Provenance Views for Module Privacy

Susan B. Davidson
University of Pennsylvania
Philadelphia, PA, USA
susan@cis.upenn.edu

Sanjeev Khanna
University of Pennsylvania
Philadelphia, PA, USA
sanjeev@cis.upenn.edu

Tova Milo
Tel Aviv University
Tel Aviv, Israel
milo@cs.tau.ac.il

Debmalya Panigrahi
CSAIL, MIT
Massachusetts, MA, USA
debmalya@mit.edu

Sudeepa Roy
University of Pennsylvania
Philadelphia, PA, USA
sudeepa@cis.upenn.edu

ABSTRACT

Scientific workflow systems increasingly store provenance information about the module executions used to produce a data item, as well as the parameter settings and intermediate data items passed between module executions. However, authors/owners of workflows may wish to keep some of this information confidential. In particular, a *module* may be proprietary, and users should not be able to infer its behavior by seeing mappings between all data inputs and outputs.

The problem we address in this paper is the following: Given a workflow, abstractly modeled by a relation R, a privacy requirement Γ and costs associated with data. The *owner* of the workflow decides which data (attributes) to hide, and provides the *user* with a view R' which is the projection of R over attributes which have *not* been hidden. *The goal is to minimize the cost of hidden data while guaranteeing that individual modules are* Γ*-private.* We call this the `Secure-View` problem. We formally define the problem, study its complexity, and offer algorithmic solutions.

Categories and Subject Descriptors

H.2.0 [**Database Management**]: General—*Security, integrity, and protection*; H.2.8 [**Database Management**]: Database applications—*Scientific databases*

General Terms

Algorithms, Theory

Keywords

workflows, provenance, privacy, approximation

1. INTRODUCTION

The importance of data provenance has been widely recognized. In the context of scientific workflows, systems such as myGrid/Taverna [19], Kepler [4], and VisTrails [10] now capture and store provenance information, and a standard for provenance representation called the Open Provenance Model (OPM) [17] has been designed. By maintaining information about the module executions (processing steps) used to produce a data item, as well as the parameter settings and intermediate data items passed between module executions, the validity and reliability of data can be better understood and results be made reproducible.

However, authors/owners of workflows may wish to keep some of this provenance information private. For example, intermediate *data* within an execution may contain sensitive information, such as the social security number, a medical record, or financial information about an individual. Although users with the appropriate level of access may be allowed to see such confidential data, making it available to all users is an unacceptable breach of privacy. Beyond data privacy, a *module* itself may be proprietary, and hiding its description may not be enough: users without the appropriate level of access should not be able to infer its functionality by observing all inputs and outputs of the module. Finally, details of how certain modules in the workflow are connected may be proprietary, and therefore showing how data is passed between modules may reveal too much of the *structure* of the workflow. *There is thus an inherent tradeoff between the utility of provenance information and the privacy guarantees that authors/owners desire.*

While data privacy was studied in the context of statistical databases and ideas related to structural privacy were dealt with in the context of workflow views, module privacy has not been addressed yet. Given the importance of the issue [7], this paper therefore focuses on the problem of preserving the privacy of *module functionality*, i.e. the mapping between input and output values produced by the module (rather than the actual *algorithm* that implements it).

Abstracting the workflow models in [19, 4, 10], we consider a module to be a finite relation which takes a set I of input data (attributes), produces a set O of output data (attributes), and satisfies the functional dependency $I \longrightarrow O$. A row in this relation represents one execution of the module. In a *workflow*, n such data processing modules are connected in a directed acyclic multigraph (network), and jointly produce a set of final outputs from a set of initial inputs. Each module receives input data from one or more modules, or from the initial external input, and sends output data to one or more modules, or produces the final output. Thus a workflow can be thought of as a relation which is

PODS'11, June 13–15, 2011, Athens, Greece.

the input-output join of the constituent module relations. Each row in this relation represents a workflow execution, and captures the provenance of data that is produced during that execution. We call this the *provenance relation*.

To ensure the privacy of module functionality, we extend the notion of ℓ-diversity [15] to our network setting[1]: A module with functionality m in a workflow is said to be Γ-private if for every input x, the actual value of the output $m(x)$ is indistinguishable from $\Gamma - 1$ other possible values w.r.t. the visible data values in the provenance relation. This is achieved by carefully selecting a subset of data items and hiding those values in *all* executions of the workflow – i.e. by showing the user a *view* of the provenance relation for the workflow in which the selected data items (attributes) are hidden. Γ-privacy of a module ensures that even with arbitrary computational power and access to the view for all possible executions of workflow, an adversary can not guess the correct value of $m(x)$ with probability $> \frac{1}{\Gamma}$.

Identical privacy guarantees can be achieved by hiding different subsets of data. To reflect the fact that some data may be more valuable to the user than other data, we assign a *cost* to each data item in the workflow, which indicates the utility lost to the user when the data value is hidden. It is important to note that, due to *data sharing* (i.e. computed data items that are passed as input to more than one module in the workflow), hiding some data can be used to guarantee privacy for more than one module in the network.

The problem we address in this paper is the following: We are given a workflow, abstractly modeled by a relation R, a privacy requirement Γ and costs associated with data. An instance of R represents the set of workflow executions that have been run. The *owner* of the workflow decides which attributes to hide, and provides the *user* with a view R' which is the projection of R over the visible attributes. *The goal is to minimize the cost of hidden data while guaranteeing that individual modules are Γ-private.* We call this the secure-view problem. We formally define the problem, study its complexity, and offer algorithmic solutions.

Contributions. Our first contribution is to formalize the notion of Γ-privacy of a private module when it is a standalone entity (*standalone privacy*) as well as when it is a component of a workflow interacting with other modules (*workflow privacy*). For standalone modules, we then analyze the computational and communication complexity of obtaining a minimal cost set of input/output data items to hide such that the remaining, visible attributes guarantee Γ-privacy (a *safe subset*). We call this the *standalone secure-view* problem.

Our second set of contributions is to study workflows in which all modules are *private*, i.e. modules for which the user has no a priori knowledge and whose behavior must be hidden. For such *all-private workflows*, we analyze the complexity of finding a minimum cost set of data items in the workflow, as a whole, to hide such that the remaining visible attributes guarantee Γ-privacy for all modules. We call this the *workflow secure-view* problem. Although the privacy of a module within a workflow is inherently linked to the workflow topology and functionality of other modules, we are able to show that guaranteeing workflow secure-views in this setting essentially reduces to implementing the standalone privacy requirements for each module. We then study two variants of the workflow secure-view problem, one in which module privacy is specified in terms of attribute sets (*set constraints*) and one in which module privacy is specified in terms of input/output cardinalities (*cardinality constraints*). Both variants are easily shown to be NP-hard, and we give poly-time approximation algorithms for these problems. While the cardinality constraints version has an linear-programming-based $O(\log n)$-approximation algorithm, the set constraints version is much harder to approximate. However, both variants becomes more tractable when the workflow has *bounded data sharing*, i.e. when a data item acts as input to a small number of modules. In this case a constant factor approximation is possible, although the problem remains NP-hard even without any data sharing

Our third set of contributions is in *general workflows*, i.e workflows which contain private modules as well as modules whose behavior is known (*public* modules). Here we show that ensuring standalone privacy of private modules no longer guarantees their workflow privacy. However, by making some of the public modules private (*privatization*) we can attain workflow privacy of all private modules in the workflow. Since privatization has a cost, the optimization problem, becomes much harder: Even without data sharing the problem is $\Omega(\log n)$-hard to approximate. However, for both all-private and general workflows, there is an LP-based $\ell_{\max}$-approximation algorithm, where $\ell_{\max}$ is the length of longest requirement list for any module.

Related Work. *Workflow privacy* has been considered in [6, 12, 11]. In [6], the authors discuss a framework to output a *partial* view of a workflow that conforms to a given set of access permissions on the connections between modules and data on input/output ports. The problem of ensuring the *lawful use* of data according to specified privacy policies has been considered in [12, 11]. The focus of the work is a policy language for specifying relationships among data and module sets, and their properties relevant to privacy. Although all these papers address workflow privacy, the privacy notions are somewhat informal and no guarantees on the quality of the solution are provided in terms of privacy and utility. Furthermore, our work is the first, to our knowledge, to address module privacy rather than data privacy.

Secure provenance for workflows has been studied in [14, 5, 13]. The goal is to ensure that provenance information has not been forged or corrupted, and a variety of cryptographic and trusted computing techniques are proposed. In contrast, we assume that provenance information has not been corrupted, and focus on ensuring module privacy.

In [16], the authors study information disclosure in data exchange, where given a set of public views, the goal is to decide if they reveal any information about a private view. This does not directly apply to our problem, where the private elements are the $(\mathbf{x}, m(\mathbf{x}))$ relations. For example, if all $\mathbf{x}$ values are shown without showing any of the $m(\mathbf{x})$ values for a module m, then information is revealed in their setting but not in our setting.[2]

Privacy-preserving data mining has received considerable attention (see surveys [2, 22]). The goal is to hide individual data attributes while retaining the suitability of data for mining patterns. For example, the technique of *anonymiz-*

[1]In the Related Work, we discuss why a stronger notion of privacy, like differential privacy, is not suitable here.

[2]In contrast, it can be shown that showing all $m(\mathbf{x})$ values while hiding the $\mathbf{x}$'s, may reveal information in our setting.

ing data makes each record indistinguishable from a large enough set of other records in certain identifying attributes [21, 15]. Privacy preserving approaches were studied for *social networks* [3, 20] *auditing queries* [18] and in other contexts. Our notion of *standalone* module privacy is close to that of ℓ-diversity [15], in which the values of *non-sensitive attributes* are generalized so that, for every such generalization, there are at least ℓ different values of *sensitive attributes*. We extend this work in two ways: First, we place modules (relations) in a network of modules, which significantly complicates the problem, Second, we analyze the complexity of attaining standalone as well as workflow privacy of modules.

Another widely used technique is that of *data perturbation* where some noise (usually random) is added to the the output of a query or to the underlying database. This technique is often used in *statistical databases*, where a query computes some aggregate function over the dataset [8] and the goal is to preserve the privacy of data elements. In contrast, in our setting the private elements are $(\mathbf{x}, m(\mathbf{x}))$ pairs for a private module m and the queries are select-project-join style queries over the provenance relation rather than aggregate queries.

Privacy in *statistical databases* is typically quantified using *differential privacy*, which requires that the output distribution is *almost* invariant to the inclusion of any particular record (see survey [9] and the references therein). Although this is an extremely strong notion of privacy, *no* deterministic algorithm can guarantee differential privacy. Since provenance is used to ensure reproducibility of experiments (and therefore data values must be accurate), adding random noise to provenance information may render it useless. Thus standard mechanisms for differential privacy are unsuitable for our purpose. Our approach of outputting a safe view allows the user to know the name of all data items and the exact values of data that is visible. The user also does not lose any utility in terms of *connections* in the workflow, and can infer exactly which module produced which visible data item or whether two visible data items depend on each other.

Organization. Section 2 defines our workflow model and formalizes the notions of Γ-privacy of a module, both when it is standalone and when it appears in a workflow. The secure-view problem for standalone module privacy is studied in Section 3. Section 4 then studies the problem for workflows consisting only of private modules, whereas Section 5 generalizes the results to general workflows consisting of both public and private modules. Finally we conclude and discuss directions for future work in Section 6.

2. PRELIMINARIES

We start by introducing some notation and formalizing our notion of privacy. We first consider the privacy of a single module, which we call *standalone module privacy*. Then we consider privacy when modules are connected in a workflow, which we call *workflow module privacy*.

2.1 Modules and Relations

We model a module m with a set I of input variables and a set O of (computed) output variables as a relation R over a set of attributes $A = I \cup O$ that satisfies the functional dependency $I \rightarrow O$. In other words, I serves as a (not necessarily minimal) key for R. We assume that $I \cap O = \emptyset$ and will refer to I as the *input attributes* of R and to O as its *output attributes*.

We assume that the values of each attribute $a \in A$ come from a finite but arbitrarily large domain Δ_a, and let $\texttt{Dom} = \prod_{a \in I} \Delta_a$ and $\texttt{Range} = \prod_{a \in O} \Delta_a$ denote the *domain* and *range* of the module m respectively. The relation R thus represents the (possibly partial) function $m : \texttt{Dom} \rightarrow \texttt{Range}$ and tuples in R describe executions of m, namely for every $t \in R$, $\pi_O(\mathbf{t}) = m(\pi_I(\mathbf{t}))$. We overload the standard notation for projection, $\pi_A(R)$, and use it for a tuple $\mathbf{t} \in R$. Thus $\pi_A(\mathbf{t})$, for a set A of attributes, denotes the projection of $\mathbf{t}$ to the attributes in A.

EXAMPLE 1. *Figure 1 shows a simple workflow involving three modules m_1, m_2, m_3 with boolean input and output attributes; we will return to it shortly and focus for now on the top module m_1. Module m_1 takes as input two data items, a_1 and a_2, and computes $a_3 = a_1 \vee a_2$, $a_4 = \neg(a_1 \wedge a_2)$ and $a_5 = \neg(a_1 \oplus a_2)$. (The symbol $\oplus$ denotes XOR). The relational representation (functionality) of module m_1 is shown in Figure 1a as relation R_1, with the functional dependency $a_1 a_2 \longrightarrow a_3 a_4 a_5$. For clarity, we have added I (input) and O (output) above the attribute names to indicate their role.*

I		O		
a_1	a_2	a_3	a_4	a_5
0	0	0	1	1
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

(a) R_1: Functionality of m_1

a_1	a_2	a_3	a_4	a_5	a_6	a_7
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	1	0	0	1
1	1	1	0	1	1	1

(b) R: Workflow executions

$I \cap V$	$O \cap V$	
a_1	a_3	a_5
0	0	1
0	1	0
1	1	0
1	1	1

(c) $R_V = \pi_V(R_1)$

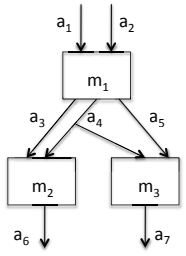

Figure 1: Module and workflow executions as relations

2.2 Standalone Module Privacy

Our approach to ensuring standalone module privacy, for a module represented by the relation R, will be to hide a carefully chosen subset of R's attributes. In other words, we will project R on a restricted subset V of attributes (called the *visible attributes* of R), allowing users access only to the view $R_V = \pi_V(R)$. The remaining, non visible, attributes of R are called *hidden attributes*.

We distinguish below two types of modules. (1) *Public modules* whose behavior is fully known to users when the name of the module is revealed. Here users have a priori knowledge about the full content of R and, even if given only the view R_V, they are able to fully (and exactly) reconstruct R. Examples include reformatting or sorting modules. (2) *Private modules* where such a priori knowledge does not exist, even if the name of the module is revealed. Here, the only information available to users, on the module's behav-

ior, is the one given by R_V. Examples include proprietary software, e.g. a genetic disorder susceptibility module[3].

Given a view (projected relation) R_V of a private module m, the *possible worlds* of m are all the possible full relations (over the same schema as R) that are consistent with R_V w.r.t the visible attributes. Formally,

DEFINITION 1. *Let m be a private module with a corresponding relation R, having input and output attributes I and O, resp., and let $V \subseteq I \cup O$. The set of* possible worlds *for R w.r.t. V, denoted $\texttt{Worlds}(R, V)$, consist of all relations R' over the same schema as R that satisfy the functional dependency $I \rightarrow O$ and where $\pi_V(R') = \pi_V(R)$.*

EXAMPLE 2. *Returning to module m_1, suppose the visible attributes are $V = \{a_1, a_3, a_5\}$ resulting in the view R_V in Figure 1c. For clarity, we have added $I \cap V$ (visible input) and $O \cap V$ (visible output) above the attribute names to indicate their role. Naturally, $R_1 \in \texttt{Worlds}(R_1, V)$. Figure 2 shows four additional sample relations $R_1^1, R_1^2, R_1^3, R_1^4$ in $\texttt{Worlds}(R_1, V)$, such that $\forall i \in [1, 4], \pi_V(R_1^i) = \pi_V(R_1) = R_V$. (Overall there are sixty four relations in $\texttt{Worlds}(R_1, V)$).*

a_1	a_2	a_3	a_4	a_5
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	1	0	1

(a) R_1^1

a_1	a_2	a_3	a_4	a_5
0	0	0	1	1
0	1	1	1	0
1	0	1	0	0
1	1	1	0	1

(b) R_1^2

a_1	a_2	a_3	a_4	a_5
0	0	1	0	0
0	1	0	0	1
1	0	1	0	0
1	1	1	0	1

(c) R_1^3

a_1	a_2	a_3	a_4	a_5
0	0	1	1	0
0	1	0	1	1
1	0	1	0	0
1	1	1	0	1

(d) R_1^4

Figure 2: $R_1^i \in \texttt{Worlds}(R_1, V)$, $i \in [1, 4]$

To guarantee privacy of a module m, the view R_V should ensure some level of uncertainly w.r.t the value of the output $m(\pi_I(\mathbf{t}))$, for tuples $t \in R$. To define this, we introduce the notion of Γ-standalone-privacy, for a given parameter $\Gamma \geq 1$. Informally, R_V is Γ-standalone-private if for every $t \in R$, the possible worlds $\texttt{Worlds}(R, V)$ contain at least Γ distinct output values that could be the result of $m(\pi_I(\mathbf{t}))$.

DEFINITION 2. *Let m be a private module with a corresponding relation R having input and output attributes I and O resp. Then m is* Γ-standalone-private *w.r.t a set of visible attributes V, if for every tuple $\mathbf{x} \in \pi_I(R)$, $|\text{OUT}_{\mathbf{x},m}| \geq \Gamma$, where $\text{OUT}_{\mathbf{x},m} = \{\mathbf{y} \mid \exists R' \in \texttt{Worlds}(R, V),\ \exists \mathbf{t}' \in R' \ s.t\ \mathbf{x} = \pi_I(\mathbf{t}') \wedge \mathbf{y} = \pi_O(\mathbf{t}')\}$.*

If m is Γ-standalone-private *w.r.t. V, then we will call V a* safe subset *for m and Γ.*

Γ-standalone-privacy implies that for *any* input the adversary cannot guess m's output with probability $> \frac{1}{\Gamma}$, even if the module is executed an arbitrary number of times.

[3]We discuss in Section 6 how partial prior knowledge can be handled by our approach.

EXAMPLE 3. *It can be verified that, if $V = \{a_1, a_3, a_5\}$ then for all $\mathbf{x} \in \pi_I(R_1)$, $|\text{OUT}_{\mathbf{x}}| \geq 4$, so $\{a_1, a_3, a_5\}$ is safe for m_1 and $\Gamma = 4$. As an example, from Figure 2, when $\mathbf{x} = (0, 0)$, $\text{OUT}_{\mathbf{x},m} \supseteq \{(0, \underline{0}, 1), (0, \underline{1}, 1), (1, \underline{0}, 0), (1, \underline{1}, 0)\}$ (hidden attributes are underlined). Also, hiding any two output attributes from $O = \{a_3, a_4, a_5\}$ ensures standalone privacy for $\Gamma = 4$, e.g., if $V = \{a_1, a_2, a_3\}$ (i.e. $\{a_4, a_5\}$ are hidden), then the input $(0, 0)$ can be mapped to one of $(0, \underline{0}, \underline{0}), (0, \underline{0}, \underline{1}), (0, \underline{1}, \underline{0})$ and $(0, \underline{1}, \underline{1})$; this holds for other assignments of input attributes as well. But, $V = \{a_3, a_4, a_5\}$ (i.e. when only input attributes are hidden) is not safe for $\Gamma = 4$: for any input $\mathbf{x}$, $\text{OUT}_{\mathbf{x},m} = \{(0, 1, 1), (1, 1, 0), (1, 0, 1)\}$, containing only three possible output tuples.*

There may be several safe subsets V for a given module m and parameter Γ. Some of the corresponding R_V views may be preferable to others, e.g. they provide users with more useful information, allow to answer more common/critical user queries, etc. Let $\overline{V} = (I \cup O) \setminus V$ denote the attributes of R that do not belong to the view. If $\mathtt{c}(\overline{V})$ denotes the penalty of hiding the attributes in $\overline{V}$, a natural goal is to choose a safe subset V that minimizes $\mathtt{c}(\overline{V})$. To understand the difficulty of this problem, we study a version of the problem where the cost function is additive: each attribute a has some penalty value $\mathtt{c}(a)$ and the penalty of hiding $\overline{V}$ is $\mathtt{c}(\overline{V}) = \Sigma_{a \in \overline{V}} \mathtt{c}(a)$. We call this optimization problem the *standalone* `Secure-View` *problem* and discuss it in Section 3.

2.3 Workflows and Relations

A workflow W consists of a set of modules $m_1, \cdots, m_n$, connected as a DAG (see, for instance, the workflow in Figure 1). Each module m_i has a set I_i of input attributes and a set O_i of output attributes. We assume that (1) for each module, its input and output attributes are disjoint, i.e. $I_i \cap O_i = \emptyset$, (2) the output attributes of distinct modules are disjoint, namely $O_i \cap O_j = \emptyset$, for $i \neq j$ (since each data item is produced by a unique module), and (3) whenever an output of a module m_i is fed as input to a module m_j the corresponding output and input attributes of m_i and m_j are the same. The DAG shape of the workflow guarantees that these requirements are not contradictory.

We model executions of W as a relation R over the set of attributes $A = \cup_{i=1}^{n}(I_i \cup O_i)$, satisfying the set of functional dependencies $F = \{I_i \rightarrow O_i : i \in [1, n]\}$. Each tuple in R describes an execution of the workflow W. In particular, for every $t \in R$, and every $i \in [1, n]$, $\pi_{O_i}(\mathbf{t}) = m_i(\pi_{I_i}(\mathbf{t}))$.

EXAMPLE 4. *Returning to the sample workflow in Figure 1, the input and output attributes of modules m_1, m_2, m_3 respectively are (i) $I_1 = \{a_1, a_2\}$, $O_1 = \{a_3, a_4, a_5\}$, (ii) $I_2 = \{a_3, a_4\}$, $O_2 = \{a_6\}$ and (iii) $I_3 = \{a_4, a_5\}$, $O_3 = \{a_7\}$. The underlying functional dependencies in the relation R in Figure 1b reflect the keys of the constituent modules, e.g. from m_1 we have $a_1a_2 \longrightarrow a_3a_4a_5$, from m_2 we have $a_3a_4 \longrightarrow a_6$, and from m_3 we have $a_4a_5 \longrightarrow a_7$.*

Note that the output of a module may be input to several modules. It is therefore possible that $I_i \cap I_j \neq \emptyset$ for $i \neq j$. We call this *data sharing* and define the degree of data sharing in a workflow as follows:

DEFINITION 3. *A workflow W is said to have* γ-bounded data sharing *if every attribute in W can appear in the left hand side of at most γ functional dependencies $I_i \rightarrow O_i$.*

In the workflow of our running example, $\gamma = 2$. Intuitively, if a workflow has γ-bounded data sharing then a data item can be fed as input to at most γ different modules. In the following sections we will see the implication of such a bound on the complexity of the problems studied.

2.4 Workflow Module Privacy

To define privacy in the context of a workflow, we first extend our notion of *possible worlds* to a workflow view. Consider the view $R_V = \pi_V(R)$ of a workflow relation R. Since the workflow may contain private as well as public modules, a possible world for R_V is a full relation that not only agrees with R_V on the content of the visible attributes, but is also consistent w.r.t the expected behavior of the public modules. In the following definitions, $m_1, \cdots, m_n$ denote the modules in the workflow W and F denotes the set of functional dependencies $I_i \rightarrow O_i$, $i \in [1, n]$ in the relation R.

DEFINITION 4. *The set of* possible worlds *for the workflow relation R w.r.t. V, denoted also* $\texttt{Worlds}(R, V)$*, consists of all the relations R' over the same attributes as R that satisfy the functional dependencies in F and where (1) $\pi_V(R') = \pi_V(R)$, and (2) for every public module m_i in W and every tuple $\mathbf{t}' \in R'$, $\pi_{O_i}(\mathbf{t}') = m_i(\pi_{I_i}(\mathbf{t}'))$.*

Note that when a workflow consists only of private modules, the second constraint does not need to be enforced. We call these *all-private workflows* and study them in Section 4. We then show in Section 5 that attaining privacy when public modules are also used is fundamentally harder.

We are now ready to define the notion of Γ-workflow-privacy, for a given parameter $\Gamma \geq 1$. Informally, a view R_V is Γ-workflow-private if for every tuple $t \in R$, and every private module m_i in the workflow, the possible worlds $\texttt{Worlds}(R, V)$ contain at least Γ distinct output values that could be the result of $m_i(\pi_{I_i}(\mathbf{t}))$.

DEFINITION 5. *A private module m_i in W is* Γ-workflow-private *w.r.t a set of visible attributes V, if for every tuple $\mathbf{x} \in \pi_{I_i}(R)$, $|\text{OUT}_{\mathbf{x},W}| \geq \Gamma$, where $\text{OUT}_{\mathbf{x},W} = \{\mathbf{y} \mid \exists R' \in \texttt{Worlds}(R, V)$, s.t., $\forall \mathbf{t}' \in R'\ \mathbf{x} = \pi_{I_i}(\mathbf{t}') \Rightarrow \mathbf{y} = \pi_{O_i}(\mathbf{t}')\}$.*

W is called Γ-private *if every private module m_i in W is Γ-workflow-private. If W (resp. m_i) is* Γ-private *(Γ-workflow-private) w.r.t. V, then we call V a* safe subset *for Γ-privacy of W (Γ-workflow-privacy of m_i).*

For simplicity, in the above definition we assumed that the privacy requirement of every module m_i is the same Γ. The results and proofs in this paper remain unchanged when different modules m_i have different privacy requirements Γ_i.

In the rest of the paper, for a set of visible attributes $V \subseteq A$, $\overline{V} = A \setminus V$ will denote the hidden attributes in the workflow. The following proposition is easy to verify, which will be useful later:

PROPOSITION 1. *If V is a safe subset for Γ-workflow-privacy of a module m_i in W, then any V' such that $V' \subseteq V$ (or, $\overline{V'} \supseteq \overline{V}$) also guarantees Γ-workflow-privacy of m_i.*

As we illustrate later, given a workflow W and a parameter Γ there may be several incomparable (in terms of set inclusion) safe subsets V for Γ-privacy of W. Our goal is to choose one that minimizes the penalty $\mathtt{c}(\overline{V}) = \Sigma_{a \in \overline{V}} \mathtt{c}(a)$ of the hidden attributes $\overline{V}$. This we call the *workflow* `Secure-View` *problem*, or simply the `Secure-View` problem. The candidates are naturally the maximal, in terms of set inclusion, safe sets V (and correspondingly the minimal $\overline{V}$s).

2.5 Complexity Classes and Approximation

In the following sections we will study the `Secure-View` problem: minimize cost of the hidden attributes that ensures that a workflow is Γ-private. We will prove that this problem is NP-hard even in very restricted cases and study polynomial time *approximation algorithms* as well as the *hardness of approximations* for different versions of the problem. We will use the following standard notions of approximation: an algorithm is said to be a *$\mu(n)$-approximation algorithm* for a given optimization problem, for some non-decreasing function $\mu(n) : \mathbb{N}^+ \rightarrow \mathbb{R}^+$, if for every input of size n it outputs a solution of value at most a multiplicative factor of $\mu(n)$ away from the optimum. An optimization problem is said to be *$\mu(n)$-hard to approximate* if a poly-time $\mu(n)$-approximation algorithm for the problem does not exist under standard complexity assumptions. In this paper we will use standard complexity assumptions of the form $NP \not\subseteq DTIME(n^{f(n)})$, where $f(n)$ is a poly-logarithmic or sub-logarithmic function of n and $DTIME$ represents deterministic time. For example, the hardness result in Theorem 5 says that there cannot be an $O(\log n)$-approximation algorithm unless all problems in NP have $O(n^{\log \log n})$-time deterministic exact algorithms. Finally, a problem is said to be *APX-hard* if there exists a constant $\epsilon > 0$ such that a $(1 + \epsilon)$-approximation in polynomial time would imply $P = NP$. If a problem is APX-hard, then the problem cannot have a *PTAS*, i.e, a $(1+\epsilon)$-approximation algorithm which runs in poly-time for all constant $\epsilon > 0$, unless $P = NP$.

3. STANDALONE MODULE PRIVACY

We start our study of workflow privacy by considering the privacy of a standalone module, which is the simplest special case of a workflow. Hence understanding it is a first step towards understanding the general case. We will also see that standalone-privacy guarantees of individual modules may be used as building blocks for attaining workflow privacy.

We analyze below the time complexity of obtaining (minimal cost) guarantees for standalone module privacy. Though the notion of Γ-standalone-privacy is similar to the well-known notion of ℓ-diversity [15], to the best of our knowledge the time complexity of this problem has not been studied.

Optimization problems and parameters. Consider a standalone module m with input attributes I, output attributes O, and a relation R. Recall that a visible subset of attributes V is called a *safe subset* for module m and privacy requirement Γ, if m is Γ-standalone-private w.r.t. V (see Definition 2). If each attribute $a \in I \cup O$ has cost $\mathtt{c}(a)$, *the standalone* `Secure-View` *problem* aims to find a safe subset V s.t. the cost of the hidden attributes, $\mathtt{c}(\overline{V}) = \sum_{a \in \overline{V}} \mathtt{c}(a)$, is minimized. The corresponding decision version will take a cost limit C as an additional input, and decide whether there exists a safe subset V such that $\mathtt{c}(\overline{V}) \leq C$.

One natural way of solving the optimization version of the standalone `Secure-View` problem is to consider all possible subsets $V \subseteq I \cup O$, check if V is safe, and return the safe subset V s.t. $\mathtt{c}(\overline{V})$ is minimized. This motivates us to define and study the simpler `Safe-View` *problem*, which takes a subset V as input and decides whether V is a safe subset.

To understand how much of the complexity of the standalone `Secure-View` problem comes from the need to consider different subsets of attributes, and what is due to the

need to determine the safety of subsets, we study the time complexity of standalone Secure-View, with and without access to *an oracle for the* Safe-View *problem*, henceforth called a Safe-View *oracle*. A Safe-View oracle takes a subset $V \subseteq I \cup O$ as input and answers whether V is safe. In the presence of a Safe-View oracle, the time complexity of the Safe-View problem is mainly due to the number of oracle calls, and hence we study the *communication complexity*. Without access to such an oracle, we also study the *computational complexity* of this problem.

In our discussion below, $k = |I| + |O|$ denotes the total number of attributes in the relation R, and N denotes the number of rows in R (i.e. the number of executions). Then $N \leq \prod_{a \in I} |\Delta_a| \leq \delta^{|I|} \leq \delta^k$ where Δ_a is the domain of attribute a and δ is the maximum domain size of attributes.

3.1 Lower Bounds

We start with lower bounds for the Safe-View problem. Observe that this also gives lower bounds for the standalone Secure-View problem without a Safe-View oracle. To see this, consider a set V of attributes and assume that each attribute in V has cost > 0 whereas all other attributes have cost zero. Then Safe-View has a positive answer for V iff the standalone Secure-View problem has a solution with cost $= 0$ (i.e. the one that only hides the attributes $\overline{V}$).

Communication complexity of Safe-View . Given a visible subset $V \subseteq I \cup O$, we show that deciding whether V is safe needs $\Omega(N)$ time. Note that just to read the table as input takes $\Omega(N)$ time. So the lower bound of $\Omega(N)$ does not make sense unless we assume the presence of a *data supplier* (we avoid using the term "oracle" to distinguish it from Safe-View oracle) which supplies the tuples of R on demand: Given an assignment $\mathbf{x}$ of the input attributes I, the data supplier outputs the value $\mathbf{y} = m(\mathbf{x})$ of the output attributes O. The following theorem shows the $\Omega(N)$ communication complexity lower bound in terms of the number of calls to the data supplier; namely, that (up to a constant factor) one indeed needs to view the full relation.

THEOREM 1. *(Safe-View Communication Complexity) Given a module m, a subset $V \subseteq I \cup O$, and a privacy requirement Γ, deciding whether V is safe for m and Γ requires $\Omega(N)$ calls to the data supplier, where N is the number of tuples in the relation R of m.*

PROOF SKETCH. This theorem is proved by a reduction from the *set-disjointness problem*, where Alice and Bob hold two subsets A and B of a universe U and the goal is decide whether $A \cap B \neq \emptyset$. This problem is known to have $\Omega(N)$ communication complexity where N is the number of elements in the universe. □

Computational Complexity of Safe-View : The above $\Omega(N)$ computation complexity of Safe-View holds when the relation R is given explicitly tuple by tuple. The following theorem shows that even when R is described implicitly in a succinct manner, there cannot be a poly-time (in the number of attributes) algorithm to decide whether a given subset V is safe unless $P = NP$.

THEOREM 2. *(Safe-View Computational Complexity) Given a module m with a poly-size (in $k = |I| + |O|$) description of functionality, a subset $V \subseteq I \cup O$, and a privacy requirement Γ, deciding whether V is safe w.r.t. m and Γ is co-NP-hard in k.*

PROOF SKETCH. The proof of this theorem works by a reduction from the UNSAT problem, where given a boolean CNF formula g on variables $x_1, \cdots, x_\ell$, the goal is to decide whether, for *all* assignments of the variables, g is not satisfied. Here given any assignment of the variables $x_1, \cdots, x_\ell$, $g(x_1, \cdots, x_\ell)$ can be evaluated in polynomial time, which simulates the function of the data supplier. □

Lower Bound of Standalone Secure-View with a Safe-View Oracle: Now suppose we have access to a Safe-View oracle, which takes care of the "hardness" of the Safe-View problem given in Theorems 1 and 2, in constant time. The oracle takes a visible subset $V \subseteq I \cup O$ as input, and answers whether V is safe for module m and privacy requirement Γ. The following theorem shows that the decision version of standalone Secure-View remains hard (i.e. not solvable in poly-time in the number of attributes):

THEOREM 3. *(Standalone Secure-View Communication Complexity, with Safe-View oracle) Given a Safe-View oracle and a cost limit C, deciding whether there exists a safe subset $V \subseteq I \cup O$ with cost bounded by C requires $2^{\Omega(k)}$ oracle calls, where $k = |I| + |O|$.*

PROOF SKETCH. The proof of this theorem involves a novel construction of two functions, m_1 and m_2, on ℓ input attributes and a single output attribute, such that for m_1 the minimum cost of a safe subset is $\frac{3\ell}{4}$ whereas for m_2 it is $\frac{\ell}{2}$ ($C = \frac{\ell}{2}$). In particular, for both m_1 and m_2, all subsets of size $< \frac{\ell}{4}$ are safe and all other subsets are unsafe, except that for m_2, there is exactly one special subset of size $\frac{\ell}{2}$ such that this subset and all subsets thereof are safe.

We show that for an algorithm using $2^{o(k)}$ calls, there always remains at least one special subset of size $\frac{\ell}{2}$ that is consistent with all previous answers to queries. Hence after $2^{o(k)}$ calls, if the algorithm decides that there is a safe subset with cost $\leq C$, we choose m to be m_1; on the other hand, if it says that there is no such subset, we set $m = m_2$. In both the cases the answer of the algorithm is wrong which shows that there cannot be such an algorithm distinguishing these two cases with $2^{o(k)}$ calls. □

3.2 Upper Bounds

The lower bound results given above show that solving the standalone Secure-View problem is unlikely in time sub-exponential in k or sub-linear in N. We now present simple algorithms for solving the Secure-View and Safe-View problems, in time exponential in k and polynomial in N.

First note that, with access to a Safe-View oracle, the standalone Secure-View problem can be easily solved in $O(2^k)$ time, by calling the oracle for all 2^k possible subsets and outputting the safe subset with minimum cost.

Without access to a Safe-View oracle, we first "read" relation R using N data supplier calls. Once R is available, the simple algorithm sketched below implements the Safe-View oracle (i.e. tests if a set V of attributes is safe) and works in time $O(2^k N^2)$: For a visible subset V, we look at all possible assignments to the attributes in $I \setminus V$. For each input value we then check if it leads to at least $\frac{\Gamma}{\prod_{a \in O \setminus V} |\Delta_a|}$ different values of the visible output attributes in $O \cap V$ (Δ_a is the domain of attribute a). This is a necessary and sufficient condition for guaranteeing Γ privacy: by all possible $\prod_{a \in O \setminus V} |\Delta_a|$ extensions of the output attributes, for each input, there will be Γ different possible output values.

We mention here also that essentially the same algorithms (with same upper bounds) can be used to output *all* safe attribute sets of a standalone module, rather than just one with minimum cost. Such exhaustive enumeration will be useful in the following sections.

Remarks. These results indicate that, in the worse case, finding a minimal-cost safe attribute set for a module may take time that is exponential in the number of attributes. Note, however, that the number of attributes of a single module is typically not large (often less than 10, see [1]), so the computation is still feasible. Expert knowledge of module designers, about the module's behavior and safe attribute sets may also be exploited to speed up the computation. Furthermore, a given module is often used in many workflows. For example, sequence comparison modules, like BLAST or FASTA, are used in many different biological workflows. We will see that safe subsets for individual modules can be used as building blocks for attaining privacy for the full workflow. The effort invested in deriving safe subsets for a module is thus amortized over all uses.

4. ALL-PRIVATE WORKFLOWS

We are now ready to consider workflows that consist of several modules. We first consider in this section workflows where all modules are *private* (called *all-private workflows*). Workflows with a mixture of private and public modules are then considered in Section 5.

As in Section 3, we want to find a safe visible subset V with minimum cost s.t. all the modules in the workflow are Γ-workflow-private w.r.t. V (see Definition 5). One option is to devise algorithms similar to those described for standalone modules in the previous section. However, the time complexity of those algorithms is now exponential in the *total number of attributes of all modules in the workflow* which can be as large as $\Omega(nk)$, n being the number of modules in the workflow and k the maximum number of attributes of a single module. To avoid the exponential dependency on n, the number of modules in the workflow,which may be large [1], and to exploit the safe attribute subsets for standalone modules, which may have been already computed, we attempt in this section to assemble workflow privacy guarantees out of standalone module guarantees. We first prove, in Section 4.1, that this is indeed possible. Then, in the rest of this section, we study the optimization problem of obtaining a safe view with minimum cost.

Let W be a workflow consisting of modules $m_1, \cdots, m_n$, where I_i, O_i denote the input and output attributes of m_i, $i \in [1, n]$, respectively. We use below R_i to denote the relation for the *standalone* module m_i. The relations $R = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$, with attributes $A = \bigcup_{1=1}^{n}(I_i \cup O_i)$, then describes the possible executions of W. Note that the projection $\pi_{I_i \cup O_i}(R)$ of R on $I_i \cup O_i$ is a subset of (but not necessarily equal to) the standalone module relation R_i.

In this section (and throughout the rest of the paper), for a set of visible attributes $V \subseteq A$, $\overline{V} = A \setminus V$ will denote the hidden attributes. Further, $V_i = (I_i \cup O_i) \cap V$ will denote the visible attributes for module m_i, whereas $\overline{V_i} = (I_i \cup O_i) \setminus V_i$ will denote the hidden attributes for m_i, for $i \in [1, n]$.

4.1 Standalone-Privacy vs. Workflow-Privacy

We show that if a set of visible attributes guarantees Γ-standalone-privacy for a module, then if the module is placed in a workflow where only a subset of those attributes is made visible, then Γ-workflow-privacy is guaranteed for the module in this workflow. In other words, in an all-private workflow, hiding the union of the corresponding hidden attributes of the individual modules guarantees Γ-workflow-privacy for all of them[4]. We formalize this next.

THEOREM 4. *Let W be an all-private workflow with modules $m_1, \cdots, m_n$. Given a parameter $\Gamma \geq 1$, let $V_i \subseteq (I_i \cup O_i)$ be a set of visible attributes w.r.t which m_i, $i \in [1, n]$, is Γ-standalone-private. Then the workflow W is Γ-private w.r.t the set of visible attributes V s.t. $\overline{V} = \bigcup_{i=1}^{n} \overline{V_i}$.*

Before we prove the theorem, recall that Γ-standalone-privacy of a module m_i requires that for every input $\mathbf{x}$ to the module, there are at least Γ *potential outputs* of $\mathbf{x}$ in the possible worlds $\texttt{Worlds}(R_i, V_i)$ of the standalone module relation R_i w.r.t. V_i; similarly, Γ-workflow-privacy of m_i requires at least Γ potential outputs of $\mathbf{x}$ in the possible worlds $\texttt{Worlds}(R, V)$ of the workflow relation R w.r.t. V. Since $R = R_1 \bowtie \cdots \bowtie R_n$, a possible approach to prove Theorem 4 may be to show that, whenever the hidden attributes for m_i are also hidden in the workflow W, any relation $R_i' \in \texttt{Worlds}(R_i, V_i)$ has a corresponding relation $R' \in \texttt{Worlds}(R, V)$ s.t. $R_i' = \pi_{I_i \cup O_i}(R')$. If this would hold, then for $\overline{V} = \bigcup_{i=1}^{n} \overline{V_i}$, the set of possible outputs, for any input tuple $\mathbf{x}$ to a module m_i, will remain unchanged.

Unfortunately, Proposition 2 below shows that the above approach fails. Indeed, $|\texttt{Worlds}(R, V)|$ can be significantly smaller than $|\texttt{Worlds}(R_i, V_i)|$ even for very simple workflows.

PROPOSITION 2. *There exist a workflow W with relation R, a module m_1 in W with (standalone) relation R_1, and a set of visible attributes V_1 that guarantees both Γ-standalone-privacy and Γ-workflow-privacy of m_1, such that the ratio of $|\texttt{Worlds}(R_1, V_1)|$ and $|\texttt{Worlds}(R, V_1)|$ is* doubly exponential *in the number of attributes of W.*

PROOF SKETCH. To prove the proposition, we construct a simple workflow with two modules m_1, m_2 connected as a chain. Both m_1, m_2 are one-one functions with k boolean inputs and k boolean outputs (e.g., assume that m_1 is an identity function, whereas m_2 complements each of its k inputs). The module m_1 gets initial input attributes I_1, produces $O_1 = I_2$ which is fed to m_2 as input, and m_2 produces final attribute set O_2. Let V_1 be an arbitrary subset of O_1 such that $|\overline{V_1}| = \log \Gamma$ (assume that Γ is a power of 2). It can be verified that, m_1 as a standalone module is Γ-standalone-private w.r.t. visible attributes V_1 and both m_1, m_2, being one-one modules, are Γ-workflow-private w.r.t. V_1.

We show that the one-one nature of m_1 and m_2 restricts the size of $\texttt{Worlds}(R, V_1)$ compared to that of $\texttt{Worlds}(R_1, V_1)$. Since both m_1 and m_2 are one-one functions, the workflow W also computes a one-one function. Hence any relation S in $\texttt{Worlds}(R, V_1)$ has to compute a one-one function as well. But when m_1 was standalone, any function consistent with V_1 could be in $\texttt{Worlds}(R_1, V_1)$. By a careful computation, the ratio can be shown to be doubly exponential in k. □

Nevertheless, we show below that for every input $\mathbf{x}$ of the module, the set of its possible outputs, in these worlds, is exactly the same as that in the original (much larger number of) module worlds. Hence privacy is indeed preserved.

[4]By Proposition 1, this also means that hiding any superset of this union would also be safe for the same Γ.

Recall that $\text{OUT}_{\mathbf{x},m_i}$ and $\text{OUT}_{\mathbf{x},W}$ denote the possible output for an input $\mathbf{x}$ to module m_i w.r.t. a set of visible attributes when m_i is standalone and in a workflow W respectively (see Definition 2 and Definition 5). The following lemma will be useful in proving Theorem 4.

LEMMA 1. *Let m_i be a standalone module with relation R_i, let $\mathbf{x}$ be an input to m_i, and let $V_i \subseteq (I_i \cup O_i)$ be a subset of visible attributes. If $\mathbf{y} \in \text{OUT}_{\mathbf{x},m_i}$ then there exists an input $\mathbf{x}' \in \pi_{I_i}(R_i)$ to m_i with output $\mathbf{y}' = m_i(\mathbf{x}')$ such that $\pi_{V_i \cap I_i}(\mathbf{x}) = \pi_{V_i \cap I_i}(\mathbf{x}')$ and $\pi_{V_i \cap O_i}(\mathbf{y}) = \pi_{V_i \cap O_i}(\mathbf{y}')$.*

The proof of Lemma 1 is simple and is in the full version; instead we illustrate the statement of the lemma with the module m_1 whose relation R_1 appears Figure 1a. Its visible portion (for visible attributes $V = \{a_1, a_3, a_5\}$) is given in Figure 1c. Consider the input $\mathbf{x} = (0, \underline{0})$ to m_1 and a candidate output $\mathbf{y} = (1, \underline{0}, 0)$ (hidden attributes are underlined). From Figure 2c, $\mathbf{y} \in \text{OUT}_{\mathbf{x},m_1}$. This is because there exists $\mathbf{x}' = (0, \underline{1})$, s.t. $\mathbf{y}' = m_1(\mathbf{x}') = (1, \underline{1}, 0)$, and, $\mathbf{x}, \mathbf{x}'$ and $\mathbf{y}, \mathbf{y}'$ have the same values of the visible (non-underlined) attributes. Note that $\mathbf{y}$ does not need to be the actual output $m_1(\mathbf{x})$ on $\mathbf{x}$ or even share the same values of the visible attributes (indeed, $m_1(\mathbf{x}) = (0, \underline{1}, 1)$).

However, in proving Theorem 4, our main technical tool is Lemma 2, which states that given a set of visible attributes V_i of a standalone module m_i, the set of possible outputs for *every* input $\mathbf{x}$ to m_i remains unchanged when m_i is placed in an all-private workflow, provided the corresponding hidden attributes $\overline{V_i}$ remains hidden in the workflow.

LEMMA 2. *Consider any module m_i and any input $\mathbf{x} \in \pi_{I_i}(R)$. If $\mathbf{y} \in \text{OUT}_{\mathbf{x},m_i}$ w.r.t. a set of visible attributes $V_i \subseteq (I_i \cup O_i)$, then $\mathbf{y} \in \text{OUT}_{\mathbf{x},W}$ w.r.t. $V_i \cup (A \setminus (I_i \cup O_i))$.*

We fix a module m_i, an input $\mathbf{x}$ to m_i and a candidate output $\mathbf{y} \in \text{OUT}_{\mathbf{x},m_i}$ for $\mathbf{x}$ w.r.t. visible attributes V_i. For $V = V_i \cup (A \setminus (I_i \cup O_i))$, $\overline{V} = A \setminus V = (I_i \cup O_i) \setminus V_i = \overline{V_i}$ (since, $V_i \subseteq I_i \cup O_i \subseteq A$). We will show that $\mathbf{y} \in \text{OUT}_{\mathbf{x},W}$ w.r.t. visible attributes V by showing the existence of a possible world $R' \in \texttt{Worlds}(R, V)$, for $\overline{V} = \overline{V_i}$, s.t. if $\pi_{I_i}(\mathbf{t}) = \mathbf{x}$ for some $\mathbf{t} \in R'$, then $\pi_{O_i}(\mathbf{t}) = \mathbf{y}$.

Two key tools used in the proof of Lemma 2 are *tuple flipping* and *function flipping*. Given a tuple $\mathbf{w}$ on a subset of attributes $P \subseteq A$, and two tuples $\mathbf{p}, \mathbf{q}$ on a subset of attributes $Q \subseteq A$, flipping $\mathbf{w}$ w.r.t. $\mathbf{p}, \mathbf{q}$ produces a tuple $\mathbf{z} = \text{FLIP}_{\mathbf{p},\mathbf{q}}(\mathbf{w})$ on P as follows: if $\mathbf{w}$ shares the same attribute as well as the same attribute value with $\mathbf{p}$ (resp. $\mathbf{q}$), replace the attribute value by that of $\mathbf{q}$ (resp. $\mathbf{p}$), otherwise copy the attribute value of $\mathbf{w}$ to $\mathbf{z}$. Flipping function m w.r.t tuples $\mathbf{p}, \mathbf{q}$ produces a function g with the same domain and range that of m, denoted by $\text{FLIP}_{m,\mathbf{p},\mathbf{q}}$, where for all input $\mathbf{w}$ to g, $g(\mathbf{w}) = \text{FLIP}_{\mathbf{p},\mathbf{q}}(m(\text{FLIP}_{\mathbf{p},\mathbf{q}}(\mathbf{w})))$ (flip the input, apply the original function m, again flip the output). Next we give a proof sketch of the lemma.

PROOF SKETCH OF LEMMA 2. Consider $\mathbf{x}, \mathbf{y}$ as given in the statement of Lemma 2. By Lemma 1, there are two tuples $\mathbf{x}', \mathbf{y}'$ such that $\pi_{V_i \cap I_i}(\mathbf{x}) = \pi_{V_i \cap I_i}(\mathbf{x}')$ and $\pi_{V_i \cap O_i}(\mathbf{y}) = \pi_{V_i \cap O_i}(\mathbf{y}')$. We replace the module sequence $\langle m_1, \cdots, m_n \rangle$ by $\langle g_1, \cdots, g_n \rangle$ by defining $g_j = \text{FLIP}_{m_j,\mathbf{p},\mathbf{q}}$, for all $j \in [1, n]$, where $\mathbf{p}$ (resp. $\mathbf{q}$) is formed by concatenating $\mathbf{x}, \mathbf{y}$ (resp. $\mathbf{x}', \mathbf{y}'$); in this process some modules may remain unchanged. We show that the relation R' produced by the join of the standalone relations of $\langle g_1, \cdots, g_n \rangle$ satisfies the properties: (1) $g_i(\mathbf{x}) = \mathbf{y}$, in other words, for all tuples $\mathbf{t} \in R'$ where $\pi_{I_i}(\mathbf{t}) = \mathbf{x}$, $\pi_{O_i}(\mathbf{t}) = \mathbf{y}$; and (2) $\pi_V(R) = \pi_V(R')$, i.e. the projections of R and R' on the visible attribute set are the same. Since every g_j is a function having domain and range the same as that of m_j, R' satisfies all functional dependencies $I_i \rightarrow O_i$, and therefore $R' \in \texttt{Worlds}(R, V)$. Together, these two properties show that $\mathbf{y} \in \text{OUT}_{\mathbf{x},W}$. □

It is important to note that the assumption of all-private workflow is crucial in proving Lemma 2 – if some of the modules m_j are public, we can not redefine them to g_j (the projection to the public modules should be unchanged - see Definition 5) and we may not get a member of $\texttt{Worlds}(R, V)$. We will return to this point in Section 5 when we consider workflows with a mixture of private and public modules.

Finally we complete the proof of Theorem 4 using Lemma 2:

PROOF OF THEOREM 4. We are given that each module m_i is Γ-standalone-private w.r.t. V_i, i.e., $|\text{OUT}_{\mathbf{x},m_i}| \geq \Gamma$ for all input $\mathbf{x}$ to m_i, for all modules m_i, $i \in [1, n]$ (see Definition 2). From Lemma 2, this implies that for all input $\mathbf{x}$ to all modules m_i, $|\text{OUT}_{\mathbf{x},W}| \geq \Gamma$ w.r.t $V' = V_i \cup (A \setminus (I_i \cup O_i))$. We already argued that, for this choice of V', $\overline{V'} = A \setminus V' = (I_i \cup O_i) \setminus V_i = \overline{V_i}$. Now, using Proposition 1, when the visible attributes set V is such that $\overline{V} = \bigcup_{i=1}^{n} \overline{V_i} \supseteq \overline{V_i} = \overline{V'}$, every module m_i is Γ-workflow-private. □

4.2 The `Secure-View` Problem

We have seen above that one can assemble workflow privacy guarantees out of the standalone module guarantees. Recall however that each individual module may have several possible safe attributes sets (see, e.g., Example 3). Assembling different sets naturally lead to solutions with different cost. The following example shows that assembling optimal (cheapest) safe attributes of the individual modules may not lead to an optimal safe attributes set for the full workflow. The key observation is that, due to data sharing, it may be more cost effective to hide expensive shared attributes rather than cheap non-shared ones (though later we show that the problem remains NP-hard even without data sharing).

EXAMPLE 5. *Consider a workflow with $n + 2$ modules, $m, m_1, \cdots, m_n, m'$. The module m gets an input data item a_1, with cost 1, and sends as output the same data item, a_2, with cost $1 + \epsilon$, $\epsilon > 0$, to all the m_i-s. Each m_i then sends a data item b_i to m' with cost 1. Assume that standalone privacy is preserved for module m if either its incoming or outgoing data is hidden and for m' if any of its incoming data is hidden. Also assume that standalone privacy is preserved for each m_i module if either its incoming or its outgoing data is hidden. As standalone modules, m will choose to hide a_1, each m_i will choose to hide the outgoing data item b_i, and m' will choose to hide any of the b_i-s. The union of the optimal solutions for the standalone modules has cost $n + 1$. However, a lowest cost solution for preserving workflow privacy is to hide a_2 and any one of the b_i-s. This assembly of (non optimal) solutions for the individual modules has cost $2 + \epsilon$. In this case, the ratio of the costs of the union of standalone optimal solutions and the workflow optimal solution is $\Omega(n)$.*

This motivates us to define the combinatorial optimization problem `Secure-View` (for workflow secure view), which generalizes the `Secure-View` problem studied in Section 3. The

goal of the Secure-View problem is to choose, for each module, a safe set of attributes (among its possible sets of safe attributes) s.t. together the selected sets yield a minimal cost safe solution for the workflow. We define this formally below. In particular, we consider the following two variants of the problem, trading-off expressibility and succinctness.

Set constraints. The possible safe solutions for a given module can be given in the form of a list of hidden attribute sets. Specifically, we assume that we are given, for each module m_i, $i \in [1,n]$, a list of pairs $L_i = \langle(\overline{I}_i^1, \overline{O}_i^1), (\overline{I}_i^2, \overline{O}_i^2) \dots (\overline{I}_i^{l_i}, \overline{O}_i^{l_i})\rangle$. Each pair $(\overline{I}_i^j, \overline{O}_i^j)$ in the list describes one possible safe (hidden) solution for m_i: $\overline{I}_i^j \subseteq I_i$ (resp. $\overline{O}_i^j \subseteq O_i$) is the set of input (output) attributes of m_i to be hidden in this solution. l_i (the length of the list) is the number of solutions for m_i that are given in the list, and we use below $\ell_{\max}$ to denote the length of the longest list, i.e. $\ell_{\max} = \max_{i=1}^n \ell_i$.

When the input to the Secure-View problem is given in the above form (with the candidate attribute sets listed explicitly) we call it *the* Secure-View *problem with set constraints.*

Cardinality constraints. Some modules may have many possible candidate safe attribute sets. Indeed, their number may be exponential in the number of attributes of the module. This is illustrate by the following two simple examples.

Example 6. *First observe that in any one-one function with k boolean inputs and k boolean outputs, hiding any k incoming or any k outgoing attributes guarantees 2^k-privacy. Thus listing all such subsets requires a list of length $\Omega(\binom{2k}{k}) = \Omega(2^k)$. Another example is majority function which takes $2k$ boolean inputs and produces 1 if and only if the number of one-s in the input tuple is $\geq k$. Hiding either $k+1$ input bits or the unique output bit guarantee 2-privacy for majority function, but explicitly listing all possible subsets again leads to exponential length lists.*

Note that, in both examples, the actual identity of the hidden input (resp. output) attributes is not important, as long as sufficiently many are hidden. Thus rather than explicitly listing all possible safe sets we could simply say what combinations of numbers of hidden input and output attributes are safe. This motivates the following variant of the Secure-View problem, called *the* Secure-View *problem with cardinality constraints*: Here for every module m_i we are given a list of pairs of numbers $L_i = \langle(\alpha_i^1, \beta_i^1) \dots (\alpha_i^{l_i}, \beta_i^{l_i})\rangle$, s.t. for each pair (α_i^j, β_i^j) in the list, $\alpha_i^j \leq |I_i|$ and $\beta_i^j \leq |O_i|$. The interpretation is that hiding any attribute set of m_i that consists of at least α_i^j input attributes and at least β_i^j output attributes, for some $j \in [1, \ell_i]$, makes m_i safe w.r.t the remaining visible attributes.

To continue with the above example, the list for the first module may consists of $(k,0)$ and $(0,k)$, whereas the list for the second module consists of $(k+1,0)$ and $(0,1)$.

It is easy to see that, for cardinality constraints, the lists are of size at most quadratic in the number of attributes of the given module (unlike the case of set constraints where the lists could be of exponential length)[5]. In turn, cardinality constraints are less expressive than set constraints that can specify arbitrary attribute sets. This will affect the complexity of the corresponding Secure-View problems.

[5]In fact, if one assumes that there is no redundancy in the list, the lists become of at most of linear size.

Problem Statement. Given an input in one of the two forms, a *feasible* safe subset V for the workflow, for the version with set constraints (resp. cardinality constraints), is such that for each module m_i $i \in [1,n]$, $\overline{V} \supseteq (\overline{I}_i^j \cup \overline{O}_i^j)$ (resp. $|\overline{V} \cap I_i| \geq \alpha_i^j$ and $|\overline{V} \cap O_i| \geq \beta_i^j$) for some $j \in [1, \ell_i]$. The goal of the Secure-View problem is to find a safe set V where $\mathtt{c}(\overline{V})$ is minimized.

4.3 Complexity results

We present below theorems which give approximation algorithms and matching hardness of approximation results of different versions of the Secure-View problem. The hardness results show that the problem of testing whether the Secure-View problem (in both variants) has a solution with cost smaller than a given bound is NP-hard even in the most restricted case. But we show that certain approximations of the optimal solution are possible. Theorem 5 and 6 summarize the results for the cardinality and set constraints versions, respectively. For space constraints we only sketch the proofs, details appear in the full version of the paper.

Theorem 5. ***(Cardinality Constraints)*** *There is an $O(\log n)$-approximation of the* Secure-View *problem with cardinality constraints. Further, this problem is $\Omega(\log n)$-hard to approximate unless* NP $\subseteq$ DTIME$(n^{O(\log\log n)})$*, even if the maximum list size $\ell_{\max} = 1$, each data has unit cost, and the values of α_i^j, β_i^j-s are 0 or 1.*

Proof Sketch. The proof of the hardness result in the above theorem is by a reduction from the set cover problem. The approximation is obtained by randomized rounding a carefully written linear program (LP) relaxation of this problem. A sketch is given below.

Our algorithm is based on rounding the fractional relaxation (called the LP relaxation) of the integer linear program (IP) for this problem presented in Figure 3.

$$\text{Minimize } \textstyle\sum_{b\in A} c_b x_b \quad \text{subject to}$$

$$\sum_{j=1}^{\ell_i} r_{ij} \geq 1 \quad \forall i \in [1,n] \tag{1}$$

$$\sum_{b\in I_i} y_{bij} \geq r_{ij}\alpha_i^j \quad \forall i \in [1,n],\ \forall j \in [1,\ell_i] \tag{2}$$

$$\sum_{b\in O_i} z_{bij} \geq r_{ij}\beta_i^j \quad \forall i \in [1,n],\ \forall j \in [1,\ell_i] \tag{3}$$

$$\sum_{j=1}^{\ell_i} y_{bij} \leq x_b, \quad \forall i \in [1,n], \forall b \in I_i \tag{4}$$

$$\sum_{j=1}^{\ell_i} z_{bij} \leq x_b, \quad \forall i \in [1,n], \forall b \in O_i \tag{5}$$

$$y_{bij} \leq r_{ij}, \quad \forall i \in [1,n],\ \forall j \in [1,\ell_i],\ \forall b \in I_i \tag{6}$$

$$z_{bij} \leq r_{ij}, \quad \forall i \in [1,n],\ \forall j \in [1,\ell_i],\ \forall b \in O_i \tag{7}$$

$$x_b, r_{ij}, y_{bij}, z_{bij} \in \{0,1\} \tag{8}$$

Figure 3: IP for Secure-View with cardinality constraints

Recall that each module m_i has a list $L_i = \{(\alpha_i^j, \beta_i^j) : j \in [1,\ell_i]\}$, a feasible solution must ensure that for each $i \in [1,n]$, there exists a $j \in [1,\ell_i]$ such that at least α_i^j input data and β_i^j output data of m_i are hidden.

In this IP, $x_b = 1$ if data b is hidden, and $r_{ij} = 1$ if at least α_i^j input data and β_i^j output data of module m_i are hidden. Then, $y_{bij} = 1$ (resp., $z_{bij} = 1$) if both $r_{ij} = 1$ and $x_b = 1$, i.e. if data b contributes to satisfying the input requirement α_i^j (resp., output requirement β_i^j) of module m_i. Let us first verify that the IP indeed solves the `Secure-View` problem with cardinality constraints. For each module m_i, constraint (1) ensures that for some $j \in [1, \ell_i]$, $r_{ij} = 1$. In conjunction with constraints (2) and (3), this ensures that for some $j \in [1, \ell_i]$, (i) at least α_i^j input data of m_i have $y_{bij} = 1$ and (ii) at least β_i^j output data of m_i have $z_{bij} = 1$. But, constraint (4) (resp., constraint (5)) requires that whenever $y_{bij} = 1$ (resp., $z_{bij} = 1$), data b be hidden, i.e. $x_b = 1$, and a cost of c_b be added to the objective. Thus the set of hidden data satisfy the privacy requirement of each module m_i and the value of the objective is the cost of the hidden data. Note that constraints (6) and (7) are also satisfied since y_{bij} and z_{bij} are 0 whenever $r_{ij} = 0$. Thus, the IP represents the `Secure-View` problem with cardinality constraints. It can be shown that simpler LP relaxations of this problem without some of the above constraints lead to unbounded and $\Omega(n)$ integrality gaps showing that an $O(\log n)$-approximation cannot be obtained from those LP relaxations (details are in the full version).

We round the fractional solution to the LP relaxation using Algorithm 1. For each $j \in [1, \ell_i]$, let I_{ij}^{min} and O_{ij}^{min} be the α_i^j input and β_i^j output data of m_i with minimum cost. Then, B_i^{min} represents $I_{ij}^{min} \cup O_{ij}^{min}$ of minimum cost.

Algorithm 1 Rounding algorithm of LP relaxation of the IP given in Figure 3,
Input: An optimal fractional solution $\{x_b | b \in A\}$,
Output: A safe subset V for Γ-privacy of W.

1: Initialize $B = \phi$.
2: For each attribute $b \in A$ (A is the set of all attributes in W), include b in B with probability $\min\{1, 16 x_b \log n\}$.
3: For each module m_i whose privacy requirement is not satisfied by B, add B_i^{min} to B.
4: Return $V = A \setminus B$ as the safe visible attribute.

The following lemma shows that step 2 satisfies the privacy requirement of each module with high probability:

LEMMA 3. *Let m_i be any module in workflow W. Then with probability at least $1 - 2/n^2$, there exists a $j \in [1, \ell_i]$ such that $|I_i^h| \geq \alpha_i^j$ and $|O_i^h| \geq \beta_i^j$.*

PROOF SKETCH. The LP solution returns a probability distribution on r_{ij}, and therefore on the pairs in list L_i. Let p be the index of the median of this distribution when list L_i is ordered by both α_i^j and β_i^j values, as described above. Our proof consists of showing that with probability $\geq 1 - 2/n^2$, $|I_i^h| \geq \alpha_{ip}$ and $|O_i^h| \geq \beta_{ip}$.

Note that since p is the median, the sum of y_{bij} over all incoming data of module v_i in the LP solution must be at least $\alpha_{ip}/2$ (from constraint (2)). Further, constraint (6) ensures that this sum is contributed to by at least $\alpha_{ip}/2$ different input data, and constraint (4) ensures that x_b for any input data b must be at least its contribution to this sum, i.e. $\sum_j y_{bij}$. Thus, at least $\alpha_{ip}/2$ different input data have a large enough value of x_b, and randomized rounding produces a good solution. An identical argument works for the output data of m_i. □

Since the above lemma holds for every module, by standard arguments, the $O(\log n)$-approximation follows.

We next show that the richer expressiveness of set constraints increases the complexity of the problem.

THEOREM 6. ***(Set Constraints)*** *The* `Secure-View` *problem with set constraints cannot be approximated to within a factor of $\ell_{\max}^{\epsilon}$ for some constant $\epsilon > 0$ (also within a factor of $\Omega(2^{\log^{1-\gamma} n})$ for all constant $\gamma > 0$) unless* NP $\subseteq$ DTIME($n^{\text{polylog } n}$). *The hardness result holds even when the maximum list size $\ell_{\max}$ is a (sufficiently large) constant, each data has unit cost, and the subsets $\overline{I}_i^j, \overline{O}_i^j$-s have cardinality at most 2. Finally, it is possible to get a factor $\ell_{\max}$-approximation in polynomial time.*

PROOF SKETCH. When we are allowed to specify arbitrary subsets for individual modules, we can encode a hard problem like *label-cover* which is known to have no polylogarithmic approximation given standard complexity assumptions. The corresponding approximation is obtained by an LP rounding algorithm which shows that a good approximation is still possible when the number of specified subsets for individual modules is not too large. □

The hardness proofs in the above two theorems use extensively data sharing, namely the fact that an output attribute of a given module may be fed as input to several other modules. Recall that a workflow is said to have γ-bounded data sharing if the maximum number of modules which takes a particular data item as input is bounded by γ. In real life workflows, the number of modules where a data item is sent is not very large. The following theorem shows that a better approximation is possible when this number is bounded.

THEOREM 7. ***(Bounded Data Sharing)*** *There is a $(\gamma + 1)$-approximation algorithm for the* `Secure-View` *problem (with both cardinality and set constraints) when the workflow has γ-bounded data sharing. On the other hand, the cardinality constraint version (and consequently also the set constraint version) of the problem remain APX-hard even when there is* no *data sharing (i.e. $\gamma = 1$), each data has unit cost, the maximum list size $\ell_{\max}$ is 2, and the values of α_i^j, β_i^j-s are bounded by 3.*

PROOF SKETCH. The APX-hardness in the above theorem is obtained by a reduction from vertex-cover in cubic graphs. This reduction also shows that the NP-completeness of this problem does not originate from data-sharing, and the problem is unlikely to have an exact solution even without any data sharing. The $\gamma + 1$-approximation is obtained by a greedy algorithm, which chooses the least cost attribute subsets for individual modules, and outputs the union of all of them. Since any attribute is produced by a unique module and is fed to at most γ modules, in any optimal solution, a single attribute can be used to satisfy the requirement of at most $\gamma + 1$ modules. This gives a $\gamma + 1$-approximation. Observe that when data sharing is not bounded, γ can be $\Omega(n)$ and this greedy algorithm will not give a good approximation to this problem. □

5. PUBLIC MODULES

In the previous section we restricted our attention to workflows where all modules are private. In practice, typical

workflows use also public modules. Not surprisingly, this makes privacy harder to accomplish. In particular, we will see below that it becomes harder to assemble privacy guarantees for the full workflow out of those that suffice for component modules. Nevertheless a refined variant of Theorem 4 can still be employed.

5.1 Standalone vs. Workflow Privacy (Revisited)

We have shown in Section 4.1 (Theorem 4) that when a set of hidden attributes guarantees Γ-standalone-privacy for a private module, then the same set of attributes can be used to guarantee Γ-workflow-privacy *in an all-private network*. Interestingly, this is no longer the case for workflows with public modules. To see why, consider the following example.

EXAMPLE 7. *Consider a private module m implementing a one-one function with k boolean inputs and k boolean outputs. Hiding any $\log\Gamma$ input attributes guarantees Γ-standalone-privacy for m even if all output attributes of m are visible. However, if m gets all its inputs from a public module m' that computes some constant function (i.e. $\forall \mathbf{x}, m'(\mathbf{x}) = \mathbf{a}$, for some constant $\mathbf{a}$), then hiding $\log\Gamma$ input attributes no longer guarantees Γ-workflow-privacy of m – this is because it suffices to look at the (visible) output attributes of m to know the value $m(\mathbf{x})$ for $x = \mathbf{a}$.*

In an analogous manner, hiding any $\log\Gamma$ output attributes of m, leaving all its input attributes visible, also guarantees Γ-standalone-privacy of m. But if m sends all its outputs to another public module m'' that implements a one-one invertible function, and whose output attributes happen to be visible, then for any input $\mathbf{x}$ to m, $m(\mathbf{x})$ can be immediately inferred using the inverse function of m''.

Modules that compute a constant function (or even one-one invertible function) may not be common in practice. However, this simple example illustrates where, more generally, the proof of Theorem 4 (or Lemma 2) fails in the presence of public modules: when searching for a possible world that is consistent with the visible attributes, one needs to ensure that the functions defined by the public modules remain unchanged. So we no longer have the freedom of freely changing the values of the hidden input (resp. output) attributes, if those are supplied by (to) a public module.

One way to overcome this problem is to "privatize" such problematic public modules, in the sense that the name of the public module is not revealed to users (either in the workflow specification or in its execution logs). Here we assume that once we rename a module the user loses all knowledge about it (we discuss other possible approaches in the conclusion). We refer to the public modules whose identity is hidden (resp. revealed) as *hidden* (*visible*) public modules. Observe that now, since the identity of the hidden modules is no longer known to the adversary, condition (2) in Definition 4 no longer needs to be enforced for them, and a larger set of possible words can be considered. Formally,

DEFINITION 6. *(Definition 4 revisited) Let P be a subset of the public modules, and, as before, let V be a set of the visible attributes. Then, the set of* possible worlds *for the relation R w.r.t. V and P, denoted $\texttt{Worlds}(R, V, P)$, consists of all relations R' over the same attributes as R that satisfy the functional dependencies in F and where (1) $\pi_V(R') = \pi_V(R)$, and (2) for every public module $m_i \in P$ and every tuple $\mathbf{t}' \in R'$, $\pi_{O_i}(\mathbf{t}') = m_i(\pi_{I_i}(\mathbf{t}'))$.*

The notion of Γ-privacy for a workflow W, with both private and public modules (w.r.t a set V of visible attributes and a set P of visible public modules) is now defined as before (Definition 5), except that the set of possible worlds that is considered is the refined one from Definition 6 above. Similarly, if W is Γ-*private* w.r.t. V and P, then we will call the pair (V, P) a *safe subset* for Γ-privacy of W.

We can now show that, by making visible only public modules whose input and output attribute values need not be masked, one can obtain a result analogous to Theorem 4. Namely, assemble the privacy guarantees of the individual modules to form privacy guarantees for the full workflow. Wlog., we will assume that $m_1, m_2, \cdots, m_K$ are the private modules and $m_{K+1}, \cdots, m_n$ are the public modules in W.

THEOREM 8. *Given a parameter $\Gamma \geq 1$, let $V_i \subseteq (I_i \cup O_i)$, $i \in [1, K]$, be a set of visible attributes w.r.t which the private module m_i is Γ-standalone-private. Then the workflow W is Γ-private w.r.t the set of visible attributes V and any set of visible public modules $P \subseteq \{m_{K+1}, \cdots, m_n\}$, s.t. $\overline{V} = \bigcup_{i=1}^{K} \overline{V_i}$ and all the input and output attributes of modules in P are visible and belong to V.*

PROOF SKETCH. The proof is similar to that of Thm. 4. Here analogous to Lemma 2, we can show that, if a public module $m_j, j \in [K+1, n]$ is redefined to g_j, then m_j is hidden. In other words, the visible public modules in P are never redefined and therefore condition (2) in Definition 6 holds. □

EXAMPLE 8. *Consider a chain workflow with three modules $m' \to m \to m''$, where m' is a public module computing a constant function, m is a private module computing a one-one function and m'' is another public module computing an invertible one-one function. If we hide only a subset of the input attributes of m, m' should be hidden, thus $P = \{m''\}$. Similarly, if we hide only a subset of the output attributes of m, m'' should be hidden. Finally, if we hide a combination of input and output attributes, both m', m'' should be hidden and in that case $P = \phi$.*

5.2 The `Secure-View` Problem (Revisited)

The `Secure-View` optimization problem in general workflows is similar to the case of all-private workflows, with an additional cost due to hiding (privatization) of public modules: when a public module m_j is hidden, the solution incurs a cost $\mathtt{c}(m_j)$. Following the notation of visible and hidden attributes, V and $\overline{V}$, we will denote the set of hidden public modules by $\overline{P}$. The total cost due to hidden public modules is $\mathtt{c}(\overline{P}) = \sum_{m_j \in \overline{P}} \mathtt{c}(m_j)$, and the total cost of a safe solution (V, P) is $\mathtt{c}(\overline{V}) + \mathtt{c}(\overline{P})$. The definition of the `Secure-View` problem, with cardinality and set constraints, naturally extends to this refined cost function and the goal is to find a safe solution with minimum cost. This generalizes the `Secure-View` problem for all-private workflows where $\overline{P} = \phi$ (and hence $\mathtt{c}(\overline{P}) = 0$).

Complexity Results In Section 4.3 we showed that the `Secure-View` problem has an $O(\log n)$-approximation in an all-private workflow even when the lists specifying cardinality requirements are $\Omega(n)$-long and when the workflow has arbitrary data sharing. But by a reduction from the label-cover problem, it can be shown that the cardinality constraints version in general workflows is $\Omega(2^{\log^{1-\gamma} n})$-hard to

approximate (for all constant $\gamma > 0$), and thus unlikely to have any polylogarithmic-approximation. In contrast, the approximation factor for the set constraints version remains the same and Theorem 6 still holds for general workflows by a simple modification to the proof. However, γ-bounded data sharing no longer give a constant factor approximation any more for a constant value of γ. By a reduction from the set-cover problem, it can be shown that the problem is $\Omega(\log n)$-hard to approximate even when the workflow has no data sharing, and when the maximum size of the requirement lists and the individual cardinality requirements in them are bounded by 1. Details of these results are deferred to the full version of the paper.

6. CONCLUSIONS

This paper proposes the use of provenance views for preserving the privacy of module functionality in a workflow. Our model motivates a natural optimization problem, `Secure-View` , which seeks to identify the smallest amount of data that needs to be hidden so that the functionality of every module is kept private. We give algorithms and hardness results that characterize the complexity of the problem.

In our analysis, we assume that users have two sources of knowledge about module functionality: the module name (identity) and the visible part of the workflow relation. Module names are informative for public modules, but the information is lost once the module name is hidden/renamed. Names of private modules are non-informative, and users know only what is given in the workflow view. However, if users have some additional prior knowledge about the behavior of a private module, we may hide their identity by renaming them, and then run our algorithms.

Our work suggests several promising directions for future research. First, a finer privacy analysis may be possible if one knows what *kind* of prior knowledge the user has on a private module, e.g. the distribution of output values for a specific input value, or knowledge about the types and names of input/output attributes (certain integers may be illegal social security numbers, certain character sequences are more likely to represent gene sequences than others, etc). Our definitions and algorithms currently assume that all data values in an attribute domain are equally possible, so the effect of knowledge of a possibly non-uniform prior distribution on input/output values should be explored. Second, some additional sources of user knowledge on functionality of public modules (e.g. types of attributes and connection with other modules) may prohibit hiding their functionality using privatization (renaming), and we would like to explore alternatives to privatization to handle public modules. A third direction to explore is an alternative model of privacy. As previously mentioned, standard mechanisms to guarantee differential privacy (e.g. adding random noise to data values) do not seem to work for ensuring module privacy w.r.t. provenance queries, and new mechanisms suitable to our application have to be developed. Other natural directions for future research include considering *non-additive cost functions*, in which some attribute subsets are more useful than others, efficiently handling *infinite or very large domains* of attributes, and exploring alternate objective functions, such as *maximizing utility* of visible data instead of minimizing the cost of hidden data.

Acknowledgements. S. B. Davidson, S. Khanna and S. Roy were supported in part by NSF-IIS Award 0803524; T. Milo was supported in part by NSF-IIS Award 1039376, the Israel Science Foundation, the US-Israel Binational Science Foundation and the EU grant MANCOOSI; and D. Panigrahi was supported in part by NSF-STC Award 0939370.

7. REFERENCES

[1] *www.myexperiment.org.*
[2] C. C. Aggarwal and P. S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer, 2008.
[3] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, pages 181–190, 2007.
[4] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In *Int. Conf. on Concept. Modeling*, pages 369–384, 2005.
[5] U. Braun, A. Shinnar, and M. Seltzer. Securing provenance. In *USENIX HotSec*, pages 1–5, 2008.
[6] A. Chebotko, S. Chang, S. Lu, F. Fotouhi, and P. Yang. Scientific workflow provenance querying with security views. In *WAIM*, pages 349–356, 2008.
[7] S. B. Davidson, S. Khanna, S. Roy, J. Stoyanovich, V. Tannen, Y. Chen, and T. Milo. Enabling privacy in provenance-aware workflow systems. In *CIDR*, 2011.
[8] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.
[9] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.
[10] J. Freire, C. T. Silva, S. P. Callahan, E. Santos, C. E. Scheidegger, and H. T. Vo. Managing rapidly-evolving scientific workflows. In *IPAW*, pages 10–18, 2006.
[11] Y. Gil, W. K. Cheung, V. Ratnakar, and K. kin Chan. Privacy enforcement in data analysis workflows. In *PEAS*, 2007.
[12] Y. Gil and C. Fritz. Reasoning about the appropriate use of private data through computational workflows. In *Intelligent Information Privacy Management*, pages 69–74, 2010.
[13] R. Hasan, R. Sion, and M. Winslett. Introducing secure provenance: problems and challenges. In *StorageSS*, pages 13–18, 2007.
[14] J. Lyle and A. Martin. Trusted computing and provenance: better together. In *TAPP*, page 1, 2010.
[15] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.
[16] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *SIGMOD*, pages 575–586, 2004.
[17] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson. The open provenance model: An overview. In *IPAW*, pages 323–326, 2008.
[18] R. Motwani, S. U. Nabar, and D. Thomas. Auditing sql queries. In *ICDE*, pages 287–296, 2008.
[19] T. 'Oinn *et al.* Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(1):3045–3054, 2003.
[20] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. In *PODS*, pages 107–116, 2009.
[21] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.
[22] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Rec.*, 33(1):50–57, 2004.