (M, A, C) (where $A \in \mathcal{I}_M, C \in \mathcal{O}_M$) has c-sensitivity. This means that

- for all $a, a' \in \mathrm{V}(A), \mathbf{b} \in \mathrm{V}(\mathcal{I}_M \backslash \{A\}), d, d' \in \mathrm{V}(C)$:
  - if M may output d on C if it gets a on A and $\mathbf{b}$ on $\mathcal{I}_M \backslash \{A\}$
  - and M may output d' on C when it gets a' on A and $\mathbf{b}$ on $\mathcal{I}_M \backslash \{A\}$
  - then $\mathrm{dist}_C(d, d') \leq c \cdot \mathrm{dist}_A(a, a')$.

In other words, if we change the input A by a certain distance then the output C can change by at most c times that distance. The component M may have sensitivity declarations for several pairs of its inputs and outputs. Denote $c(A, C) = c$ if (M, A, C) has c-sensitivity and $c(A, C) = \infty$ if there does not exist c such that (M, A, C) has c-sensitivity, or such c has not been given.

All sensitivity declarations involving a certain wire (either as an input or an output of a component, or as a global input) must use the same distance dist_w on the values of that wire. If the values are databases then distance may be e.g. the number of records differing in the two versions of the database. If the values are scalars then the distance may be the absolute value of the difference of the two versions of the value.

If we know $\beta(A)$ and dist_A for all $A \in \mathcal{A}$ then we can find the set of distributions $\mathcal{D}$ used implicitly in $d_M(\mathcal{A}; \mathcal{C})$ and $q_M(\mathcal{A}; \mathcal{C})$ to denote $d_M^{\mathcal{D}}(\mathcal{A}; \mathcal{C})$ and $q_M^{\mathcal{D}}(\mathcal{A}; \mathcal{C})$, respectively:

$$\mathcal{D} = \mathcal{D}_{\mathcal{A}} = \{P \mid \forall A \in \mathcal{A}, a, a' \in \mathrm{supp}\, P|_A .\ \mathrm{dist}_A(a, a') \leq \beta(A)\} \tag{5}$$

6.2 Differential Privacy

Consider a component M and one of its inputs A. Let d_{dp} be the differential-privacy distance defined on the distributions of a subset of its outputs $\mathcal{C}$.

Our analysis can make use of declarations that $(M, A, \mathcal{C})$ has ε-differential privacy. This means that for all $a, a', \boldsymbol{b}$: $d_{\mathsf{dp}}(P_{\mathcal{C}|A=a, \mathcal{I}_M \backslash \{A\}=\boldsymbol{b}}, P_{\mathcal{C}|A=a', \mathcal{I}_M \backslash \{A\}=\boldsymbol{b}}) \leq \varepsilon \cdot \mathrm{dist}_A(a, a')$. If such declaration exists for some M, A, and $\mathcal{C}$, then denote this value ε by $\varepsilon(A, \mathcal{C})$. Put $\varepsilon(A, \mathcal{C}) = \infty$, if no such declaration exists.

Our analysis can also make use of declarations that $(M, \mathcal{A}, \mathcal{C})$ has sensitivity-less ϵ-differential privacy. This means that for all $\boldsymbol{a}, \boldsymbol{a'}, \boldsymbol{b}$:

$d_{\mathsf{dp}}(P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a},\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}, P_{\mathcal{C}|\mathcal{A}=\boldsymbol{a'},\mathcal{I}_M\setminus\mathcal{A}=\boldsymbol{b}}) \le \epsilon$. If such declaration exists for some M, A, and $\mathcal{C}$, then denote this value ϵ by $\epsilon(A, \mathcal{C})$. Put $\epsilon(A, \mathcal{C}) = \infty$, if no such declaration exists.

6.3 Mutual Information

Our analysis can make use of declarations that a component M leaks at most q bits from a subset $\mathcal{A}_i$ of its inputs to a subset $\mathcal{C}_j$ of its outputs, i.e. $q_M^{\mathcal{D}(\mathcal{A}_i)}(\mathcal{A}_i, \mathcal{C}_j) \le q$. This implies $q_M(\mathcal{A}_i, \mathcal{C}_j) \le q$. These are the mutual information declarations for $(M, \mathcal{A}_i, \mathcal{C}_j)$, meaning that $(M, \mathcal{A}_i, \mathcal{C}_j)$ has at most q bits of mutual information. Here the triangle inequality does not hold.

7 Analysis

The goal of our analysis is to conservatively estimate (i.e. upper-bound) (1). To compute it, we make several passes over the description of the workflow. These passes result us in finding $q_M(\mathcal{A}, \mathcal{C})$ for each component M, for all subsets $\mathcal{A}$ of its inputs and all subsets $\mathcal{C}$ of its outputs. We will then invoke a graph-theoretic algorithm that computes (1) from all $q_M(\mathcal{A}, \mathcal{C})$. We describe the computations below.

7.1 Bounding the Information Flow Through Components

Computing β for all wires. In Sect. 6.1, we defined $\beta(w)$ for all global inputs w and we showed that it is an upper bound of $\beta_0(w)$ (4) in this case. For any other wire C (taken in topological order), which belongs to $\mathcal{O}_M$ for some component M, we can compute $\beta(C)$ as

$$\beta(C) = \sum_{A \in \mathcal{I}_M} \beta(A) \cdot c(A, C)$$

It is easy to see, by induction and using the triangle inequality for dist_C, that $\beta(w) \ge \beta_0(w)$ for all wires w. If we know that $\beta(w) = s$ then we know that the distribution of the values on w is such that any two values with non-zero probability are at a distance at most s from each other.

Parallel Composition of Differential Privacy. For each component M and $\mathcal{A} \subseteq \mathcal{I}_M, \mathcal{C} \subseteq \mathcal{O}_M$, let

$$\gamma(M, \mathcal{A}, \mathcal{C}) = \min\{\epsilon(\mathcal{A}, \mathcal{C}), \sum_{A \in \mathcal{A}} \min\{\varepsilon(A, \mathcal{C}) \cdot \beta(A), \epsilon(A, \mathcal{C})\}\}.$$

It is easy to see that $\epsilon(\mathcal{A}, \mathcal{C}) \ge d_M^{\mathcal{D}(\mathcal{A})}(\mathcal{A}, \mathcal{C})$, $\epsilon(A, \mathcal{C}) \ge d_M^{\mathcal{D}(A)}(A, \mathcal{C})$, $\varepsilon(A, \mathcal{C}) \cdot \beta(A) \ge d_M(A, \mathcal{C})$. Now, using the triangle inequality for d_{dp}, we get that

$$\gamma(M, \mathcal{A}, \mathcal{C}) \ge d_M(\mathcal{A}, \mathcal{C}) = d_M^{\mathcal{D}}(\mathcal{A}, \mathcal{C}), \tag{6}$$

where $\mathcal{D}$ is as in (5).

Bounding the Mutual Information Through a Component. Consider a component M. Let $\mathcal{A}$ be the subset of its inputs and $\mathcal{C}$ the subset of its outputs that are on the path from the source to the sink. Suppose we want to find a bound on how much information can flow through M from $\mathcal{A}$ to $\mathcal{C}$, i.e. an upper bound on $q_M^{\mathcal{D}}(\mathcal{A};\mathcal{C})$, where $\mathcal{D}$ is the set of distributions into which the actual distribution of $\mathcal{A}$ is known to belong. $\mathcal{D}$ is determined by the sensitivity declarations, as described in Sect. 6.1. If there are no sensitivity declarations about the wires in $\mathcal{A}$ then $\mathcal{D} = \mathcal{D}(A)$.

If we have a mutual-information declaration for $(M, \mathcal{A}, \mathcal{C})$ then we can use the bound from that declaration. If we have a mutual-information declaration for $(M, \mathcal{A}', \mathcal{C}')$ where $\mathcal{A} \subseteq \mathcal{A}'$ and $\mathcal{C} \subseteq \mathcal{C}'$ then by monotonicity we can also use that bound. If we get bounds from several declarations then we take the minimum of those bounds.

If we have a differential-privacy declaration for $(M, \mathcal{A}, \mathcal{C})$ then we use that to find an upper bound on $d_M(\mathcal{A}, \mathcal{C})$. If we have differential-privacy declarations for $(M, A, \mathcal{C})$ for each $A \in \mathcal{A}$ then we use (6) to find an upper bound on $d_M(\mathcal{A}, \mathcal{C})$. Then we convert the bound on $d_M^{\mathcal{D}}(\mathcal{A}, \mathcal{C})$ to a bound on $q_M^{\mathcal{D}}(\mathcal{A}, \mathcal{C})$ using Lemma 2.

7.2 Maximum Information Flow in a Workflow

After we have obtained the upper bounds on the mutual information between the inputs and outputs of each component, we use Algorithm 1 to find the maximum information flow F in the whole workflow. This is an upper bound on the amount of information that an adversary can leak from S to T. Based on the workflow, and the input and output wires, the algorithm constructs a network

Algorithm 1. Maximum information flow in a system

Input: A set of components and directed wires between them, forming a dag. Some wires have no beginning component, these are the global inputs. Some wires may have no end component. S is a subset of global inputs. T is a subset of all wires.

```
Find (e.g. using breadth-first search) all wires and components through which there
is a path from $S$ to $T$.
Remove all other wires and components.
Set the capacity of each wire to be the maximum entropy of the data that can be
sent over the wire (e.g. the number of bits for fixed-length data).
for each remaining component $M$ do
    Find its remaining input wires $A_M$ and its remaining output wires $C_M$.
    Find a bound on $q_M(A_M; C_M)$ as described in Sect. 6.3.
    Replace the component $M$ with vertices $\mathsf{In}_M$ and $\mathsf{Out}_M$ so that
        the wires $A_M$ now enter $\mathsf{In}_M$ and
        the wires $C_M$ now begin from $\mathsf{Out}_M$.
    Add an edge from $\mathsf{In}_M$ to $\mathsf{Out}_M$ with capacity $q_M(A_M; C_M)$.
Add a vertex Source from which the wires $S$ begin.
Add a vertex Sink into which the wires $T$ enter.
Find the maximum flow from Source to Sink.
return the maximum flow.
```

(a directed graph, where each arc has been labeled with its capacity, together with distinguished source and sink vertices), such that the maximum flow in this graph is the upper bound that we seek. The following theorem states that F is indeed an upper bound to the amount of information that can be leaked.

Theorem 1 (Correctness of Algorithm 1). *Suppose that Algorithm 1 has been run, finding the maximum flow F in a system. Assume that* $\mathrm{d}(S) \in \mathcal{D}_S$ *and* $\mathrm{d}(\mathcal{G}\backslash S) \in \mathrm{Const}(\mathcal{G}\backslash S)$*. Then* $I(S;T) \leq F$*.*

Proof. Let C be a minimum cut of the transformed graph in Algorithm 1. The inputs and the outputs of a component M in the transformed graph, are A_M and C_M, respectively. In this proof, the occurrences of words like "edge", "path", etc. refer to the transformed graph, not the original graph. W.l.o.g. we can assume that C contains all zero-capacity edges of the transformed graph (because adding edges with zero capacity to the cut does not change the minimality of the cut). Let D be the set of edges outside C from which there is a path to T that does not contain any of the edges in C. Let $e_1, \ldots, e_s$ be the edges in $C \cup D$ in a topological order. Each edge corresponds to either a wire or a component in the original workflow. For each edge e, let

$$o(e) = \begin{cases} C_M & \text{if}\, e \text{ corresponds to a component}\, M \\ \{w\} & \text{if } e \text{ corresponds to a wire}\, w \end{cases}$$

$$c(e) = \begin{cases} \text{the capacity of } M & \text{if } e \in C \,\text{and } e \text{ corresponds to a component } M \\ \text{the capacity of } w & \text{if } e \in C \,\text{and } e\text{corresponds to a wire}\, w \\ 0 & \text{if } e \in D \end{cases}$$

Then we prove by induction that for all $i \leq s$,

$$I\left(S; \bigcup_{j=1}^{i} o(e_j)\right) \leq \sum_{j=1}^{i} c(e_j)$$

The case $i = 0$ holds because $I(S; \emptyset) = 0$.

Now suppose that

$$I\left(S; \bigcup_{j=1}^{i} o(e_j)\right) \leq \sum_{j=1}^{i} c(e_j)$$

holds. Let $Q = \bigcup_{j=1}^{i} o(e_j)$.

First consider the case where $e_{i+1} \in D$ corresponds to a component M. Consider an edge e corresponding to an input wire w of M. If $e \notin C$ then the path obtained by adding e to the beginning of a path from e_{i+1} to T that does not intersect C, is a path from e to T that does not intersect C, thus $e \in D$. Thus $e \in C \cup D$. Because there is path from e to e_{i+1}, e must be earlier in the topological order, i.e. $e = e_k$ for some $k < i + 1$. Because e corresponds to a wire w, $o(e_k) = w$, also $o(e_k) \subseteq Q$, thus $w \in Q$. Thus $A_M \subseteq Q$. Because of topological order, there is no path from M to $Q \backslash A_M$. Thus by Lemma 1, $I(S, Q\backslash A_M; C_M | A_M) = 0$. Also $c(e_{i+1}) = 0$. Now

$$I\left(S; \bigcup_{j=1}^{i+1} o(e_j)\right) = I(S; Q \cup C_M) = I(S; Q) + I(S; C_M|Q)$$

$$\leq I(S; Q) + I(S, Q\backslash A_M; C_M|A_M) = I(S; Q) \leq \sum_{j=1}^{i} c(e_j) = \sum_{j=1}^{i+1} c(e_j)$$

Now consider the case where $e_{i+1} \in C$ corresponds to a component M. Because of topological order, there is no path from M to $Q\backslash A_M$. Thus by Lemma 1, $I(S, Q\backslash A_M; C_M|A_M) = 0$. Also $c(e_{i+1}) \geq I(A_M; C_M)$. Now

$$\begin{aligned} & I(S; C_M|Q) \leq I(S, Q; C_M) \leq I(S, Q \cup A_M; C_M) \\ & = I(A_M; C_M) + I(S, Q\backslash A_M; C_M|A_M) \leq c(e_{i+1}) \\ & I\left(S; \bigcup_{j=1}^{i+1} o(e_j)\right) = I(S; Q \cup C_M) = I(S; Q) + I(S; C_M|Q) \\ & \leq \left(\sum_{j=1}^{i} c(e_j)\right) + c(e_{i+1}) = \sum_{j=1}^{i+1} c(e_j) \end{aligned}$$

Now consider the case where $e_{i+1} \in D$ corresponds to a wire w. Then there is a path from w to T that does not intersect C. w cannot be a global input because otherwise there would be a path from S to T that does not intersect C, thus it also would not contain zero-capacity edges, thus it would be an augmenting path with positive capacity, contradicting the minimality of the cut C. Thus w is an output of a component M. Consider an edge e corresponding to an input wire w of M. If $e \notin C$ then the path obtained by adding e to the beginning of a path from e_{i+1} to T that does not intersect C, is a path from e to T that does not intersect C, thus $e \in D$. Thus $e \in C \cup D$. Because there is path from e to e_{i+1}, e must be earlier in the topological order, i.e. $e = e_k$ for some $k < i + 1$. Now $w \in o(e_k)$ and $o(e_{i+1}) \subseteq o(e_k) \subseteq Q$. Also $c(e_{i+1}) = 0$. Thus

$$I\left(S; \bigcup_{j=1}^{i+1} o(e_j)\right) = I\left(S; \bigcup_{j=1}^{i} o(e_j)\right) \leq \sum_{j=1}^{i} c(e_j) = \sum_{j=1}^{i+1} c(e_j)$$

Now consider the case where $e_{i+1} \in C$ corresponds to a wire w. Then $c(e_{i+1}) \geq H(w)$, the entropy of the value on the wire. Thus

$$I(S; w|Q) \leq I(S, Q; w) = H(w) + H(S, Q) - H(S, Q, w) \leq H(w) \leq c(e_{i+1})$$

$$I\left(S; \bigcup_{j=1}^{i+1} o(e_j)\right) = I(S; Q, w) = I(S; Q) + I(S; w|Q)$$

$$\leq \left(\sum_{j=1}^{i} c(e_j)\right) + c(e_{i+1}) = \sum_{j=1}^{i+1} c(e_j)$$

We have thus proved the induction step for all cases. Now we can estimate $I(S; T)$. Consider any edge e corresponding to a wire in T. If $e \notin C$ then there is a path from e to T that does not intersect C, thus $e \in D$. Thus $e \in C \cup D$. Thus $T \subseteq C \cup D = \bigcup_{j=1}^{s} o(e_j)$.

$$I(S; T) \leq I(S; C \cup D) \leq \sum_{j=1}^{s} c(e_j) = F$$

Here the second inequality holds by the result we proved by induction. The equality holds by the maximum-flow-minimum-cut theorem ($\sum_{j=1}^{s} c(e_j)$ is the value of the minimum cut C). □

8 Completeness of Algorithm 1

We can also show the completeness of Algorithm 1 in some sense, i.e. that under certain conditions, certain (very strong) adversaries can bring the leakage arbitrarily close to the bound F, with arbitrarily small (but positive) error probability.

Suppose that for each port $p \in \mathit{Ports}$, the set Ports also contains ports $p^{(1)}, p^{(2)}, \ldots$ with $\mathrm{V}(p^{(i)}) = \mathrm{V}(p)$. For a set of ports P, let $P^{(1..n)}$ denote the set of ports $\{p^{(i)} \,|\, p \in P, i \in \{1, \ldots, n\}\}$. For a component M, let $M^{(n)}$ be the component "executing n copies of M in parallel". I.e. the input and output ports of $M^{(n)}$ are $\mathsf{ip}_{M^{(n)}} = \mathsf{ip}_M^{(1..n)}$, and $\mathsf{op}_{M^{(n)}} = \mathsf{op}_M^{(1..n)}$. The function $f_{M^{(n)}}$ takes the n copies of the inputs and independently applies f_M to each copy, resulting in n different sets of outputs.

Let M be a component and P_I, P_O subsets of its input and output ports. Let $f_I : \prod_{p \in P_I} \mathrm{V}(p) \to \mathcal{D}(\prod_{p \in P_I} \mathrm{V}(p))$ and $f_O : \prod_{p \in P_O} \mathrm{V}(p) \to \mathcal{D}(\prod_{p \in P_O} \mathrm{V}(p))$. Let $\mathbf{a}_I \in \prod_{p \in \mathsf{ip}_M \setminus P_I} \mathrm{V}(p)$. Let the mapping $\overline{f_M}$ have the same type as f_M, and be constructed by first applying f_I to the values appearing on P_I, then f_M to the results of f_I and the values $\mathbf{a}_I$ (i.e. the values on ports $\mathsf{ip}_M \backslash P_I$ are ignored), and finally f_O only to the outputs of f_M that would go to ports P_O in M (other outputs pass beside f_O). The *augmentation of M with $P_I, P_O, f_I, f_O, \mathbf{a}_I$* is the component $\mathsf{aug}(P_I, f_I, \mathbf{a}_I; \underline{M}; f_O, P_O)$ with the same input and output ports as M, and with the function $\overline{f_M}$.

The augmentation of a component is used to “change the encoding” of its inputs and outputs. If the mutual information between the inputs P_I and outputs P_O of M was q, then this is the bound also for the mutual information between the same inputs and outputs of $\mathsf{aug}(P_I, f_I, \mathbf{a}_I; M; f_O, P_O)$.

Let $WF = (\mathcal{M}, \mathcal{W}, \mathsf{s}, \mathsf{t})$ be a workflow. For each component $M \in \mathcal{M}$, let $P_{M;I}$ and $P_{M;O}$ be subsets of ip_M and op_M, respectively. For each n, let $\mathcal{S}^n_{M;I}$ and $\mathcal{S}^n_{M;O}$ be mappings with the following types:

$$\mathcal{S}^n_{M;I} : \prod_{p \in P^{(1..n)}_{M;I}} \mathrm{V}(p) \to \mathcal{D}(\prod_{p \in P^{(1..n)}_{M;I}} \mathrm{V}(p))$$
$$\mathcal{S}^n_{M;O} : \prod_{p \in P^{(1..n)}_{M;O}} \mathrm{V}(p) \to \mathcal{D}(\prod_{p \in P^{(1..n)}_{M;O}} \mathrm{V}(p)).$$

Also, let $\mathcal{S}^n_{M;\mathsf{v}} \in \prod_{p \in \mathsf{ip}_{M^{(n)}} \setminus P^{(1..n)}_{M;I}} \mathrm{V}(p)$. We consider $\mathcal{S}$ to be a function that maps a number n and a component (name) M into a pair of mappings and a tuple of values. We call the tuple of subsets of ports $[(P_{M;I}, P_{M;O})]_{M \in \mathcal{M}}$ the *type* of $\mathcal{S}$. We call $\mathcal{S}$ a *simulator* for WF.

The workflow $WF^{(n)}_{\mathcal{S}}$ intuitively executes n copies of WF, where each component $M^{(n)}$ has been augmented using $\mathcal{S}$. Formally, $WF^{(n)}_{\mathcal{S}} = (\mathcal{M}_n, \mathcal{W}_n, \mathsf{s}, \mathsf{t})$, where

- $\mathcal{M}_n = \{\mathsf{aug}(P_{M;I}, \mathcal{S}^n_{M;I}, \mathcal{S}^n_{M;\mathsf{v}}; M^{(n)}; \mathcal{S}^n_{M;O}, P_{M;O}) \,|\, M \in \mathcal{M}\}$;
- $\mathcal{W}_n = \{(w, i) \,|\, w \in \mathcal{W}, i \in \{1, \ldots, n\}\}$;
- $\mathsf{s}(p^{(i)}) = (\mathsf{s}(p), i)$ and $\mathsf{t}(p^{(i)}) = (\mathsf{t}(p), i)$ for all output and input ports of the components in $\mathcal{M}_n$.

Theorem 2. *Suppose that Algorithm 1 has been run, finding the maximum flow F in the workflow WF. For each component M, let $\mathcal{D}_M$ be the set of allowed probability distributions of A_M, as restricted by the sensitivity declarations. If for each component M, the bound $q_M = q^{\mathcal{D}_M}_M(A_M; C_M)$ found by the algorithm is tight, i.e. there exists $P \in \mathcal{D}_M$ such that if $A_M \sim P$ then $I(A_M; C_M) = q_M$, then for all $\epsilon > 0$, there exists a simulator $\mathcal{S}$ with type $[(A_M, C_M)]_{M \in \mathcal{M}}$, such that for each $\delta > 0$, there exists $n > 0$ such that the workflow $WF^{(n)}_{\mathcal{S}}$ can leak at least $n(F - \epsilon)$ bits of information with the error probability at most δ.*

Proof. Consider a component M. The weight of the edge e corresponding to this component in the flow graph is $q_0 = q^{\mathcal{D}}_M(\mathcal{A}; \mathcal{C})$. Let us run the maximum flow algorithm again with the weight of each edge corresponding to a component reduced by ϵ_0, i.e. $q = q_0 - \epsilon_0$. Then the maximum flow in this modified network is at least $F - K\epsilon_0$ where K is the number of components in the network and F is the flow in the original network. The flow through the edge e determined by the maximum flow algorithm is $f \leq q$.

Let $\mathrm{d}(\mathcal{A}) \in \mathcal{D}$ and $\mathrm{d}(\mathcal{I} \setminus \mathcal{A}) \in \mathrm{Const}(\mathcal{I} \setminus \mathcal{A})$ be such that maximize $I(\mathcal{A}; \mathcal{C})$ in (2). There are n copies of the workflow executed in parallel. The simulator $\mathcal{S}$ consists of pre- and postprocessing tools for each component M. There is a (single) preprocessor $\mathcal{S}^n_{M;A_M}$ before the n copies of M that takes the total of nf bits (assumed to be from the uniform distribution) on the n copies of the wires

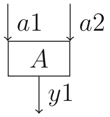
$\mathcal{A}$ destined to M and encodes them into an n-tuple whose components are each from the distribution $\mathrm{d}(\mathcal{A})$ (not necessarily independent). The tuple of constants $\mathcal{S}^n_{M;\mathsf{v}}$ has been picked from the constant distribution $\mathrm{d}(\mathcal{I}\backslash\mathcal{A})$; these are sent to the n copies of the wires $\mathcal{I}\backslash\mathcal{A}$ destined to M. There is a (single) postprocessor $\mathcal{S}^n_{M;C_M}$ after the n copies of M that takes the n-tuple from the n copies of $\mathcal{C}$ and decodes them into a total of nf bits.

By well-known results from information theory, the encoding/decoding (for using a channel with capacity at least $f + \epsilon_0$ for n times) can be chosen in such a way that these Nf bits are with probability at least $1 - \delta_0$ equal to the nf bits that were encoded by the simulator before the n copies of M. The probability that for each component M, the bits sent to the encoder before M are equal to the bits received from the decoder after M, is at least $1 - K\delta_0$. Thus also the probability that the $n(F - K\epsilon_0)$ bits of the source are equal to the $n(F - K\epsilon_0)$ bits of the sink, is at least $1 - K\delta_0$ (with the variables quantified as follows: $\forall\epsilon_0\forall\delta_0\exists n$). We can take $\epsilon = K\epsilon_0$ and $\delta = K\delta_0$ and get that the augmented workflow can leak $n(F - \epsilon)$ bits from the source to the sink with probability at least $1 - \delta$. □

9 Component Types

Here is a (non-exhaustive) list of component types that can be expressed in our system. Diagrams of the components are shown on the left and the corresponding declarations read by our analyzer are shown on the right.

9.1 Database Aggregator



```
comp A a1 a2 -> y1 ;
leak sens 20.0 a1 -> y1 ;
leak sens 50.0 a2 -> y1 ;
```

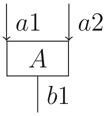
The declarations mean that $(A, a1, y1)$ has 20.0-sensitivity and $(A, a2, y1)$ has 50.0-sensitivity.

The inputs (here $a1$ and $a2$ but in general 1 or more inputs) are database tables and the component aggregates them to a scalar value $y1$. E.g. $y1$ may be the linear correlation coefficient of $a1$ and $a2$. If there is only one input table (e.g. $a1$) then $y1$ may be e.g. the mean, median, or standard deviation of $a1$.

The distance defined on any of its inputs a_i is the number of records by which the two database tables differ. The distance defined on its output $y1$ is the absolute value of the difference between the two scalar values.

For each input a_i, the component has sensitivity $c(a_i, y1)$. E.g. if $y1$ is the mean of a_i and each value in a_i is in the range $[L, R]$ then $c(a_i, y1) = \frac{R-L}{n}$, where n is the number of values (records) in a_i.

9.2 Database Linker



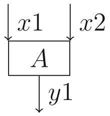
```
comp A a1 a2 -> b1 ;
leak sens 3 a1 -> b1 ;
leak sens 1 a2 -> b1 ;
```

The declarations mean that $(A, a1, b1)$ has 3-sensitivity and $(A, a2, b1)$ has 1-sensitivity.

The database tables $a1$ and $a2$ are linked by a column in each table. Let us call this column the *provenance column* and the possible values in this column the *provenances*. The table $a1$ must have at most one record with each provenance but $a2$ may contain up to r records with each provenance. Then the sensitivities are: $c(a1, b1) = r$ and $c(a2, b1) = 1$. This can be generalized to the case of linking more than 2 tables, of which only one may have non-unique provenances.

The output of a database linker may be used as an input of a database aggregator.

9.3 Scalar Combiner

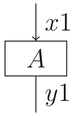


```
comp A x1 x2 -> y1 ;
leak sens 1.0 x1 -> y1 ;
leak sens 1.0 x2 -> y1 ;
```

Here $(A, x1, y1)$ has 1.0-sensitivity and $(A, x2, y1)$ has 1.0-sensitivity. The inputs (2 or more of them, here $x1$ and $x2$) are scalars. They are combined to calculate the output $y1$ (also a scalar).

This can be used to combine outputs of database aggregators. E.g. if $x1$ and $x2$ are the lower and upper quartile, respectively, of a database table then $y1$ may be the difference $x2 - x1$. In this case $c(x1, y1) = c(x2, y1) = 1$.

9.4 Laplace Randomizer



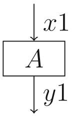
```
comp A x1 -> y1 ;
leak dpr 0.01 x1 -> y1 ;
```

The declarations mean that $(A, x1, y1)$ has 0.01-differential privacy.

The input $x1$ is a scalar value and the output $y1$ is calculated by adding Laplace noise from $\mathsf{Laplace}(\lambda)$ to $x1$. Here $\frac{1}{\lambda} = \varepsilon(x1, y1) = 0.01$. If $x1$ has sensitivity $\beta_0(x1) = c$ with respect to the global inputs then $\gamma(A, x1, y1) = \frac{c}{\lambda}$.

This can be combined with a database aggregator or scalar combiner to make their result differentially private.

9.5 Laplace Randomizer Without Sensitivity

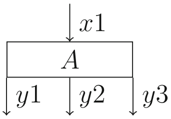


```
comp A x1 -> y1 ;
leak dp 2.0 x1 -> y1 ;
```

Here $(A, x1, y1)$ has sensitivity-less 2.0-differential privacy, with the keyword `leak dp` instead of `leak dpr` indicating that sensitivity is not used.

The input $x1$ is a scalar value and the output $y1$ is calculated by adding Laplace noise from $\mathsf{Laplace}(\lambda)$ to $x1$. The input does not need to have any sensitivity bound derived from sensitivity declarations. If it does have such a bound, it is ignored. Instead, we assume that $x1$ is in a certain range $[L, R]$ and if it is not there (by some mistake) then it is clipped into that range. Then we add Laplace noise from $\mathsf{Laplace}(\lambda)$ to $x1$. The result $y1$ is $\frac{R-L}{\lambda}$-differentially private. E.g. we may assume that $x1$ is a result of computing a linear correlation coefficient, being in the range $[-1, 1]$, and take $\lambda = 1$. Then the result is 2-differentially private, i.e. $\gamma(A, x1, y1) = 2$.

9.6 Secret Sharing



```
comp A x1 -> y1 y2 y3 ;
leak mi 0.0 x1 -> y1 y2 ;
leak mi 0.0 x1 -> y1 y3 ;
leak mi 0.0 x1 -> y2 y3 ;
leak mi 64.0 x1 -> y1 y2 y3 ;
```

The declarations mean that $(A, x1, \{y1, y2\})$ has at most 0.0 bits of mutual information, $(A, x1, \{y1, y3\})$ has at most 0.0 bits of mutual information, $(A, x1, \{y2, y3\})$ has at most 0.0 bits of mutual information, $(A, x1, \{y1, y2, y3\})$ has at most 64.0 bits of mutual information.

Here we secret share $x1$ into three shares $y1, y2, y3$. In the case of additive secret sharing, we would have $y1 \oplus y2 \oplus y3 = x1$, where $\oplus$ is addition modulo 2^k, where k is the bit length of each of the four values.

Here we have information-theoretical bounds on the flows. E.g. $q(x1; y1, y2) = q(x1; y1, y3) = q(x1; y2, y3) = 0$ but $q(x1; y1, y2, y3) = k$.

We can also express other kinds of secret sharing.

10 Implementation

We have implemented (in C++) Algorithm 1. The maximum flow from Source to Sink is computed using Edmonds-Karp algorithm. The implementation reads the description of the system, transforms it to a flow network, and finds the maximum flow in this graph. If the system has V components and E wires then the generated directed graph has at most $2V + 2$ nodes and at most $E + V$ edges. Thus the complexity is $O(VE^2)$. It can be improved by using a faster maximum-flow algorithm.

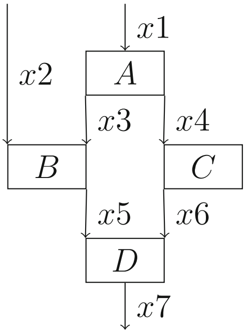


Fig. 1. A system

```
input x1 x2 ;
output x7 ;
comp A x1 -> x3 x4 ;
leak dp 0.2 x1 -> x3 ;
leak dp 0.2 x1 -> x4 ;
leak dp 0.4 x1 -> x3 x4 ;
comp B x2 x3 -> x5 ;
leak dp 0.2 x2 -> x5 ;
leak dp 0.2 x3 -> x5 ;
comp C x4 -> x6 ;
leak dp 0.2 x4 -> x6 ;
comp D x5 x6 -> x7 ;
leak dp 0.2 x5 -> x7 ;
leak dp 0.2 x6 -> x7 ;
check x1 -> x7 ;
check x2 -> x7 ;
check x1 x2 -> x7 ;
```

Fig. 2. Input file describing the system in Fig. 1

We have also implemented the idea in Sect. 5. We apply the triangle inequality for the inputs and get a bound on $d_M(\mathcal{A};\mathcal{C})$. We convert it to a bound on $q_M(\mathcal{A};\mathcal{C})$. We get another bound on $q_M(\mathcal{A};\mathcal{C})$ using only the known bounds on q_M and monotonicity (triangle inequality cannot be applied here). Either or both of the two bounds may also be infinite (i.e. no bound can be derived). Then we take the minimum of the two bounds.

11 Example

Figure 1 shows an example of a system with components A, B, C, D and wires $x1, x2, x3, x4, x5, x6, x7$. The corresponding input file describing this system is shown in Fig. 2. This file is read by our implementation.

The file describes the leakages using differential-privacy epsilons, which are shown as 0.2 for each single input and single output of each component. For the component A, we also give the leakage from $\{x1\}$ to $\{x3, x4\}$ because the triangle inequality cannot be used for outputs. The triangle inequality does hold for inputs and it is used to find the leakages involving more than one input of the same component. For example, consider component B. Its leak from $x2$ to $x5$ is 0.2, and from $x3$ to $x5$ is also 0.2. Then its leak from $(x2, x3)$ to $x5$ is $d_B(\{x2, x3\}; \{x5\}) = 0.4$. Then we convert these into upper bounds for the mutual-information-based leakages:

$$\begin{aligned} q_B(\{x2, x3\}; \{x5\}) &\leq 0.114 \\ q_B(\{x2\}; \{x5\}) &\leq 0.029 \\ q_B(\{x3\}; \{x5\}) &\leq 0.029 \end{aligned}$$

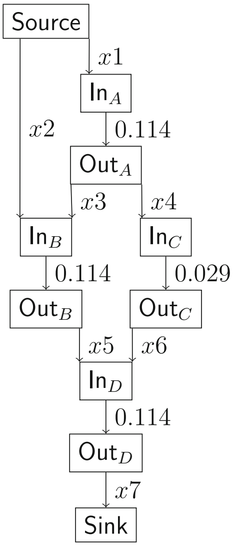


Fig. 3. Flow network from $\{x1, x2\}$ to $\{x7\}$ corresponding to the system in Fig. 1

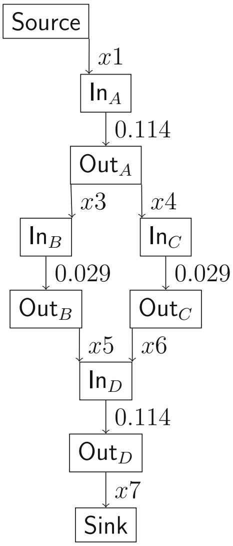


Fig. 4. Flow network from $\{x1\}$ to $\{x7\}$ corresponding to the system in Fig. 1

As we see, the triangle inequality does not hold for q_B.

Then a flow network for a subset of the global inputs and outputs is generated for the system. The result for the input subset $\{x1, x2\}$ and the output subset $\{x7\}$ is shown in Fig. 3. The wires with finite capacity have their capacity shown next to them, instead of their name. The direction of the edges is downwards. We find the maximum flow from Source to Sink, which is 0.114.

Considering the input subset $\{x1\}$ and the output subset $\{x7\}$, we get the flow network in Fig. 4. The capacity of the edge from In_B to Out_B is now 0.029 instead of 0.114, Reducing the maximum flow from Source to Sink to 0.058.

We also find the maximum flow from the input subset $\{x2\}$ to the output subset $\{x7\}$, getting 0.029. Thus the triangle inequality also does not hold for the global system, as $0.029 + 0.058 < 0.114$.

12 Conclusion

We have presented a method for analyzing leakages in workflows using leakage bounds for the individual components of the workflow. We combine both mutual information and differential privacy in our analysis to get better bounds on the leakages. We have also implemented the method. We conclude that using both differential privacy and mutual information can improve the privacy guarantees of workflows, compared to using either of them alone.

Acknowledgements. This research was funded by the Air Force Research laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under contract FA8750-16-C-0011. The views expressed are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This work has also been supported by Estonian Research Council, grant No. IUT27-1.

References

1. Alvim, M.S., Andrés, M.E., Chatzikokolakis, K., Degano, P., Palamidessi, C.: On the information leakage of differentially-private mechanisms. J. Comput. Secur. **23**(4), 427–469 (2015)
2. Backes, M., Köpf, B., Rybalchenko, A.: Automatic discovery and quantification of information leaks. In: 30th IEEE Symposium on Security and Privacy (S&P 2009), 17–20 May 2009, Oakland, pp. 141–153. IEEE Computer Society (2009)
3. Barros, J., Servetto, S.D.: Network information flow with correlated sources. IEEE Trans. Inf. Theory **52**(1), 155–170 (2006)
4. Barthe, G., Köpf, B., Olmedo, F., Béguelin, S.Z.: Probabilistic relational reasoning for differential privacy. ACM Trans. Program. Lang. Syst. **35**(3), 9 (2013)
5. Chatzikokolakis, K., Chothia, T., Guha, A.: Statistical measurement of information leakage. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 390–404. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12002-2_33
6. Clark, D., Hunt, S., Malacaria, P.: A static analysis for quantifying information flow in a simple imperative language. J. Comput. Secur. **15**(3), 321–371 (2007)
7. Cuff, P., Yu, L.: Differential privacy as a mutual information constraint. In: CCS 2016 (2016). http://arxiv.org/pdf/1608.03677
8. Dumas, M., García-Bañuelos, L., Laud, P.: Differential privacy analysis of data processing workflows. In: Kordy, B., Ekstedt, M., Kim, D.S. (eds.) GraMSec 2016. LNCS, vol. 9987, pp. 62–79. Springer, Cham (2016). doi:10.1007/978-3-319-46263-9_4
9. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006). doi:10.1007/11787006_1
10. Ebadi, H., Sands, D.: Featherweight PINQ. CoRR, abs/1505.02642 (2015)
11. Frau, S., Gorrieri, R., Ferigato, C.: Petri net security checker: structural non-interference at work. In: Degano, P., Guttman, J., Martinelli, F. (eds.) FAST 2008. LNCS, vol. 5491, pp. 210–225. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01465-9_14
12. Gaboardi, M., Haeberlen, A., Hsu, J., Narayan, A., Pierce, B.C.: Linear dependent types for differential privacy. In: Giacobazzi, R., Cousot, R. (eds.) The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2013, Rome, 23–25 January 2013, pp. 357–370. ACM (2013)
13. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229. ACM (1987)
14. Gover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. Wiley, New York (2006)
15. McCamant, S., Ernst, M.D.: Quantitative information flow as network flow capacity. In: Gupta, R., Amarasinghe, S.P. (eds.) Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, 7–13 June 2008, pp. 193–205. ACM (2008)

16. McSherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Çetintemel, U., Zdonik, S.B., Kossmann, D., Tatbul, N. (eds.) Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, 29 June–2 July 2009, pp. 19–30. ACM (2009)
17. Reed, J., Pierce, B.C.: Distance makes the types grow stronger: a calculus for differential privacy. In: Hudak, P., Weirich, S. (eds.) Proceeding of the 15th ACM SIGPLAN International Conference on Functional Programming, ICFP 2010, Baltimore, 27–29 September 2010, pp. 157–168. ACM (2010)
18. Smith, G.: On the foundations of quantitative information flow. In: Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 288–302. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00596-1_21