Lineage-Preserving Anonymization of the Provenance of Collection-Based Workflows

Khalid Belhajjame
PSL, Université Paris-Dauphine, LAMSADE
Paris, France
khalid.belhajjame@dauphine.fr

ABSTRACT

We examine in this paper the problem of anonymizing the provenance of collection-oriented workflows, in which the constituent modules use and generate sets of data records. Despite their popularity, this kind of workflows has been overlooked in the literature w.r.t privacy. We, therefore, set out in this paper to examine the following questions: *How the provenance of a collection-based module can be anonymized? Can lineage information be preserved? Beyond a single module, how can the provenance of a whole workflow be anonymized?* As well as addressing the above questions, we report on evaluation exercises that assess the effectiveness and efficiency of our solution. In particular, we tease apart the parameters that impact the quality of the obtained anonymized provenance information.

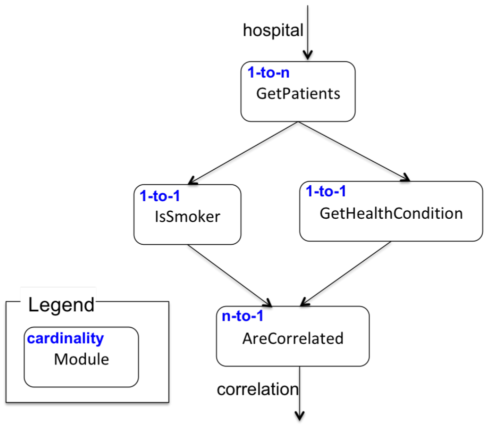

Fig. 1: Example workflow.

1 INTRODUCTION

Automated workflows have been shown to facilitate and accelerate scientific data exploration and analysis in many areas of sciences [11]. Figure 1 illustrates a simple workflow that is used to establish correlations between smoking and health conditions. Workflow provenance information, recorded during workflow executions, facilitates the interpretation of the results delivered by workflow execution. Beyond verification, workflow provenance information represents a useful dataset on its own right, that can be leveraged to answer queries that are relevant for an experiment that is (possibly related but) different from the original experiment, to learn new hypotheses, or to gain insight on the characteristics and quality of the data generated by given modules. Collected workflow provenance information can also be used to respond to the requirements of funding agencies that are increasingly requesting the publication of the data generated in the context of research investigations.

In fields such as biomedicine and social sciences, workflow executions manipulate and generate sensitive information about individuals. To promote the publication and sharing of the provenance of workflow executions, we set out in this paper to examine the problem of anonymizing workflow provenance.

1.1 Related Work

Related work has focused on the problem of securing workflow provenance and policing their access. For example, Chebotko *et al* [9] and Biton *et al* [7] proposed solutions that derive a partial view on a workflow provenance by hiding the data records of given modules Our objective is different from the above line of work in that we seek to provide the user with the provenance of all the modules of the workflow by leveraging anonymization.

Davidson *et al.* [12] investigated the problem of module privacy, whereby some of the parameters (attributes) characterizing the inputs and outputs of the modules are hidden to guarantee the privacy of modules. In our work, we seek, instead, to guarantee the privacy of the data records used and generated by the modules, instead of the behavior of the module.

We have examined in a previous work, the problem of identification of the k-anonymity degree that needs to be enforced when anonymizing the datasets used and generated by workflows [6]. In doing, we assumed that the modules that compose the workflow are 1-to-1 in that they produce a single data record, given a single data record, and we did not give much thought to the problem of lineage preservation. In this paper, we are interested in what we refer to as collection-based workflows [16–18, 27]. The modules that compose such workflows can take as input a collection of data records and deliver a collection of data records. Such workflows have been advocated as a way to better meet the needs of non-expert users to model scientific data [20], and to structure complex relationships among related pieces of information that are processed together by the workflow [22]. This class of workflows has been overlooked in the literature w.r.t. privacy.

Different techniques have been proposed in the literature for protecting the privacy of individuals, notably, k-anonymity [26] and differential privacy [15]. In particular, differential privacy [15] has recently gained momentum as the method of choice in statistical databases. It involves adding random noise to the data so that the distribution of the resulting dataset is almost invariant to the inclusion of any data record. While powerful, differential privacy is not suitable for our purposes. It assumes that the user knows up-front the queries s/he wants to issue prior to the anonymization. This is not the case in our setting, where the scientist issues exploratory queries for understanding and eventually interpreting the results of the workflows. Furthermore, for it to be useful, the scientist should be able to inspect individual data records and their relationships (lineage), both of which are not possible using differential privacy. Indeed, differential privacy is more suited for statistical (i.e., aggregation-based) queries.

For our work, we chose to use k-anonymity [26]. This method is not as powerful as differential privacy when it comes to privacy guarantees. Yet, it is better suited for our purposes since it can be instrumented, as we will show, to allow users to query and examine individual data records and their lineage within workflow provenance. k-anonymity is also still perceived by practitioners as sufficient for mitigating risk while maximizing utility, and real-world applications still utilize it for data sanitization (see e.g., [3, 10]). It is also widely popular and is used, e.g., in the healthcare world [1, 25], and is still recommended by data protection agencies (see e.g., [2]). This technique has been extensively investigated in the database and data mining communities [28]. Most of the proposals have focused on anonymizing a single relational table. In workflow provenance, however, we need to anonymize different datasets considering and preserving lineage relationships between them. One solution that can be used to anonymize workflow using k-anonymity would be to create a global relational table that is obtained by joining relations representing the input and output data records of the modules that compose the workflows. However, this solution suffers from the following issues. First, information about the same individual can be found in different records. This is because we consider collection based modules, e.g., a patient can be associated with multiple practitioners. Second, the same tuple in the global table may contain information about multiple individuals, e.g., a patient, one of its practitioners, etc. Moreover, as we will see later, different kinds of individuals may be associated with different k-anonymity degrees. For example, the k-anonymity degree associated with patients may be higher than that associated with practitioners. Traditional k-anonymity is not equipped to deal with the above issues. In this respect, the proposal by Nergiz *et al.* [24] is related to ours. They elaborated a technique that anonymizes multiple relations of a given database schema. While useful, this proposal makes a number of limiting assumptions. In particular, they consider snowflake schemas, in which there is a single relational table that represents individuals with the remaining relations containing quasi-attributes and having a single foreign key. In our work, we drop these assumptions and show that the anonymization of workflow provenance can be achieved in the presence of multiple datasets representing individuals with multiple relationships (foreign keys constraints) between them.

1.2 Contributions

Our first contribution is the formulation of the problem of k-anonymization of the provenance of collection-based workflows. This is, to our knowledge, the first paper that extends the notion of k-anonymization from a single relation to the provenance of workflows. Our second contribution is a technique for k-anonymizing the provenance of a single module, i.e., input and output records together with their lineage information. Indeed, lineage information tracing the dependencies between the output and input of a module (and more generally a workflow) is key for third-party scientists to understand and examine the validity of workflow results. We examine this problem for modules that use and generate collections of data records. Our third contribution extends the technique proposed to cater for the anonymization of the provenance of a workflow as a whole. Central to the solution we present is the notion of k-group anonymity, which we define based on the k-anonymity degree and the magnitude of the smallest input (or output) set of data records used and generated by a module. This concept allows us to gracefully reason over the different k-anonymity degrees that may be associated with the inputs and outputs of the workflow's modules. We also show how the NP-hard problem of identifying the sets of data records to be grouped together into equivalence classes that meet k-anonymity requirements can be cast as a scheduling problem that we solve using integer programming.

The paper is organized as follows. We start by laying the foundations of our work and stating the problem in Section 2. We then focus on the problem of anonymizing the provenance of a module in Section 3, and the provenance workflow in Section 4. In Section 5, we address an issue that is inherent to our anonymization technique, namely grouping sets of data records, and cast it as a scheduling problem. We report on evaluation exercises that we empirically conducted to assess the effectiveness and efficiency of our solution in Section 6, and conclude the paper in Section 7.

2 FOUNDATIONS

2.1 Collection-Based Module and Workflow

Definition 2.1 (module). A module $\mathtt{m}$ is defined by the tuple $(\mathtt{I_m}, \mathtt{O_m}, \mathtt{card})$, where $\mathtt{I_m}$ (resp. $\mathtt{O_m}$) is a set of ordered input (resp. output) ports, and $\mathtt{card}$ specifies the cardinality of $\mathtt{m}$. A port $\mathtt{p} = \langle \mathtt{a_1}, \ldots, \mathtt{a_n} \rangle$ is a list of attributes, each characterized with a basic type, e.g., String, Integer.

Assigning a data value to each attribute in a port gives rise to a data item, and assigning a data item to each input (output) port of a module gives rise to a data record.

$\mathtt{card} \in$ {1-to-1,1-to-n,n-to-1,n-to-n}: 1-to-1 specifies that the invocation of $\mathtt{m}$ takes as input a single data record and produces a single data record; n-to-1 (resp. 1-to-n) specifies that the invocation of $\mathtt{m}$ takes as input a list (ordered set) of data records (resp. single data record) and produces a single data record (resp. a list of data records); n-to-n specifies that the invocation of the module takes as input a list of data records and produces a list of data records.

Definition 2.2 (data link). A data link $\mathtt{dl}$ is defined by the pair $\mathtt{dl} = (\mathtt{m_i} : \mathtt{o_{m_i}}, \mathtt{m_j} : \mathtt{i_{m_j}})$, where $\mathtt{m_i} : \mathtt{o_{m_i}}$ designates an output port $\mathtt{o_{m_i}}$ of the module $\mathtt{m_i}$, and $\mathtt{m_j} : \mathtt{i_{m_j}}$ designates an input port $\mathtt{i_{m_j}}$ of the module $\mathtt{m_j}$.

Definition 2.3 (workflow). A workflow specification is defined by a pair $\mathtt{w} = (\mathtt{M}, \mathtt{E})$, where $\mathtt{M}$ is a set of modules and $\mathtt{E}$ is a set of data links. $\mathtt{w}$ has one initial module with no incoming data links, and one final module with no outgoing data links.

We consider acyclic workflows that have a single initial module and a single final module, and where each module in the workflow, other than the initial module, is reachable from the initial module. Workflow execution follows a pure dataflow model: a module $\mathtt{m}$ is invoked (is fireable) as soon as all of its input ports are bound to data items. During the workflow execution, data items are transferred between connected output and input ports. For example, the following data link binding $((\mathtt{m_1} : \mathtt{o_{m_1}}, \mathtt{m_2} : \mathtt{i_{m_2}}), \mathtt{di})$ specifies that the data item $\mathtt{di}$ was transferred using the data link connecting the output port $\mathtt{o_{m_1}}$ of $\mathtt{m_1}$ to the input port $\mathtt{i_{m_2}}$ of $\mathtt{m_2}$. The technical report [5] provides more information about the workflow execution model. It specifies how input data records of a module are constructed using the data records of the preceding modules in the workflow. It also specifies how mismatches in cardinalities between connected modules' outputs and inputs are resolved at execution time.

Table 1: Input and Output Provenance of admittedTo.

Input Patient DataSet			
ID	**name**	birth	lin
p_1	Garnick	1990	$\{r_1, r_2\}$
p_2	Hiyoshi	1987	$\{r_3, r_4\}$
p_3	Suessmith	1989	$\{r_5, r_6\}$
p_4	Solares	1985	$\{r_7, r_8\}$
p_5	Kading	1992	$\{r_9, r_{10}\}$
p_6	Pero	1988	$\{r_{11}, r_{12}\}$
p_7	Pehl	1986	$\{r_{13}, r_{14}\}$
p_8	Barriga	1995	$\{r_{15}, r_{16}\}$

Output Hospital DataSet		
ID	hospital	Lin
h_1	St Louis	$\{p_1, p_3\}$
h_2	St Anton	
h_3	St Anne	$\{p_2, p_4\}$
h_4	St August	
h_5	Holby	$\{p_5, p_7\}$
h_6	Larib.	
h_7	St James	$\{p_6, p_8\}$
h_8	St Mary	

2.2 Workflow Provenance as Relations

Definition 2.4. Given a workflow w, its provenance, denoted by prov(w), is the collection of modules and data link bindings that take place as a result of the executions of w.

For ease of exposition of our anonymization solution, we encode the provenance of a module m using two relational tables denoted by prov(m, w).in and prov(m, w).out. They contain the data records that were used and generated, respectively, by the invocations of m within the executions of a workflow w. we call prov(m, w).in (resp. prov(m, w).out) the input (resp. output) provenance of m. When the referred workflow w is clear from the context, we abuse the notation and simply use prov(m).in and prov(m).out to refer to such relations. The schema of such relations contains the attributes of the input ports (resp. output ports) of m. We assume that the attribute names are unique within the input (resp. output) ports of a module. From a provenance point of view, we do not keep information about the order of the data records in an input or output list, which is, therefore, viewed as a set. This is the case, for example, in the Taverna workflow system [30]. Because of this, we use in what follows the terms input/output set of data records, as opposed to list of data records. W.l.o.g, we assume that the attributes of two succeeding modules that have the same name are connected (via their ports) by data links. In other words, we can deduce data link bindings from module bindings, which allows us to write: $\mathrm{prov(w)} = \bigcup_{\mathrm{m} \in \mathrm{w.M}} (\mathrm{prov(m).in} \cup \mathrm{prov(m).out})$

Consider a module admittedTo that given a set of patients returns a set of hospitals that those patients were admitted to[1]. Table 1 illustrates an example of two relations representing input provenance and output provenance of the admittedTo module. The names of identifying attributes are written in bold, and the names of quasi-identifying attributes are underlined. Notice that the relations contain also two additional attributes: ID and Lin. The first is an ID that is generated internally by the workflow system to identify data records, and the second is used to encode lineage information. In the case of the input provenance, Lin specifies the data records produced by the preceding modules in the workflow and that were used in the construction of the data record in question. For example, the data record p_1 was constructed using two data records r_1 and r_2 that were produced by some preceding modules. The Lin column is empty for the relational table used to store the data records used as input to the initial module in the workflow. Regarding the output provenance of admittedTo, the Lin column identifies the data records that were used as input to obtain the output data record in question. For example, it specifies that h_1 and h_2 were generated given the inputs p_1 and p_3. The lineage information we consider here is in line with the *why provenance* semantics (see [8]).

2.3 Problem Statement

Adversary Model. The data records used and generated by a workflow module are characterized by three kinds of attributes: (i) Identifying attributes allow identifying individuals, e.g., the attribute name is an identifying attribute. (ii) Sensitive attributes are attributes that carry sensitive information, e.g., health condition. (iii) Quasi-identifying attributes are non-identifying attributes, but their combination can be used to identify an individual, e.g., address, phone number, etc. Notice that the ID attribute is not considered as an identifying attribute because it is generated by the workflow system and does not carry information that allows identifying individuals such as name for example.

We assume that an adversary may know identifying and quasi-identifying attribute values about individuals, e.g., name, address, date of birth. However, we assume that s/he does not know sensitive attribute values, e.g., health-condition, income tax.

In relational databases, a relation r is k-anonymized, where k is an integer greater than 2, if any data record d in r is not distinguishable from (at least) $k - 1$ other records in r. This condition is met by masking the values of identifying attributes, and generalizing the values of quasi-identifying attributes (e.g., address, visited hospital, etc.). Sensitive attributes, such as health condition, salary, are not masked: adversaries are assumed not to be knowledgeable of the values of sensitive attributes. In what follows, we use the term identifier record to refer to a data record that has an identifying attribute value, and the term quasi-identifier record to refer to a data record that has no identifying attribute value but has a quasi-identifying attributes value. A module input (resp. output) that is bound to identifier records following module invocation is called identifier input (resp. output). It is called quasi-identifier input (resp. output) if it is bound to quasi-identifier records.

Anonymity degree of Identifier Inputs and Outputs. We assume that every identifier input (resp. identifier output) of a module m is associated with an anonymity degree, which we denote by k^{i}_{m} (resp. k^{o}_{m}) to be enforced. Note that non-identifier module inputs and output are not associated with an anonymity degree because they are not bound at execution time to records that represent individuals. We do not make the assumption that the anonymity degrees associated with the identifier inputs and outputs of the modules that compose the workflow are the same. This is because the modules that compose a workflow are likely to use different underlying data sources that are supplied by different providers who may impose different requirements when it comes to the anonymity degree to be enforced on their data. Moreover, the same data provider may impose different anonymity degrees depending on the data that is retrieved from its source. For example, an input that provides information about patients and their health condition is likely to be associated with an anonymity degree that is higher than an output that informs on the trips of practitioners. In this paper, we apply k-anonymization to the provenance prov(w) of a workflow prov(w) by creating equivalence classes for the relations prov(m).in and prov(m).out for each identifier input and output of the modules in w.M.

Definition 2.5 (Equivalence Classes). Consider the input provenance prov(m).in of a module m. We say that the set $\{E1^{m}_{in}, \ldots, E1^{m}_{in}\}$, $n \geq 1$, is a set of input equivalence classes for m and write $\mathrm{prov}^{a}(\mathrm{m}).\mathrm{in} = \{E1^{m}_{in}, \ldots, E1^{m}_{in}\}$ iff:

1)- The set $\{E1^{m}_{in}, \ldots, E1^{m}_{in}\}$ forms a partitioning for prov(m).in. That is $\mathrm{prov(m).in} = \bigcup_{i \in [1,n]} Ei^{m}_{in}$, and $Ei^{m}_{in} \cap Ej^{m}_{in} = \emptyset$ for $i, j \in [1, n]$ s.t. $i \neq j$.

2)- The identifying attribute values of the data records in every equivalence class Ei^{m}_{in} are masked, and their quasi-identifying attribute values are generalized such that the data records in an

[1]A hospital appears in the result only if it was visited by each of the patients in the input set.

Table 2: Input and Output Provenance of `admittedTo` where the Input Provenance is 2-anonymized.

2-anonymized Patient DataSet			
ID	**name**	birth	lin
p_1	★	{1987,1990}	$\{\mathtt{r}_1, \mathtt{r}_2\}$
p_2	★	{1987,1990}	$\{\mathtt{r}_3, \mathtt{r}_4\}$
p_3	★	{1985,1989}	$\{\mathtt{r}_5, \mathtt{r}_6\}$
p_4	★	{1985,1989}	$\{\mathtt{r}_7, \mathtt{r}_8\}$
p_5	★	{1988,1992}	$\{\mathtt{r}_9, \mathtt{r}_{10}\}$
p_6	★	{1988,1992}	$\{\mathtt{r}_{11}, \mathtt{r}_{12}\}$
p_7	★	{1986,1995}	$\{\mathtt{r}_{13}, \mathtt{r}_{14}\}$
p_8	★	{1986,1995}	$\{\mathtt{r}_{15}, \mathtt{r}_{16}\}$

Hospital DataSet		
ID	hospital	Lin
h_1	St Louis	$\{p_1, p_3\}$
h_2	St Anton	
h_3	St Anne	$\{p_2, p_4\}$
h_4	St August	
h_5	Holby	$\{p_5, p_7\}$
h_6	Larib.	
h_7	St James	$\{p_6, p_8\}$
h_8	St Mary	

equivalence class $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$ are indistinguishable w.r.t. their quasi-identifying attribute values.

A set of output equivalence classes are defined in a similar manner: $\mathtt{prov^a(m).out} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{out}}, \ldots, \mathtt{E1}^{\mathtt{m}}_{\mathtt{out}}\}$.

Note that the `ID` and `Lin` attribute values of the data records are not generalized. This is because the values of the `ID` attribute are generated internally by the workflow system. In other words, they are not meaningful for human users. More importantly, they are used within the `Lin` attribute to encode lineage information that we seek to preserve.

Lineage information needs to be considered when k-anonymizing the input provenance (resp. output provenance) of an identifier module input (resp. output). To illustrate this, let us consider the `admittedTo` module. It has an identifier input and a quasi-identifier output. Consider that the anonymity degree associated with its input is $\mathtt{k}^{\mathtt{admittedTo}}_{\mathtt{i}} = 2$. Notice that its output is not associated with an anonymity degree because it is not an identifier output. Table 2 illustrates the input and output provenance of `admittedTo`, where the input provenance is 2-anonymized. The anonymization consisted in partitioning the set of input data records into input equivalence classes of size ≥ 2. Notice that this anonymization operation does not guarantee k-anonymization, however. To illustrate this, consider that an adversary knows that `Garnick` was born in `1990` and that he visited the `StLouis` hospital. By examining the output data records together with lineage information in `prov(admittedTo).out` (see Table 2), an adversary will be able to infer that the data record $\mathtt{p}_1$ refers to `Garnick`. This can be more of an issue when the data record contains sensitive information such as health condition.

Problem 1 (K-anonymization of the input and output provenance of a module). *Consider a module* `m` *with an identifier input (resp. output). k-anonymizing the input provenance* `prov(m).in` *(resp.* `prov(m).out`*) of* `m` *using an anonymity degree* $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$ *(resp.* $\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}$*) gives rise to anonymized input provenance* $\mathtt{prov^a(m).in}$ *(resp. anonymized output provenance* $\mathtt{prov^a(m).out}$*) where:*
1)- $\mathtt{prov^a(m).in} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{in}}, \ldots, \mathtt{En}^{\mathtt{m}}_{\mathtt{in}}\}$ *(resp.* $\mathtt{prov^a(m).out} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{out}}, \ldots, \mathtt{En}^{\mathtt{m}}_{\mathtt{out}}\}$*), is a set of input (resp. output) equivalence classes for the input (resp. output) provenance of* `m`*, with* $\mathtt{n} \geq 1$.
2)- *An input equivalence class* $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$ *(resp. output equivalence class* $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{out}}$*) contains at least* $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$ *(resp.* $\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}$*) data records.*
3)- *The data records in an input equivalence class* $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$ *(resp. output equivalence class* $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{out}}$*) cannot be distinguished by examining their lineage, i.e., by examining the data records that (transitively) contributed to the data records in* $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$ *(resp.* $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{out}}$*) or by examining the data records that the records in* $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$ *(resp.* $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{out}}$*) contributed to through workflow execution.*

Condition (3) is formally defined in the technical report using the notions of backward- and forward-lineage in a workflow [5].

Problem 2 (K-anonymization of the provenance of a workflow). *The provenance of a workflow* `w` *is said to be k-anonymized iff the input provenance of every identifier module input and the output provenance of every identifier module output in* `w.M` *are k-anonymized.*

The above problem is NP-Hard: Meyerson and Williams [21] demonstrated that optimal k-anonymity for a single relational table without considering lineage is an NP-hard problem. We present in this paper a heuristic that seeks to satisfy k-anonymity, to reduce the generalization (information-loss) incurred as a result, and to preserve lineage information in doing so.

3 ANONYMIZATION OF MODULE PROVENANCE

We show, in this section, how the input provenance and output provenance of a module are anonymized. The solution we present is applicable to many-to-many modules but also to modules with other cardinalities. We distinguish the case where the module input is an identifier input and its output is a quasi-identifier output, and the case where the module input and output are identifier input and identifier output. In the first case, the attribute values of the output data records are treated as quasi-identifying attribute values for their counterpart input data records. The second case is slightly more complex in the sense that the attribute values of the output data records are treated as quasi-identifying attribute values for their counterpart input data records, and vice-versa. We will not examine the case where both the module input and output carry quasi-identifier records. Indeed, it only makes sense to perform the anonymization when the input and/or the output carry identifier records, and as such associated with an anonymity degree to be enforced. That said, we will show in Section 4 how modules that carry quasi-identifier input and output records are dealt with in situations where they are used in workflows containing other modules with identifier records.

3.1 Module with Identifier Input and Quasi-Identifier Output

Consider the `admittedTo` module, presented earlier, that given a set of individuals returns a set of hospitals that those patients visited (see Table 1). And consider that the input dataset has been 2-anonymized as illustrated in Table 2. As discussed earlier, the lineage associating the output dataset to the input dataset may allow an adversary to pinpoint patients in the input dataset, even if this is anonymized. To avoid this, the hospital dataset needs to be anonymized in a way not to be able to distinguish between the hospitals visited by the patients that belong to the same equivalence class as a result of the anonymization of the patient dataset. For example, p_1 and p_2 must be associated with the same set of hospitals, and so do p_3 and p_4. Given lineage information, one way to do so consists in generalizing the hospitals in a way not to be able to distinguish between the hospitals corresponding to $\{\mathtt{p}_1, \mathtt{p}_3\}$ and those corresponding to $\{\mathtt{p}_2, \mathtt{p}_4\}$. An example of generalization of the hospital dataset that achieves this is illustrated in Table 3. Notice that similar generalization is applied to the hospitals corresponding to the groups of patients $\{\mathtt{p}_5, \mathtt{p}_7\}$ and $\{\mathtt{p}_6, \mathtt{p}_8\}$.

While acceptable, there is a more effective manner in this case to anonymize the patient and hospital datasets that yields less generalization of the quasi-attributes, thereby reducing the information loss incurred by the anonymization. Indeed, we can exploit the fact that patients are grouped into input sets to guide

Table 3: Input and Output Provenance of `admittedTo` **where the Input and Output are 2-anonymized.**

2-anonymized Patient DataSet			
ID	**name**	birth	Lin
p_1	★	{1987,1990}	$\{\mathtt{r}_1, \mathtt{r}_2\}$
p_2	★	{1987,1990}	$\{\mathtt{r}_3, \mathtt{r}_4\}$
p_3	★	{1985,1989}	$\{\mathtt{r}_5, \mathtt{r}_6\}$
p_4	★	{1985,1989}	$\{\mathtt{r}_7, \mathtt{r}_8\}$
p_5	★	{1988,1992}	$\{\mathtt{r}_9, \mathtt{r}_{10}\}$
p_6	★	{1988,1992}	$\{\mathtt{r}_{11}, \mathtt{r}_{12}\}$
p_7	★	{1986,1995}	$\{\mathtt{r}_{13}, \mathtt{r}_{14}\}$
p_8	★	{1986,1995}	$\{\mathtt{r}_{15}, \mathtt{r}_{16}\}$

2-anonymized Hospital DataSet		
ID	hospital	Lin
h_1	{St Louis, St Anne}	$\{p_1, p_3\}$
h_2	{St Anton, St August}	
h_3	{St Louis, St Anne}	$\{p_2, p_4\}$
h_4	{St Anton, St August}	
h_5	{Holby, St James}	$\{p_5, p_7\}$
h_6	{Larib., St Mary}	
h_7	{Holby, St James}	$\{p_6, p_8\}$
h_8	{Larib., St Mary}	

Table 4: Input and Output Provenance of `admittedTo` **where the Input is 2-anonymized and the output does not need to be.**

Input Patient DataSet			
ID	**name**	birth	Lin
p_1	★	{1989,1990}	$\{\mathtt{r}_1, \mathtt{r}_2\}$
p_2	★	{1985,1987}	$\{\mathtt{r}_3, \mathtt{r}_4\}$
p_3	★	{1989,1990}	$\{\mathtt{r}_5, \mathtt{r}_6\}$
p_4	★	{1985,1987}	$\{\mathtt{r}_7, \mathtt{r}_8\}$
p_5	★	{1986,1992}	$\{\mathtt{r}_9, \mathtt{r}_{10}\}$
p_6	★	{1988,1995}	$\{\mathtt{r}_{11}, \mathtt{r}_{12}\}$
p_7	★	{1986,1992}	$\{\mathtt{r}_{13}, \mathtt{r}_{14}\}$
p_8	★	{1988,1995}	$\{\mathtt{r}_{15}, \mathtt{r}_{16}\}$

Output Hospital DataSet		
ID	hospital	Lin
h_1	St Louis	$\{p_1, p_3\}$
h_2	St Anton	
h_3	St Anne	$\{p_2, p_4\}$
h_4	St August	
h_5	Holby	$\{p_5, p_7\}$
h_6	Larib.	
h_7	St James	$\{p_6, p_8\}$
h_8	St Mary	

the anonymization process. In particular, we put sets of data records that are used as input to a module invocation within the same equivalence class. For example, the patients p_1 and p_3 are put within the same equivalence class. Using this approach, we obtain the 2-anonymized patient dataset illustrated in Table 4. Notice that by doing so, we actually do not need to anonymize the hospital dataset. Indeed, starting from the hospital dataset, we cannot single out any patient: the same set of hospitals are visited by 2 patients. The approach we have just described is more effective as far as information loss is concerned. For example, one would know that $\mathtt{p}_1$ and $\mathtt{p}_3$ visited St Louis and St Antonio. Using the previous approach (described in Table 3), we would infer less specific information: that $\mathtt{p}_1$ and $\mathtt{p}_3$ visited St Louis or St Anne, and St Antonio or St Augustine.

With the above consideration in mind, we revisit the definition of equivalence classes introduced in Section 2 by requiring equivalence classes to contain sets of data records that are used as input or generated as output of module invocations. We will also introduce the notion of $\mathtt{k-group}$ anonymity degree, which allows us to gracefully reason about k-anonymity for collection-oriented modules.

Definition 3.1 (Equivalence Classes - Revisited). Given a module $\mathtt{m}$, we say that the set $\{\mathtt{E1}^{\mathtt{m}}_{\mathtt{in}}, \ldots, \mathtt{En}^{\mathtt{m}}_{\mathtt{in}}\}$ (resp $\{\mathtt{E1}^{\mathtt{m}}_{\mathtt{out}}, \ldots, \mathtt{En}^{\mathtt{m}}_{\mathtt{out}}\}$) is a set of input (resp. output) equivalence classes for $\mathtt{m}$, and write: $\mathtt{prov^a(m).in} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{in}}, \ldots, \mathtt{E1}^{\mathtt{m}}_{\mathtt{in}}\}$ (resp. $\mathtt{prov^a(m).out} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{out}}, \ldots, \mathtt{E1}^{\mathtt{m}}_{\mathtt{out}}\}$) iff:
1)- The conditions in Definition 2.5 are satisfied.
2)- An input (resp. output) equivalence class $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$ (resp. $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{out}}$) contains entire sets of input sets (resp. output sets) of data records. That is, two data records that belong to the same input set (resp. output set) that was used (resp. generated) by the invocation of $\mathtt{m}$ in $\mathtt{prov(m).in}$ (resp. $\mathtt{prov(m).out}$) cannot belong to different input (resp. output) equivalence classes.

Definition 3.2 (k-group anonymity (kg)). We say that the input provenance $\mathtt{prov^a(m).in}$ (resp. output provenance $\mathtt{prov^a(m).out}$) of a module $\mathtt{m}$ is k-group anonymized using the k-group anonymity degree $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}}$ (resp. $\mathtt{kg}^{\mathtt{m}}_{\mathtt{o}}$) iff each equivalence class in $\mathtt{prov^a(m).in}$ (resp. $\mathtt{prov^a(m).out}$) contains at least $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}}$ input sets of data records (resp. $\mathtt{kg}^{\mathtt{m}}_{\mathtt{o}}$ output sets of data records)

Property 1. *Consider a module* $\mathtt{m}$ *with an identifier input associated with an anonymity degree* $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$ *(resp. identifier output with an anonymity degree* $\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}$*). And, let* $\mathtt{l}^{\mathtt{m}}_{\mathtt{i}}$ *(resp.* $\mathtt{l}^{\mathtt{m}}_{\mathtt{o}}$*) be the magnitude of the smallest input (resp. output) set of data records in* $\mathtt{prov(m).in}$ *(resp.* $\mathtt{prov(m).out}$*). k-group anonymyzing* $\mathtt{prov(m).in}$ *(resp.* $\mathtt{prov(m).out}$*) using the k-group anonymity degree* $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}} = \left\lceil \frac{\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}}{\mathtt{l}^{\mathtt{m}}_{\mathtt{i}}} \right\rceil$ *(resp.* $\mathtt{kg}^{\mathtt{m}}_{\mathtt{o}} = \left\lceil \frac{\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}}{\mathtt{l}^{\mathtt{m}}_{\mathtt{o}}} \right\rceil$*) yields input provenance* $\mathtt{prov^a(m).in}$ *(resp. output provenance* $\mathtt{prov^a(m).in}$*) that is k-anonymized using the anonymity degree* $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$ *(resp.* $\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}$*).*

Proof. An input equivalence class in $\mathtt{prov^a(m).in}$ contains at least $\left\lceil \frac{\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}}{\mathtt{l}^{\mathtt{m}}_{\mathtt{i}}} \right\rceil$ input sets of data records. Given that $\mathtt{l}^{\mathtt{m}}_{\mathtt{i}}$ is the magnitude of the smallest input set in $\mathtt{prov(m).in}$, we conclude that an input equivalence class in $\mathtt{prov^a(m).in}$ contains at least $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}} \cdot \mathtt{l}^{\mathtt{m}}_{\mathtt{i}}$ data records, which is equal to or greater than $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$. In other words, $\mathtt{prov^a(m).in}$ is k-anonymized using the k-anonymity degree of $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$. The same reasoning can be applied to show that the output provenance is k-anonymized using the degree of $\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}$. □

We are now ready to discuss the general case. To anonymize the input provenance and output provenance of a module $\mathtt{m}$ with an identifier input and quasi-identifier output, we start by k-group anonymizing its input provenance using the k-group anonymity degree of $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}} = \left\lceil \frac{\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}}{\mathtt{l}^{\mathtt{m}}_{\mathtt{i}}} \right\rceil$. This yields input provenance $\mathtt{prov^a(m).out}$ that is k-anonymized using the degree of $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$ (see Property 1). Because the data records in the output provenance act as quasi-identifying records for the data records in the input provenance, we also need to anonymize the output provenance. To do so, we partition $\mathtt{prov(m).out}$ into a set of output equivalence classes $\mathtt{prov^a(m).out}$. This is done by putting the output sets of data records, that correspond to input sets pertaining to the same input equivalence class in $\mathtt{prov^a(m).in}$, within the same output equivalence class in $\mathtt{prov^a(m).out}$. This way, an adversary cannot distinguish the data records in an input equivalence class by examining their corresponding output data records, since these belong to the same output equivalence class and as such have the same quasi-identifying attribute values.

The above solution is applicable to modules with quasi-identifier input and identifier output, by inverting the roles of the input and output used above.

3.2 Modules with Identifier Input and Identifier Output

Consider a module $\mathtt{m}$ with an identifier input and an identifier output. To anonymize the input provenance and output provenance of $\mathtt{m}$, we reason using the k-group anonymity degrees associated with the input and output of $\mathtt{m}$. Specifically, we distinguish the following cases:

Case 1: $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}} \geq \mathtt{kg}^{\mathtt{m}}_{\mathtt{o}}$. We k-group the input provenance using the k-group degree $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}}$. This yields k-anonymized input provenance with an anonymity degree of $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$ (according to Property 1). The output provenance is anonymized by partitioning it into a set of output equivalence classes $\mathtt{prov^a(m).out}$: output sets of data records, that correspond to input sets pertaining to the same input equivalence class in $\mathtt{prov^a(m).in}$, are put within the same output equivalence class in $\mathtt{prov^a(m).out}$.

An output equivalence contains the sets of data records that correspond to input sets of data records in the same input equivalence class. Given that an input equivalence class contains at least $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}}$ input sets of data records, it follows that an output equivalence class in $\mathtt{prov^a(m).out}$ contains at least $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}}$ output sets of data records. Given that $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}} \geq \mathtt{kg}^{\mathtt{m}}_{\mathtt{o}}$, it follows that $\mathtt{prov^a(m).out}$ is k-group anonymized using the k-group anonymity degree of

Table 5: Input and Output Provenance of `getPractitioners`.

Input Patient DataSet		
ID	name	birth
p_1	Facello	1953
p_2	Simmel	1964
p_3	Bamford	1959
p_4	Koblick	1954
p_5	Maliniak	1955
p_6	Preusig	1953
p_7	Zielinski	1957
p_8	Kalloufi	1958

Output Practitioner DataSet			
ID	name	birth	Lin
pr_1	Rosch	1996	$\{p_1, p_2\}$
pr_2	Bellone	1987	
pr_3	Gargeya	1993	
pr_4	Gubsky	1988	$\{p_3, p_4\}$
pr_5	Heyers	1985	
pr_6	Tokunaga	1991	
pr_7	Camarinopoulos	1995	$\{p_5, p_6\}$
pr_8	Miculan	1986	
pr_9	Birrer	1992	
pr_{10}	Keustermans	1999	$\{p_7, p_8\}$
pr_{11}	Mancunian	2001	
pr_{12}	Bond	1982	

$\mathtt{kg}_o^m$, which implies that $\mathtt{prov}^a(\mathtt{m}).\mathtt{out}$ is k-anonymized using the anonymity degree of $\mathtt{k}_o^m$ (see Property 1). Note also that data records in the same input (resp. output) equivalence class cannot be distinguished by examining their corresponding output (resp. input) data records. This is because the data records in a given input equivalence class will have their corresponding output data records in the same output equivalence class, and, therefore, cannot be distinguished by examining their quasi-identifying attribute values, and vice-versa.

As an example, consider a module, `getPractitioners`, that takes a set of patients and returns the set of practitioners that have examined those patients[2]. Table 5 illustrates the input provenance and the output provenance of `getPractitioners`. We omit the lineage information (Lin column) in the input provenance because it is not useful in the example. Consider that the input of `getPractitioners` is associated with $\mathtt{k}_i^{\mathtt{getPractitioners}} = 2$, and its output with $\mathtt{k}_o^{\mathtt{getPractitioners}} = 2$. Given that $\mathtt{l}_i^{\mathtt{getPractitioners}} = 2$ and $\mathtt{l}_o^{\mathtt{getPractitioners}} = 3$ (see Table 5), we have $\mathtt{kg}_i^{\mathtt{getPractitioners}} = \mathtt{kg}_o^{\mathtt{getPractitioners}} = 1$. The k-group anonymity degree for both input and output in this case is 1. Tables 6 shows the anonymized input and output provenance obtained using the solution we have just described. Notice that the resulting patient dataset is 2-anonymized and that the resulting practitioner dataset is 3-anonymized. Moreover, we cannot distinguish between the practitioners of the patients in the same input equivalence class, and, similarly, we cannot distinguish between the patients of the practitioners that belong to the same output equivalence class.

Case 2: $\mathtt{kg}_i^m < \mathtt{kg}_o^m$. We perform the same processing as in (case 1) by inverting the roles of the input and output.

Table 6: 2-anonymized Input and 3-anonymized Output Provenance of `getPractitioners`.

Input Patient DataSet		
ID	name	birth
p_1	⋆	{53, 64}
p_2	⋆	{53, 64}
p_3	⋆	{54, 59}
p_4	⋆	{54, 59}
p_5	⋆	{53, 55}
p_6	⋆	{53, 55}
p_7	⋆	{57, 58}
p_8	⋆	{57, 58}

Output Practitioner DataSet			
ID	name	birth	Lin
pr_1	⋆	{87, 93, 96}	$\{p_1, p_2\}$
pr_2	⋆	{87, 93, 96}	
pr_3	⋆	{87, 93, 96}	
pr_4	⋆	{85, 88, 91}	$\{p_3, p_4\}$
pr_5	⋆	{85, 88, 91}	
pr_6	⋆	{85, 88, 91}	
pr_7	⋆	{86, 92, 95}	$\{p_5, p_6\}$
pr_8	⋆	{86, 92, 95}	
pr_9	⋆	{86, 92, 95}	
pr_{10}	⋆	{82, 99, 01}	$\{p_7, p_8\}$
pr_{11}	⋆	{82, 99, 01}	
pr_{12}	⋆	{82, 99, 01}	

4 DATA PRIVACY OF WORKFLOW PROVENANCE

Given a workflow `w`, we seek to anonymize its provenance `prov(w)` by anonymizing the input provenance and output provenance of its constituent modules. In doing so, we can use the method presented in the previous section as is to anonymize the data used and generated by each module, in an independent fashion. Unfortunately, this solution may lead to a breach of privacy. Indeed, equivalence classes will be formed without consideration to lineage between data records output by given modules and the data records used to feed the succeeding modules within the workflow, which may lead to a privacy breach. The technical report [5] contains a detailed example that illustrates this case.

We, therefore, designed an algorithm that ensures that lineage information cannot be used by an adversary to uncover private information about individuals. For the purpose of the anonymization algorithm, we will be shortly presenting, we group the workflow modules into levels as illustrated in Figure 2. A module belongs to level `0` if it does not have a previous module. A module belongs to a level `i` where `i > 0`, if it has at least an incoming data link connected to a module in level `i − 1`, and it does not have any incoming data link connected to a module in level ≥ `i`.

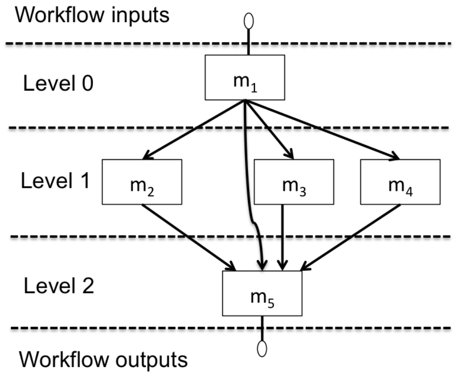

Fig. 2: Workflow levels.

Algorithm 1 Anonymize Workflow Provenance

```
Input: w, the workflow specification.
Input: Modules = {L_0, ..., L_k} // workflow modules grouped into levels (breadth)
       from the sink to the source.
           prov(w), the provenance of the workflow w.
           kg, group anonymity degree.
Output: Prov^a(w) // anonymized provenance
 1: for Level in Modules do
 2:     for m in Level do
 3:         if (m is the initial module) then
 4:             prov^a(m).in ← anonymizeInitialInput(m, prov(m).in, kg)
 5:             prov^a(m).out ← anonymizeOutput(m, prov(m).out, prov^a(m).in)
 6:         else
 7:             prov^a(m).in ← constructInputRecords(m, prec(m))
 8:             prov^a(m).out ← anonymizeOutput(m, prov(m).out, prov^a(m).in)
 9:         end if
10:     end for
11: end for
12: prov^a(w) ← ⋃_{m∈w.M} prof^a(m)
13: return prov^a(w)
```

To anonymize the provenance of a workflow `w`, Algorithm 1 takes as input the modules that compose the workflow organized into levels from the source to the sink, the provenance of the workflow `prov(w)`, as well as a group anonymity degree `kg` to be applied to the input provenance of the initial module of the workflow. (We will see later which k-group anonymity degree is used.) The algorithm examines the modules by level. For the first level composed of the initial module $\mathtt{m}_{\mathtt{init}}$, the operation `anonymizeInitialinput()` (line 4) anonymizes the input data of such module using the k-group degree `kg` and produces the set of equivalence classes $\mathtt{prov}^a(\mathtt{m}_{\mathtt{init}}).\mathtt{in}$. The output data records of the initial module are used to feed the operation

[2]A practitioner appears in the output set only if it has examined every patient in the input set.

anonymizeOutput(), which produces $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}_{\mathtt{init}}).\mathtt{out}$ (line 5). For a module m that belongs to a level other than the initial one, the algorithm starts by constructing the anonymized input data records by using the anonymized data records of the preceding output in the previous level using the constructInputRecords() operation (lines 7). The output data records of m are, then, anonymized by using the same grouping applied to its inputs to produce $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out}$ using the anonymizeOuput() operation (line 8). The algorithm terminates when the provenance of the module that belongs to the sink level is anonymized.

Having described how Algorithm 1 operates, we will describe in detail the operations used in the algorithm. We will also list the guarantees respected by each operation. Due to space limitations, the poofs of the guarantees can be found in the technical report [5]. Before proceeding to the presentation of the operations used in our algorithm, we start by defining the notion of lineage-related equivalence classes.

Definition 4.1 (Lineage-Related Equivalence Classes). Let E1 and E2 be two different equivalence classes. We say that E1 and E2 are lineage-related iff there are data records in E1 that (transitively) contributed through workflow executions to data records in E2, or vice-versa.

anonymizeInitialInput(m, prov(m).in, $\mathbf{k_g}$). This operation generates input equivalence classes for the initial module m: $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{in}}, \ldots, \mathtt{En}^{\mathtt{m}}_{\mathtt{in}}\}$, $n \geq 1$. Such equivalences classes are obtained by partitioning the input sets of data records in prov(m).in into groups, each containing at least $\mathtt{k_g}$ input sets. Each group gives rise to an equivalence class by masking the identifying attribute values of its data records and generalizing their quasi-identifying attribute values such that the data records in the group are indistinguishable w.r.t. their quasi-identifying attribute values. Details about the partitioning operation are presented later in Section 5.

<u>*Guarantees:*</u>

- $\mathbf{G_1}$: $\mathtt{prov}(\mathtt{m}_{\mathtt{init}})^{\mathtt{a}}.\mathtt{in}$ is $\mathtt{k_g}$ group anonymized.

anonymizeOutput(m, prov(m).out, $\mathbf{prov^a}$(m).in). Given anonymized input provenance $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{in}}, \ldots, \mathtt{En}^{\mathtt{m}}_{\mathtt{in}}\}$ of a module m, this operation generates anonymized output provenance of that module: $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{out}}, \ldots, \mathtt{En}^{\mathtt{m}}_{\mathtt{out}}\}$. To do so, for every input equivalence class $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$, a group $\mathtt{Gi}^{\mathtt{m}}_{\mathtt{out}}$ containing the output sets of data records in prov(m).out that are lineage dependent on input sets of data records in $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$, is constructed. Each group $\mathtt{Gi}^{\mathtt{m}}_{\mathtt{out}}$ gives rise to an output equivalence class $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{out}}$ by masking the identifying attribute values of the data records in $\mathtt{Gi}^{\mathtt{m}}_{\mathtt{out}}$ and by generalizing their quasi-identifying attribute values such that the data records in the group are indistinguishable w.r.t. their quasi-identifying attribute values.

<u>*Guarantees:*</u>

- $\mathbf{G_2}$: for every equivalence class $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$, anonymizeOutput() generates one lineage-related equivalence $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$ in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out}$.
- $\mathbf{G_3}$: anonymizeOutput() preserves k-group anonymity degree. That is, the number of output sets of data records in an output equivalence class $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$, generated by anonymizeOutput(), is equal to the number of input sets of data records in the input equivalence class $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ that is used as input to that operation.

ConstructInputRecords(m, prec(m)). construct input equivalence classes for m given the output equivalence classes of its preceding modules prec(m). We distinguish two cases:

Case 1: m is preceded by one module: prec(m) = {m′} There are data links connecting the output ports of m′ to the input ports of m. Given the anonymized output provenance $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}').\mathtt{out} = \{\mathtt{E1}^{\mathtt{m}'}_{\mathtt{out}}, \ldots, \mathtt{En}^{\mathtt{m}'}_{\mathtt{out}}\}$ of m′, this operation generates the anonymized input provenance of m, $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in} = \{\mathtt{E1}^{\mathtt{m}}_{\mathtt{in}}, \ldots, \mathtt{En}^{\mathtt{m}}_{\mathtt{in}}\}$, as follows: For every output equivalence class $\mathtt{Ei}^{\mathtt{m}'}_{\mathtt{out}}$, a group $\mathtt{Gi}^{\mathtt{m}}_{\mathtt{in}}$ is constructed containing the input sets of data records in prov(m).in that are lineage dependent on output sets of data records in $\mathtt{Ei}^{\mathtt{m}'}_{\mathtt{out}}$. The data records in $\mathtt{Gi}^{\mathtt{m}}_{\mathtt{in}}$ are, then, anonymized by masking their identifying attribute values, and by replacing their quasi-identifying attribute values, with the values used in their lineage-dependent data records in $\mathtt{Ei}^{\mathtt{m}'}_{\mathtt{out}}$. Thereby, each group $\mathtt{Gi}^{\mathtt{m}}_{\mathtt{in}}$ gives rise to an input equivalence class $\mathtt{Ei}^{\mathtt{m}}_{\mathtt{in}}$.

Case 2: m is preceded by multiple modules. Suppose that $\mathtt{prec}(\mathtt{m}) = \{\mathtt{m}_1, \mathtt{m}_2\}$. Cases, where a module has more than two preceding modules, are handled in the same manner. Given the anonymized output provenance $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}_1).\mathtt{out}$ of $\mathtt{m}_1$ and the anonymized output provenance $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}_2).\mathtt{out}$ of $\mathtt{m}_2$, the anonymized input provenance $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$ of m is obtained using the following process:

For pair $(\mathtt{Ei}^{\mathtt{m}_1}_{\mathtt{out}}, \mathtt{Ej}^{\mathtt{m}_2}_{\mathtt{out}})$ in $(\mathtt{prov}^{\mathtt{a}}(\mathtt{m}_1).\mathrm{out}, \mathtt{prov}^{\mathtt{a}}(\mathtt{m}_1).\mathrm{out})$, if there exists a data record d in prov(m).in that is lineage dependent on a data record in $\mathtt{Ei}^{\mathtt{m}_1}_{\mathtt{out}}$ and a data record in $\mathtt{Ej}^{\mathtt{m}_2}_{\mathtt{out}}$, a group $\mathtt{Gij}^{\mathtt{m}}_{\mathtt{in}}$ containing the data records in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$ that are lineage-dependent on data records from both equivalence classes $\mathtt{Ei}^{\mathtt{m}_1}_{\mathtt{out}}$ and $\mathtt{Ej}^{\mathtt{m}_2}_{\mathtt{out}}$, is constructed. The data records in $\mathtt{Gij}^{\mathtt{m}}_{\mathtt{in}}$ are anonymized, thereby giving rise to $\mathtt{Eij}^{\mathtt{m}}_{\mathtt{in}}$, as follows. The identifying attribute values of the data records in $\mathtt{Gij}^{\mathtt{m}}_{\mathtt{in}}$ are masked, and their quasi-identifying value attributes are replaced by the attribute values of their lineage-wise corresponding data records in $\mathtt{Ei}^{\mathtt{m}_1}_{\mathtt{out}}$ and $\mathtt{Ej}^{\mathtt{m}_2}_{\mathtt{out}}$.

<u>*Guarantees:*</u>

- $\mathbf{G_4}$: Using the operation ConstructInputRecords(), an output equivalence class of a module in prec(m) contributes to one lineage-related input equivalence class of m.
- $\mathbf{G_5}$: The operation constructInputRecord() preserves k-group anonymity. In other words, the number of input sets of data records in an input equivalence class $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ is equal to or greater than the number of output sets of data records in the corresponding output equivalence classes of the preceding modules.

4.1 Privacy Analysis

We will show, in this section, that an adversary cannot break k-anonymized workflow provenance that is obtained using Algorithm 1. In doing so, we need to show that: i)- The data records in the anonymized input provenance $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$ (resp. anonymized output provenance $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out}$) of every identifier input (resp. identifier output) of a module m in w, belong to equivalence classes of size $\geq k_i^m$ (resp. $\geq k_o^m$). ii)- The data records in an equivalence class in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$ (resp. $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out}$) cannot be distinguished by examining their lineage (i.e., by examining the data records they have been (transitively) generated from or the data records that they have (transitively) contributed to within workflow executions). To do so, we present in what follows a lemma and a theorem, each of which is accompanied by proof.

Lemma 1. An input equivalence class (resp. output equivalence class) of a given module:

(1) is lineage-related with at most one input equivalence class and one output equivalence class of a different module in the same workflow, and
(2) is lineage-related with one output equivalence class (resp. input equivalence class) of the same module, and
(3) is not lineage-related with any input equivalence class (resp. output equivalence class) of the same module.

Proof. We start by showing (1). Let $\mathtt{m}$ and $\mathtt{m}'$ be two modules in a workflow $\mathtt{w}$, and let $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$ be an input and output equivalence classes of $\mathtt{m}$. There are three possible cases:
a)- There is exist a dataflow path connecting $\mathtt{m}$ to $\mathtt{m}'$ in $\mathtt{w}$. For ease of exposition, we denote $\mathtt{m}$ by $\mathtt{m}_1$, and $\mathtt{m}'$ by $\mathtt{m}_n$, and, therefore, the data flow path connecting $\mathtt{m}$ to $\mathtt{m}'$, can be represented by the sequence $(\mathtt{m}_1, \ldots, \mathtt{m}_n)$, with $n \geq 2$. The sequence $(\mathtt{m}_1, \ldots, \mathtt{m}_n)$ denotes a dataflow, i.e. there are data links connecting the output ports of $\mathtt{m}_i$ to the input ports of $\mathtt{m}_{i+1}$ for $i \in [1, n-1]$. Given the guarantees G_2 and G_4, it follows that

- Every input equivalence class in $\mathtt{m}_i$ gives rise to one lineage-related output equivalence class of $\mathtt{m}_i$ for $i \in [1, n]$.
- Every output equivalence class in $\mathtt{m}_i$ gives rise to one lineage-related input equivalence class of $\mathtt{m}_{i+1}$ for $i \in [1, n-1]$.

Given that we consider acyclic workflow, a module cannot appear twice in the dataflow path $(\mathtt{m}_1, \ldots, \mathtt{m}_n)$, which allows us to conclude that every input or output equivalence class of $\mathtt{m}_1$ gives rise to one lineage-related input equivalence class for $\mathtt{m}_n$ and one lineage-related output equivalence class for the output of $\mathtt{m}_n$. Given that we use $\mathtt{m}_1$ to denote $\mathtt{m}_1$ and $\mathtt{m}_n$ to denote $\mathtt{m}'$, we can conclude that $\mathtt{m}'$ has one input equivalence class and one output equivalence class that are lineage-related with $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ (resp. $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$).
b)- There is exist a dataflow path connecting $\mathtt{m}'$ to $\mathtt{m}$ in the workflow. The same analysis in (a) allows to conclude that $\mathtt{m}'$ has one input equivalence class and one output equivalence class that are lineage-related with $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ (resp. $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$).
c)- There does not exist a data flow path connecting $\mathtt{m}$ to $\mathtt{m}'$, or vice-versa. Given that we consider a data-driven workflow execution module, the data records used and generated by $\mathtt{m}$ cannot possibly contribute to the data records used and generated by $\mathtt{m}'$, and vice-versa. It follows from Definition 4.1 that the equivalence classes associated with the input or output of $\mathtt{m}$ cannot be lineage-related to the equivalence classes associated with the input or output of $\mathtt{m}$.

(a), (b) and (c) allows us to conclude (1).
We now show (2). According to G_2, given a module $\mathtt{m}$ in a workflow $\mathtt{w}$, the operation `anonymizeOutput()` generates one lineage-related output equivalence class $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$ of $\mathtt{m}$, for every input equivalence class $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ of $\mathtt{m}$. Given that $\mathtt{w}$ is acyclic, it follows that $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ contains all the data records bound to the input of $\mathtt{m}$ that contributed to $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$, and the data records in $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ contribute to no data records bound to the output of $\mathtt{m}$, other than those in $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$. In other words, $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$ is the only output equivalence class of $\mathtt{m}$ that is lineage-related with $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$, and $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ is the only input equivalence class of $\mathtt{m}$ that is lineage-related with $\mathtt{E}^{\mathtt{m}}_{\mathtt{out}}$.
We now show (3). Given that we consider acyclic workflows, input data records (resp. output data records) of a given module cannot possibly contribute data records bound to the same input module (resp. module output). It follows then that an input equivalence class (resp. output equivalence class) of a given module is not lineage-related with any input equivalence class (resp. output equivalence class) of the same module. □

Theorem 4.2 (Soundness). *The workflow provenance generated for the provenance* $\mathtt{prov}(\mathtt{w})$ *of a workflow* $\mathtt{w}$ *by Algorithm 1 using as input the k-group anonymity degree:*

$$\mathtt{kg}^{\mathtt{max}} = \mathtt{max}\Big(\bigcup_{\mathtt{m_j} \in \mathtt{WF.modules}} \{\mathtt{kg}^{\mathtt{m_j}}_{\mathtt{i}}, \mathtt{kg}^{\mathtt{m_j}}_{\mathtt{o}}\}\Big) \qquad (1)$$

is k-annoymized.

Proof. To prove this theorem, we need to show: i) that every equivalence class in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$ of an identifier input of (resp. $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out}$ of an identifier output) of every module $\mathtt{m}$ in the workflow $\mathtt{w}$, contains at least $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$ (resp. $\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}$) data records, and ii) that data records in an equivalence class of the input or output of $\mathtt{m}$ cannot be distinguished by examining their lineage within the executions of $\mathtt{w}$ (see the problem statements in Section 2).

The input equivalence classes of the initial module in the workflow are generated using `anonymizeInitialInput`. According to G_1, the input equivalence classes generated by `anonymizeInitialInput` are k-grouped using the k-group anonymity degree $\mathtt{kg}^{\mathtt{max}}$. Such equivalence classes give rise to other equivalence classes by repeatedly applying the the operations `anonymizeOutput()` and `ConstructInputRecords()`. According to the guarantees G_3 and G_5 , such operations preserve k-group anonymity. It follows, then, that every equivalence class of an input (or output) of a module $\mathtt{m}$ that is generated by the algorithm contains a number of input (or output) sets that is equal to or greater than $\mathtt{kg}^{\mathtt{max}}$. In other words, the equivalence classes in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$ (resp. $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out}$) contain at least $\mathtt{kg}^{\mathtt{max}} \cdot \mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ data records (resp. $\mathtt{kg}^{\mathtt{max}} \cdot \mathtt{l}^{\mathtt{m}}_{\mathtt{out}}$ data records). Given that, $\mathtt{kg}^{\mathtt{max}} \cdot \mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ is equal to or greater than $\mathtt{kg}^{\mathtt{m}}_{\mathtt{i}} \cdot \mathtt{l}^{\mathtt{m}}_{\mathtt{i}}$, which by definition is equal to or greater than $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$. It follows that the equivalences classes in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$ contain at least $\mathtt{k}^{\mathtt{m}}_{\mathtt{i}}$ data records. Similarly, given that, $\mathtt{kg}^{\mathtt{max}} \cdot \mathtt{l}^{\mathtt{m}}_{\mathtt{o}}$ is equal to or greater than $\mathtt{kg}^{\mathtt{m}}_{\mathtt{o}} \cdot \mathtt{l}^{\mathtt{m}}_{\mathtt{o}}$, which by definition is equal to or greater than $\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}$. It follows that the equivalences classes in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out}$ contain at least $\mathtt{k}^{\mathtt{m}}_{\mathtt{o}}$ data records. Thereby, we have just shown (i).

We will now show (ii). Every input equivalence class $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{in}$ is, according to lemma 1, not lineage-related to any equivalence class of the same input, and it is lineage-related with at most one input equivalence class $\mathtt{E}^{\mathtt{m}'}_{\mathtt{in}}$ of any other module input in the workflow, and is lineage-related with at most one output equivalence class $\mathtt{E}^{\mathtt{m}'}_{\mathtt{in}}$ of any module in the workflow (including $\mathtt{m}$). The data records in any lineage-related input equivalence class $\mathtt{E}^{\mathtt{m}'}_{\mathtt{in}}$ or output equivalence class $\mathtt{E}^{\mathtt{m}'}_{\mathtt{out}}$) do not carry identifying attribute values and are indistinguishable w.r.t. their quasi-identifying attribute values. Therefore, an adversary is unable to distinguish between the data records in $\mathtt{E}^{\mathtt{m}}_{\mathtt{in}}$ of an input equivalence class of a module $\mathtt{m}$ by examining the data records of its lineage-related equivalence classes. The same reasoning can be applied to the output equivalences classes in $\mathtt{prov}^{\mathtt{a}}(\mathtt{m}).\mathtt{out}$. This implies that data records in an equivalence class $\mathtt{E}$ cannot be distinguished by examining the records of any of its lineage-related equivalence class $\mathtt{E}'$. And, by recursion, the data records in $\mathtt{E}'$ cannot be distinguished by examining the data records in the equivalence classes that are lineage-related with E'., etc. That is, the data records in an equivalence class cannot be distinguished by examining their lineage, thereby showing (ii). □

5 GROUPING OF DATA RECORD SETS

Consider that the initial module of a workflow took as input the following sets of records $D = D_1 \cup \cdots \cup D_n$, with $l = \min_{i \in [1 \cdots n]} |D_i|$. Consider now that the target k-group anonymity degree is kg, i.e., the target anonymity degree $k = kg * l$. If $k > 1$, i.e. $kg > 1$, then the method for anonymization that we have described so far states that the inputs sets in $D = D_1 \cup \cdots \cup D_n$ need to be grouped (unionied) into equivalence sets of a magnitude at least equal to k. In doing so, the method we presented does not specify which inputs sets in D to group together to form equivalence classes. A naïve solution to this problem would be to union all the datasets in D into a single group, i.e. a single equivalence class, and anonymize the quasi-identifying attributes of the data records accordingly. However, the records obtained using this approach are likely to be useless since the scientists will not be able to distinguish between any of the data records used as input to the module in question. A more desirable solution would, therefore, generate groups that have a small magnitude of at least k, and yet try to keep the magnitude of such groups as close as possible to k. We can formally define the above problem as follows[3].

Given sets of data records $D = D_1 \cup \cdots \cup D_n$, and an anonymity degree k, group the sets D_i, $i = 1 \ldots n$, into groups $G = G_1 \cup \cdots \cup G_m$, $m \le n$, such that:

(1) $|G_i| \ge k$, and
(2) $\max_{i=1 \ldots m}(|G_i|)$ is minimal.

The above problem can be viewed as a variant of the scheduling problem [29], in which the datasets D_i represent independent and non-preemptive jobs, and the cardinalities of such datasets represent jobs' lengths. There is a maximum of n machines. If a machine is used then its load must be greater or equal to k. The objective of such a scheduling problem is to minimize the makespan. To our knowledge, there does not exist any variant of the scheduling problem in the literature that meets the above criteria. We prove in the technical report that this is a strongly NP-hard problem, by reducing the *3-partition* problem to the above problem (see [5], page 14, for the proof).

Given that our problem is strongly NP-hard, we turn our attention to approximation algorithms. In particular, we devised the minimizeG integer problem (see below) to produce a good quality solution. x_{ij} is an integer that can takes the value 1 if the set D_i participates in the union that forms the group G_j, and 0, otherwise (Constraint C_4). y_j is an integer that can takes the value 1 if the group G_i contains at least one set in D, and 0, otherwise (Constraint C_5). $card_i$ represents the cardinality of the set D_i. Constraints (C_1) states that a set D_i must participate in the union of exactly one group. Constraint (C_2) specifies that a group G_j can have a cardinality of 0 (when y_j equals to 0), or a cardinality greater or equal to k (when y_j equals to 1). Constraint (C_3) specifies that the cardinalities of the groups $G_1, \ldots, G_n$ is smaller than a variable Z, which represents the makespan. The objective of the integer program is, therefore, to minimize the value of Z. Constraints (C_6) states that y_j is definitely equal to 1 if x_{ij} is equal to 1. More specifically, if the set D_i has been affected to the group G_j (i.e., $x_{ij} = 1$), then the group G_j contains at least one set (i.e.$y_j = 1$).

$$
\begin{array}{llll}
\text{minimizeG} & Z & & \\
\text{subject to} & \sum_{j \in \{1,\cdots,n\}} x_{ij} = 1, & i = 1, \ldots, n & (C_1) \\
 & \sum_{i \in \{1,\cdots,n\}} card_i \cdot x_{ij} \ge k.y_j, & j = 1, \ldots, n & (C_2) \\
 & \sum_{i \in \{1,\cdots,n\}} card_i \cdot x_{ij} \le Z, & j = 1, \ldots, n & (C_3) \\
 & x_{ij} \in \{0,1\}, & i,j = 1, \ldots, n & (C_4) \\
 & y_j \in \{0,1\}, & j = 1, \ldots, n & (C_5) \\
 & y_j \ge x_{ij}, & i,j = 1, \ldots, n & (C_6)
\end{array}
$$

Notice that we need to invoke the minimiseG program only once per workflow to identify the way the input sets of the initial module are to be grouped. The output of the initial module, as well as the input and output of the remaining modules in the workflow, use groupings that are derived based on lineage information (see Algorithm 1, lines 4 – 8).

6 VALIDATION

We implemented the solution that we have described in this paper using Python 2.7. We used the COIN Branch and Cut solver (CBC) provided by the LP Modeler Pulp[4] for solving the integer program MinimizeG presented in Section 5.

6.1 Experimental Setup

There is no existing solution that we can utilize as a base solution for comparison. Nonetheless, the approach that we have described raises the question as to which parameters impact the quality of the provenance anonymized using our solution. The analysis of the k-group anonymity degree computed for a workflow (see Equation 1), which dictates the degree of generalization, i.e., information loss, to be applied to the provenance of a workflow, reveals that the quality of the provenance (level of generalization) can be influenced by the anonymity degrees and magnitudes of the sets of data records used and generated by the parameters (inputs or outputs) of the workflow's modules. Note that on the other hand, the same equation allows us to rule out the topology (structure) of the workflow as a possible influencing factor. Because of this, we focus in our experiment on assessing the impact that the difference in the anonymity degrees and the magnitudes of the sets associated with two module parameters, which we take w.l.o.g to be the input and output of a module, has on the quality of anonymized provenance.

To be able to control the parameters of our experiment, we implemented a python program that given l_{in}^m, l_{out}^m and a number of module invocations, automatically generates module provenance. The provenance identifies the data records that are automatically generated by our tool. Regarding the content of data records, we use the Adult dataset [14], a de facto benchmark for anonymization solutions.

To assess the quality of anonymized data, we used the average equivalence class size [19] and the discernability metric [19]. For conciseness sake, we focus in what follows on the average equivalence class size. Readers interested in examining the results for discernability are referred to [5].

The average equivalence class measures how well equivalence classes created by the anonymization do not exceed what is required by the anonymization degree k. It can be defined as follows: $AEC(DS^*) = \frac{|DS|}{|EQ(DS^*)| \cdot k}$
where $EQ(DS^*)$ represents the set of equivalence classes created as a result of anonymizing DS, i.e., $|EQ(DS^*)|$ is the number of equivalence classes created. k represents the k-anonymity degree

[3]The problem statement formulated in Section 2.3 contains this condition.

[4] https://pypi.org/project/PuLP

required. The best value of AEC is 1. It means that none of the equivalence classes created as a result of anonymization exceeds the required anonymity degree when performing the generalization. We chose AEC as a measure because it is a good indicator for the quality of the anonymized data, with respect to a minimum requirement that is set by the anonymity degree.

As well as examining the impact of the anonymity degree and magnitude of sets of data records on the quality of anonymized provenance (in Sections 6.2, 6.3 and 6.4), we assess the utility of anonymized workflow provenance by examining the degree to which they can be used for answering workflow provenance challenge queries using real-world workflows [23] (in Section 6.5), and assess the efficiency of our solution (in Section 6.6).

6.2 Impact of the Disparity of $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$, $\mathtt{k}^{\mathtt{m}}_{\mathtt{out}}$ on the Quality of Anonymization

Given the provenance of the module m, one would expect that disparity between $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{k}^{\mathtt{m}}_{\mathtt{out}}$, or more specifically between the ratios $\mathtt{kg}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{kg}^{\mathtt{m}}_{\mathtt{out}}$ have an impact on the quality of the obtained anonymized input and output datasets of m. Specifically, if $\mathtt{kg}^{\mathtt{m}}_{\mathtt{out}}$ is larger than $\mathtt{kg}^{\mathtt{m}}_{\mathtt{in}}$ then the inputs records of m will be grouped into equivalence classes that are beyond what is required by $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$ to meet $\mathtt{k}^{\mathtt{m}}_{\mathtt{output}}$. Thereby, the average equivalence class of the obtained anonymized input datasets is likely to suffer as a results. On the contrary, if $\mathtt{kg}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{kg}^{\mathtt{m}}_{\mathtt{out}}$ are close then one would expect that the average equivalence class for both the input and output anonymized datasets to be of good quality. To assess this intuition, we ran an experiment in which:
1) We generated the provenance of a module m that associates sets of input data records with sets of output data records. (Note that we ran our experiments using different numbers of module invocations, namely 50, 100, 200, 300, 400 and 500 module invocation. The results we obtained presented similar trends. We, therefore, focus on reporting on the results obtained for 100 module invocations.) We set $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{l}^{\mathtt{m}}_{\mathtt{out}}$ to the same value, viz. 1. Specifically, an input (resp. output) set of data records that are used or generated by m has a magnitude between 1 and 3 (resp. 1 and 4). (We did so to examine the interplay between $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{k}^{\mathtt{m}}_{\mathtt{out}}$. Later on, we report on an experiment that we ran to assess the impact of the magnitudes of the sets of data records and their variability.) **2)** We then set the value of $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$ to 2, and anonymized the input and output datasets using our method by varying the value of $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$ between 2 and 20.

We ran this experiment three times. Figures 3 illustrates the average of the AEC obtained. Notice that the AEC of the output dataset is close (if not equal) to 1, indicating that the quality of the anonymized dataset is optimal as far as the constructed equivalence classes are concerned. On the other hand, we observe that the AEC of the input dataset increases as the disparity between $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{k}^{\mathtt{m}}_{\mathtt{out}}$ increases. This confirms our initial observation.

6.3 Impact of the Disparity of $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$, $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ on the Quality of Anonymization

Another aspect that can impact the quality of the anonymized datasets is the difference between the anonymity degree and the magnitude of the smallest input (resp. output) set of data records. Without loss of generality, let us consider the input of a module m. If $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ is greater than the anonymity degree $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$, then the magnitude equivalence classes obtained as a result of anonymization will be greater than what is required by $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$, thereby impacting negatively the AEC. To empirically examine this aspect, we set

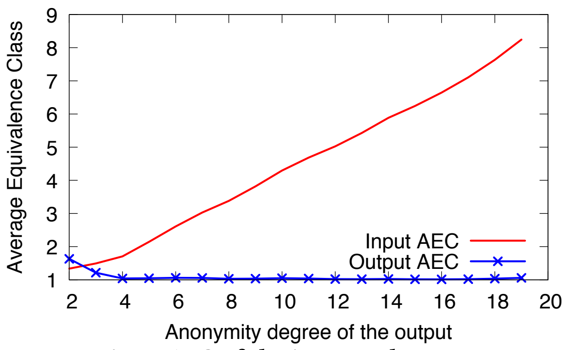

Fig. 3: AEC of the input and output.

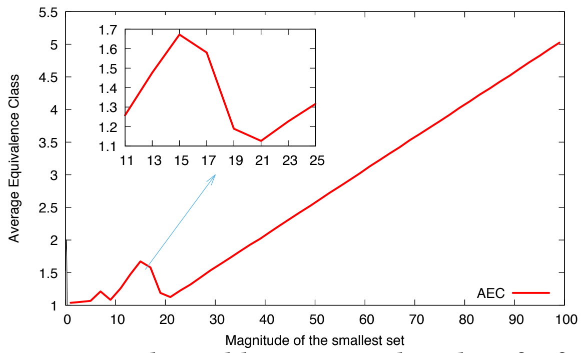

Fig. 4: AEC obtained by sweeping the value of $\mathtt{l}$ for $\mathtt{k} = 20$.

the anonymity degree $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}}$ to 20. We then varied the parameter $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ between 1 and 99, with a unit of 2, i.e., $[1, 3, \ldots, 97, 99]$. In particular, for a given value of $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$, the input sets generated for a module have a magnitude between $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}} + 3$. In other words, the input sets have a magnitude that is close to the value of $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$. We did so to factor out the impact that the variability in the magnitude of the input sets, which we will examine later on.

For each value of $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$, we generated the provenance for the module m (100 module invocation) and anonymized the obtained input dataset. We ran this experiment three times, and averaged the results, which are depicted in Figure 4 for the average equivalence class. The chart can be partitioned into two parts. The first part where $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ ranges from 1 to 20, and in the second part where it ranges from 20 to 100. In the first part, we notice that the AEC remains relatively close to 1 until it reaches the value of 15 and 17 where we notice an increase of the AEC 1.5. The AEC then decreases to values that are close to 1 for $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ values of 19 and 21. To explain this increase in the AEC, consider the case where $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}} = 15$. The magnitude of the input sets ranges between 15 and 18 according to the above experiment setting. Consequently, the magnitude of the sets obtained using the grouping ranges between 30 and 36. Indeed, an input set on its own has a magnitude lower than the required anonymity degree of 20, and two unionied input sets will definitely have a magnitude between 30 and 36, which is larger than the required anonymity degree. This explains the fact that the AEC value is close to 1.5. In the second phase, we observe that the value of the AEC grows linearly as the magnitude of the smallest set grows. This can be explained by the following. For values of $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ greater than 20, no set grouping is actually performed: the magnitude of the input set is greater than the required anonymity degree. The larger is the magnitude of $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$, the larger the disparity between $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{k}^{\mathtt{m}}_{\mathtt{in}} = 20$, and subsequently, the larger is the AEC.

6.4 Impact of the Disparity of the Size of Input (resp. Output) Sets

In the experiment that we ran this far, we assumed that the size of the input (resp. output) set of data records are close to $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ (resp. $\mathtt{l}^{\mathtt{m}}_{\mathtt{out}}$). We have examined the provenance of the workflows available in ProvBench[5], namely the workflow provenance collected the workflow systems Taverna and Wings (120 workflows). For each workflow and each of its modules, we computed $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ and $\mathtt{l}^{\mathtt{m}}_{\mathtt{out}}$. We then examined the variability of the magnitude of the input and output sets. This analysis revealed that in the majority of the cases the magnitude of the sets used and output by the modules that compose the workflow follows a uniform distribution. However, for an important proportion of the modules ($\approx$ 15%), we observed that the distribution is instead geometric in the sense that the input (resp. output) sets have a magnitude that is close to $\mathtt{l}^{\mathtt{m}}_{\mathtt{in}}$ (resp. $\mathtt{l}^{\mathtt{m}}_{\mathtt{out}}$).

Given the above results, we decided to empirically examine the variability of the magnitude of the parameter sets on the quality of the anonymization considering the two distributions. For the random uniform distribution, we used three distributions where the maximum magnitude of a set is 20, 50 and 100, respectively. Regarding the geometric distribution, we used three distributions with the probabilities of 30, 50 and 80, respectively.

We then ran an experiment in which we computed the AEC by varying the anonymity degree $\mathtt{k}_{\mathtt{in}}$ between 2 and 20. The results of the experiment for geometric distributions are illustrated in Figure 5, and those obtained for uniform distribution are illustrated in Figure 6. For geometric distribution, we observe that the higher the success probability, the better the AEC obtained. For example, the AEC quickly converges to the value of 1 when the success probability is equal to 0.8. On the other hand, the AEC converge to 1 only when the anonymity degree reaches 11 when the success probability is set to 0.3. That said, overall, geometric distribution delivers better results compared with uniform distribution: the AEC is much smaller. This can be explained by the fact that the variability in the magnitudes of the sets of data records is smaller in the case of the geometric distribution. And, the lower the variability of the magnitude of the data record sets, the better is the grouping of sets in the sense that it yields groups (i.e. equivalence classes) with magnitudes close to k, and therefore the better the AEC obtained (close to 1).

6.5 Assessing Utility Using Real Workflows

We assessed the degree to which anonymized provenance can be used to answer the following 3 queries that are representative of the queries defined by the workflow provenance challenge community [23].[6].

q_1 Find the workflow executions that led to a given record in the workflow results.

q_2 Find the input data records that contributed to a given data record in the workflow result.

q_3 Find the difference between two workflow execution.

For this experiment, we used 14 real-world Taverna workflows. The size of the workflows ranges from 3 modules to 24 modules, and have different structures patterns. We ran each workflow 30 times, and captured the provenance obtained using the Taverna workflow systems. We then anonymized the provenance by

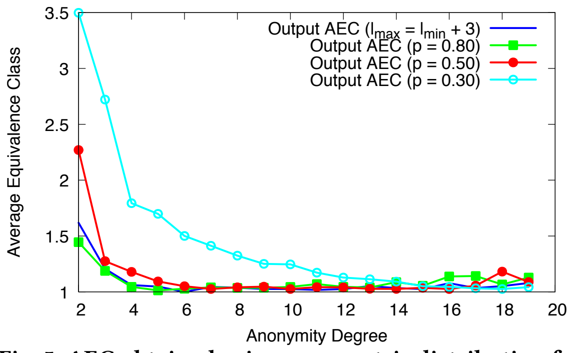

Fig. 5: AEC obtained using a geometric distribution for the magnitude of $\mathtt{l}$

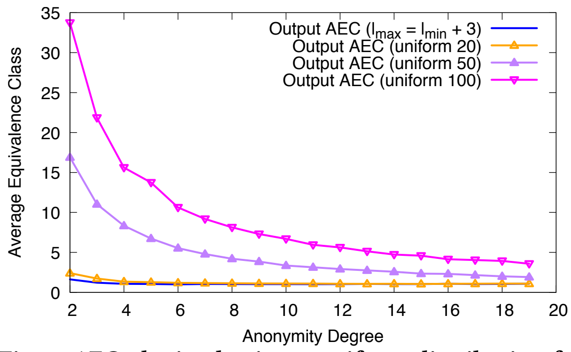

Fig. 6: AEC obtained using a uniform distribution for the magnitude of $\mathtt{l}$

varying the group anonymity degree $\mathtt{kg}^{\mathtt{max}}$ from 1 to 10, and examined whether queries of the form listed above can be answered using the anonymized provenance.

Regarding $\mathtt{q}_1$ and $\mathtt{q}_2$, a user is presented with anonymized workflow provenance, and as such cannot pinpoint a single data record in the results that can be used as input to $\mathtt{q}_1$ and $\mathtt{q}_2$. Instead, s/he chooses a (set) of data records that belong to the same equivalence class. As expected, the larger is the anonymity degree, the larger is the set of data records to be considered (see Table 7). Note that on the other hand, the query results obtained had 100% precision and recall, regardless of the value of group-anonymity degree used. This was possible thanks to the fact that our anonymization method preserves lineage across data records.

Table 7: Size of the data record sets used as input to $\mathtt{q}_1$ and $\mathtt{q}_2$ given $\mathtt{kg}^{\mathtt{max}}$, averaged over the 14 workflows.

$\mathtt{kg}^{\mathtt{max}}$	1	2	4	6	8	10
avg size of the set of data records	3	6	11	20	25	33

Regarding $\mathtt{q}_3$, the provenance challenge does not formally specify what it is meant by the difference of workflow executions [23]. That said, this question has later been thoroughly examined by Bao *et al.* [4]. They defined the difference between workflow executions of the same workflow specification using the edit distance which is the minimum number of edit operations that transform one provenance graph structure to the other. Using this definition, the edit distance between every pair of anonymized provenance graphs (of the 14 workflow specifications that we used) was the same as the edit distance computed using their counterpart original provenance graphs. This can be explained by the fact that our anonymization solution preserves the structure of the provenance graph as-is (since one of the requirements that we set is to preserve lineage information). Therefore, the

[5]https://github.com/provbench

[6]We could not use the provenance challenge queries as they are since they were specified for a single specific workflow on image processing.

original provenance graph of a given workflow specification is homomorphic to its anonymized counterpart.

This experiment has shown the utility of the workflow provenance anonymized using our solution since we were able to answer the three classes of queries. This evaluation exercise has also shown that for q_1 and q_2, the input of the query size (number of records) depends on the anonymity degree. Smaller anonymity degrees allow having smaller sets of data records that can be used for the queries, and vice-versa.

6.6 Efficiency

The only operation that is costly in the anonymization solution presented is the grouping of sets of data records, which we implemented using the integer program `minimizeG` (Section 5). Note, however, that such an operation is performed only once for the input of the initial module of the workflow. Indeed, the remaining parameters of the modules that compose the workflow apply the same grouping as the one applied to the input of the initial module. That said, we investigated the cost of such an operation to group n data sets, where n = 50, 100, 100, $\cdots$, 500. As expected, this experiment showed that time required increased as does the number of module invocations. Interestingly, the experiment also showed that the time required for performing the grouping of the sets of data records is primarily impacted by the distribution of the magnitude of the sets of data records to be grouped.

This experiment showed that using a uniform distribution, the range has little impact on the time required for grouping: it took between 16 and 18 seconds. On the other hand, it showed that sets that follow a geometric distribution with high success probability (50% and higher) require a considerable time compared to sets that follow a uniform distribution: it took up to 17 minutes. This can be explained by the fact that for a geometric distribution with high success probability, the majority of the sets have the same (or close) magnitudes. Therefore, many of the groupings that are explored by the integer program yield similar values for the objective function, hence recording little progress. The above results prompted us to develop an alternative solution when the magnitudes of the sets follow a geometric distribution with high success probability. The alternative solution we developed is simple, yet it group sets of data records in the orders of microseconds with a small impact on the AEC, which was higher by a margin of 0.03 on average compared with the situation in which we used our integer program `minimizeG`. More details can be found in the technical report [5].

7 CONCLUSIONS

We presented, in this paper, a solution for systematically anonymizing the provenance of collection-oriented workflows. Evaluation exercises allowed us to tease apart the aspects that impact the quality of the anonymization, namely the disparity between the anonymity degrees of the input and output sets of a module (or more generally the inputs and outputs of the modules that compose the workflow), the disparity between the anonymity degree and the magnitude of the sets of data records, and the distribution of the magnitudes of the record sets. We also examined the utility of the anonymized provenance using real-world workflows, and assessed the efficiency of our solution. In our ongoing work, we are investigating the applicability of our solution to anonymization techniques, other than k-anonymity, e.g., l-diversity and t-closeness [13]. We are also investigating the incorporation of vocabularies (hierarchies of concepts) to our solution. In the solution we presented, we substitute each quasi-identifier attribute value with a set containing the values that that attribute takes given a group (i.e. equivalence class) of data records. We will investigate how the use of vocabularies can be incorporated in our solution for generalizing the values of quasi-identifier attributes.

ACKNOWLEDGMENTS

I warmly thank Eun Jung Kim and Florian Sikora from the Paris-Dauphine University for the enlightening discussions on scheduling problems and integer programming. I would also like to thank the reviewers for their time and constructive comments, which helped improve the quality of the paper.

REFERENCES

[1] K. Abouelmehdi, A. B. Hssane, and H. Khaloufi. Big healthcare data: preserving security and privacy. *J. Big Data*, 5:1, 2018.
[2] AEPD. k-anonymity as a privacy measure. *Spanish Agency for Data Protection*, 2018. https://www.aepd.es/media/notas-tecnicas/nota-tecnica-kanonimidad-en.pdf.
[3] V. Ayala-Rivera, P. McDonagh, T. Cerqueus, and L. Murphy. A systematic comparison and evaluation of k-anonymization algorithms for practitioners. *Trans. Data Privacy*, 7(3):337–370, 2014.
[4] Z. Bao, S. C. Boulakia, S. B. Davidson, et al. Differencing provenance in scientific workflows. In *ICDE*, pages 808–819. IEEE, 2009.
[5] K. Belhajjame. On Anonymizing the Provenance of Collection-Based Workflows. Research report, Université Paris-Dauphine, PSL Research University, Jan. 2020. https://hal.inria.fr/hal-02430624/file/techreport.pdf.
[6] K. Belhajjame, N. Faci, Z. Maamar, V. A. Burégio, E. Soares, and M. Barhamgi. Privacy-preserving data analysis workflows for escience. In *EDBT/ICDT Workshops*, volume 2322 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.
[7] O. Biton, S. C. Boulakia, and S. B. Davidson. Zoom*userviews: Querying relevant provenance in workflow systems. In *VLDB*. ACM, 2007.
[8] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, volume 1973, pages 316–330. Springer, 2001.
[9] A. Chebotko, S. Chang, et al. Scientific workflow provenance querying with security views. In *WAIM*, pages 349–356. IEEE CS, 2008.
[10] C. Clifton and T. Tassa. On syntactic anonymity and differential privacy. *Transactions on Data Privacy*, 6(2):161–183, 2013.
[11] R. F. da Silva et al. Automating environmental computing applications with scientific workflows. In *e-Science*, pages 400–406. IEEE, 2016.
[12] S. B. Davidson, S. Khanna, T. Milo, et al. Provenance views for module privacy. In *PODS*, pages 175–186, 2011.
[13] J. Domingo-Ferrer and V. Torra. A critique of k-anonymity and some of its enhancements. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 990–993. IEEE, 2008.
[14] D. Dua and C. Graff. UCI machine learning repository, 2017.
[15] C. Dwork. Differential privacy. In *ICALP*, pages 1–12. Springer, 2006.
[16] R. Filgueira, A. Krause, M. P. Atkinson, et al. dispel4py: An agile framework for data-intensive escience. In *e-Science*, pages 454–464. IEEE, 2015.
[17] E. Griffis, P. Martin, and J. Cheney. Semantics and provenance for processing element composition in dispel workflows. In *WORKS*. ACM, 2013.
[18] J. Hidders, N. Kwasnikowska, J. Sroka, J. Tyszkiewicz, and J. V. den Bussche. DFL: A dataflow language based on petri nets and nested relational calculus. *Inf. Syst.*, 33(3):261–284, 2008.
[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, page 25. IEEE, 2006.
[20] T. M. McPhillips, S. Bowers, D. Zinn, and B. Ludäscher. Scientific workflow design for mere mortals. *FGCS*, 25(5):541–551, 2009.
[21] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *PODS*, pages 223–228. ACM, 2004.
[22] P. Missier, N. W. Paton, and K. Belhajjame. Fine-grained and efficient lineage querying of collection-based workflow provenance. In *EDBT*. ACM, 2010.
[23] L. Moreau, B. Ludäscher, I. Altintas, et al. Special issue: The first provenance challenge. *CCPE*, 20(5):409–418, 2008.
[24] M. E. Nergiz, C. Clifton, and A. E. Nergiz. Multirelational k-anonymity. *IEEE Trans. Knowl. Data Eng.*, 21(8):1104–1117, 2009.
[25] N. Park, M. Mohammadi, K. Gorde, et al. Data synthesis based on generative adversarial networks. *PVLDB*, 11(10):1071–1083, 2018.
[26] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, page 188. ACM Press, 1998.
[27] J. Sroka, J. Hidders, P. Missier, and C. A. Goble. A formal semantics for the taverna 2 workflow model. *J. Comput. Syst. Sci.*, 76(6):490–508, 2010.
[28] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *VLDB Endowment*, 1(1):115–125, 2008.
[29] F. Werner, L. Burtseva, and Y. Sotskov. *Algorithms for Scheduling Problems.* MDPI, 2018.
[30] K. Wolstencroft et al. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *NAR*, 41, 2013.