Privacy Issues in Scientific Workflow Provenance

[Position Paper]

Susan B. Davidson, Sanjeev Khanna, Sudeepa Roy
Department of Computer and Information Science, University of Pennsylvania
Philadelphia, PA 19104, USA
{susan, sanjeev, sudeepa}@cis.upenn.edu

Sarah Cohen Boulakia
Laboratoire de Recherche en Informatique
Université Paris-Sud
91405 Orsay cedex, France
cohen@lri.fr

ABSTRACT

A scientific workflow often deals with proprietary modules as well as private or confidential data, such as health or medical information. Hence providing exact answers to provenance queries over all executions of the workflow may reveal private information. In this paper we first study the potential privacy issues in a scientific workflow – *module privacy*, *data privacy*, and *provenance privacy*, and frame several natural questions: (i) can we formally analyze module, data or provenance privacy giving provable privacy guarantees for an unlimited/bounded number of provenance queries? (ii) how can we answer provenance queries, providing as much information as possible to the user while still guaranteeing the required privacy? Then we look at module privacy in detail and propose a formal model from our recent work in [11]. Finally we point to several directions for future work.

1. INTRODUCTION

Provenance in scientific workflows is of great interest, as evidenced by several recent workshops and surveys on the topic. A number of tools to capture, query, and manage provenance have also been developed in workflow systems such as myGrid/Taverna [24], Kepler [7], and VisTrails [15]. By maintaining information about the processing steps used to produce a data item, as well as the parameter settings and intermediate data items passed between steps, the validity and reliability of data can be better understood and results be made reproducible.

However, making complete provenance information available may raise privacy concerns. For example, intermediate data may contain sensitive information, such as the social security number or a medical record of an individual. Although using the data in an analysis may be acceptable, revealing the data itself would be a breach of privacy. A processing step (or *module*) may also be proprietary, and therefore allowing a user to see its inputs and outputs over a large number of executions could violate privacy by revealing its behavior. Finally, details of how modules in the workflow are connected may be private, and therefore showing how data is passed between modules would reveal the structure of the workflow. There is therefore a trade off between the amount of provenance information that can be revealed and the privacy guarantees of the data and modules comprising the workflow.

While the problem of answering user queries over a private source of information has been extensively studied (see, for example, [12], [25]), these results do not address core aspects of privacy in workflows. For instance, a commonly used idea is to introduce *noise* in the answer to a query. However, in scientific workflows the accuracy of revealed intermediate data is critical, and thus adding noise to the output of a module is undesirable. On the other hand, if intermediate data is revealed with any meaningful precision over a large number of executions of the workflow, the user may learn the behavior of private modules in the workflow.

Hence, addressing privacy in workflows requires a formal notion of privacy and suitable mechanisms for hiding provenance information to attain a required privacy guarantee. In this paper, we will call the part of the workflow or executions of the workflow that are revealed to the user a *privacy-preserving view*. Clearly, providing a view which consists only of the inputs to and outputs from the workflow, with all details of the workflow execution hidden in a "black box" composite module, guarantees any privacy requirements. However, it does not reveal any provenance information. This leads to a natural optimization problem which we call the *secure-view problem* - how to find a privacy-preserving view of the workflow which maximizes the provenance information revealed to the user. In our current research, we focus on obtaining a mathematical model of privacy in workflows with respect to different private components and finding exact or good approximate solutions to the secure-view problem.

Related Work.

Privacy-preserving data mining has received considerable attention in recent years (see surveys [1, 28], and the references therein). Here, the goal is to hide individual data attributes while retaining the suitability of the data for mining patterns. For example, the technique of *anonymizing data* makes each record indistinguishable from a large enough set of other records in certain identifying attributes [27, 21, 3]. Privacy preserving approaches have been studied for *social networks* [19, 4, 25, 9], *auditing queries* [23, 22] and in other contexts. 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 on the dataset [2, 12]. Privacy in statisti-

WANDS Indianapolis, IN, USA, June 6th, 2010

cal databases is often quantified using the framework of *differential privacy*, which requires that the output distribution is *almost* invariant to the inclusion of any particular record (see, for example, the surveys [13, 14]).

The problem of preserving privacy in a workflow has been considered in the papers [10, 17, 16]. In [10], 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. In particular, the authors define specifications for *port level security* (for data values produced or consumed by ports of modules), *channel level security* (for the presence of edges between modules) and *task level security* (for all input/output data of a module or sub-workflow). The problem of ensuring the lawful use of data according to some specified privacy policies has been considered in [17, 16], even when access control on the data has been enforced. The relationships among data and module sets, and their properties relevant to privacy, is specified by a policy language. Workflow systems incorporated with privacy awareness properties enable scientists to ensure data privacy throughout the data analysis process. However, to the best of our knowledge, a formal study of privacy issues specific to workflows with provable privacy guarantees has not yet been studied. Although the above papers address privacy issues in scientific workflows, the privacy notions are somewhat informal, and no guarantees on the quality of the solution are provided in terms of their privacy and utility.

Secure provenance for workflows has been studied [20, 8, 18]. Here 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. The secure provenance problem is quite different from that of privacy in workflow provenance, where the goal is to ensure that privacy is maintaned while still maximizing the user's utility in terms of provenance information.

Contribution and Organization.

The contribution of this paper is a discussion of privacy issues in workflow provenance, and a description of an approach for preserving module privacy in workflows with provable guarantees.

We start by describing our workflow model in Section 2, and discuss privacy issues in workflow provenance in Section 3. In Section 4, we highlight our recent work [11], which provides a formal definition of module privacy in a workflow and an approach to guarantee privacy while maximizing the provenance information revealed. We discuss several interesting and challenging open problems regarding privacy in workflows in Section 5, and conclude in Section 6.

2. WORKFLOW MODEL

We describe an abstract model that captures the workflows considered in most of workflow systems (e.g. MyGrid, Taverna, Vistrails, etc). A workflow can be thought of as a program, producing a set of *final output data* from a set of *initial input data* in an *execution*. Internally, a workflow system comprises a set of *modules* (i.e. programs) with input and output data ports. A module receives initial input data or data generated by other modules on its input ports and sends its output data or generates final data on its output ports; data items sent by a module to another module are called *intermediate data*. *Parameters* can also be specified for modules and, for uniformity, are considered to be initial inputs as well. In this paper, we assume that the connections form an *acyclic* graph on the modules, which is common in many of the aforementioned workflow systems [15, 24].

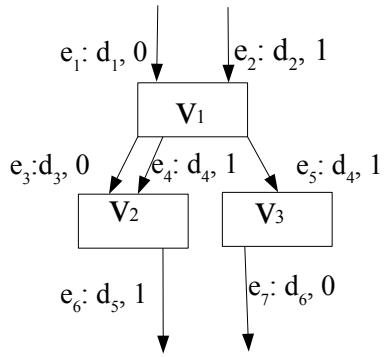

Figure 1: Example of a workflow execution

An example of a workflow is shown in Figure 1 which (for simplicity) uses boolean data and functions. There are two initial input data d_1 and d_2 on edges e_1 and e_2, respectively, d_5, d_6 on edges e_6 and e_7 are the final output data. The three modules in the workflow are v_1, v_2 and v_3. Their underlying functions f_1, f_2 and f_3 are defined as follows: (a) f_1 computes two intermediate data d_3 and d_4, where $d_3 = d_1$ and $d_4 = d_1 \vee d_2$, (b) $f_2 = d3 \oplus d_4$[1] and produces one output data d_5 which is 1 iff $d_3 \neq d_4$, (c) the other output data $d_6 = 1 - d_4$ is produced by f_3. Note that the data d_4 produced by v_1 acts as input to both v_2 and v_3; in general, a data output by a module can act as input to multiple modules as well as (possibly) be a final output data. We call this phenomenon *data sharing*. Note that a data item cannot be output by more than one module. Also, observe that a workflow may be a multigraph: in this example, v_1 sends two data d_3 and d_4 to v_2.

Figure 1 also shows a sample execution of the workflow when $d_1 = 0$ and $d_2 = 1$ are the initial inputs; then $d_3 = 0, d_4 = 1, d_5 = 1$ and $d_6 = 0$. In this execution, the provenance of $d_6 = 0$ includes data values $d_1 = 0, d_2 = 1$ and $d_4 = 1$, and modules v_1 and v_3. On the other hand, the provenance of $d_5 = 1$ includes data values $d_1 = 0, d_2 = 1, d_3 = 0$ and $d_4 = 1$, and modules v_1 and v_2.

Given a workflow, we assume that there is a *workflow owner* and a *user*. The owner decides which components (data and modules) in the workflow are private for this user. The user can execute the workflow on different initial input data an *arbitrary* number of times. The user is always able to see the initial input data and the final output data. Further, the user wants to see some of the intermediate data as the response to a provenance query. However, the owner may or may not grant the user access to some of the intermediate data to ensure the privacy of the private components in the workflow. We say that a data is *hidden* if the user is *not* granted the access to see the corresponding data value for all executions of the workflow, otherwise the data is *visible*. Note that the user can always see the connections with the unique identifiers of the data, only the value of the data may be hidden. The private components of the workflow and their desired privacy level can be specific to a particular user; in our current work we have considered a single user and leave the problem of handling multiple users with different private components, i.e. handling more general *role based access control*, for future work.

[1] $a \oplus b$ denotes the XOR of a and b: If a, b are two bits, $a \oplus b = 1$ iff $a \neq b$.

3. MOTIVATING EXAMPLES AND PRIVACY ISSUES

Let us first consider some real life scenarios with potential privacy concerns.

Example 1. A module f in a medical diagnosis workflow detects the presence of a disease X by taking as input a set of patient attributes (eg. age, gender, smoking habit, blood pressure etc.) and produces as output the probability that the patient has a particular disease. The *functionality* of this module, represented by its input-output behavior $(I, f(I))$, may be highly sensitive. However, if a large enough set of such pairs are publicly available, then $f(I)$ could be exactly or approximately calculated for some patient input I, which may be socially, legally, or morally unacceptable. The module may also correspond to a proprietary or private algorithm, and the owner may be concerned that some approximation of the algorithm could be reverse-engineered, or that the relative importance of attributes in I could be revealed, by exposing its functionality.

Example 2. In a microarray experiments biologists must cope with variation in experimental conditions which is unrelated to the biological differences they seek to understand. *Normalization* is an attempt to compensate for experimental variation to enable the biological differences between samples to be more clearly seen. Microarray companies (and companies who provide the robots used to perform the experiments) therefore provide software to normalize results obtained using their chips/robots, which may then be used in scientific workflows. However, the software frequently uses *private data* obtained from experiments performed by other groups using the software, which should not be revealed.

Example 3. With the exponentially growing number of sequenced genomes that are becoming available, techniques for *functional annotation* – determining the function or role of each protein in a genome – have become important. To provide annotations of good quality, annotation workflows are being developed. One such annotation workflow may take as input a protein sequence and produce an annotation expressed using a set of Gene Ontology (GO) terms. In one such workflow, three modules are used: (i) Module M_1 compares the input protein sequence to the sequences of other genomes that have already been annotated, and provides the annotation of the closest protein found; (ii) Module M_2 searches for domains and outputs the set of sequences representing domains found in the protein (each domain may have its own function); (iii) Module M_3 locates the sequence in the genome as a whole to make sure that the sequence is complete, and completes it if not. These three tasks can be performed in different orders (either sequentially or in parallel), and some of the tasks may be skipped under certain conditions; in one workflow execution only M_1 may be executed, while in another execution of the workflow both M_1 and M_2 may be executed in parallel. For example, if M_2 provides only one sequence (the protein is composed of one single domain) then the annotation provided by M_1 is the final output. Otherwise the original result obtained by M_1 is flushed, and M_1 is executed for all the sequences provided by M_2; the final output is the union of the sets of GO terms of each domain. The creator of the workflow may be willing to reveal the modules used in the workflow, but not the order in which they are executed since this influences the efficiency of their annotation. In this case the algorithm used in the workflow, or the provenance information of any annotated genomes is private information.

Motivated by the above examples and the privacy model given in [10], we list three types of privacy issues in scientific workflows:

Module Privacy: The functionality of one or more modules in a workflow may be private information, as described in Example 1. In other words, the user should not be able to guess with a specified degree of certainty the output $f(x)$ of module f for any input x to that module. Workflow modules may also involve proprietary code or algorithms, in which case the algorithm to compute the function f may be private information.

Data Privacy: A module may consult a private database and output data containing sensitive personal information, e.g. a medical record or proprietary experimental results (see Example 2). While the module cannot be used in isolation, it could be used as an internal component of a workflow, or packaged within a *composite* module in which its output data is desensitized by a downstream module. Thus data flowing between modules in a workflow (*intermediate data*) may contain private data. For example, in Example 1 the value of $f(I)$ could be private.

Provenance Privacy: Details of how a particular data product has been generated in an execution of the workflow may also be private (see Example 3).

To further illustrate these three privacy issues, consider the example workflow in Figure 1. Suppose that module v_2 is proprietary, i.e. the function f_2 is not known and is private. Then *module privacy* requires that no adversarial user should be able to guess the value of $f_2(d_3, d_4)$ with high probability for any given assignment of the bits d_3 and d_4. For *data privacy*, if d_5 is a private data then the value of d_5 should not be revealed with high probability in any execution. Note the difference between module and data privacy: For module privacy, we may reveal the value of d_5 as long as we do not know the values of d_3, d_4 such that $f_2(d_3, d_4) = d_5$. However, for data privacy d_5 can never be revealed. Finally, *provenance privacy* for d_5 means that how the value $d_5 = 1$ has been generated in this execution (i.e. the graph itself in Figure 1) must be kept private.

In the next section we highlight our recent work on module privacy [11], in which the functionality of each module in the workflow is considered private.

4. PRESERVING MODULE PRIVACY

In [11] we model module privacy for workflows in which *all* modules are private and the user has no apriori knowledge of their functionality[2]. Although requiring all modules to be private is a restricted case, the privacy issues are already complex and yield significant insight into handling the more general case in which some modules may be public (i.e. not private).

It is easy to see that if information about all intermediate data is repeatedly given for multiple executions of a workflow on different initial inputs, then partial or complete functionality of modules may be revealed. As an example, suppose that the provenance data for module v_3 includes the following: For input $d_1 = 0, d_2 = 1, d_4 = 1$ and $d_6 = 0$, while for input $d_1 = 0, d_2 = 0, d_4 = 0$ and $d_6 = 1$. This reveals the function f_3 (i.e. $f_3(0) = 1$ and $f_3(1) = 0$).

Our mechanism to achieve privacy of modules in a workflow is to *hide a carefully chosen subset of intermediate data*, thereby limiting the amount of provenance data shown to the user. Users will

[2]The *functionality* of a module with an underlying function f refers to the input-output behavior of the module (i.e., the $(x, f(x))$ pairs for all possible inputs x to the module) and *not* the underlying algorithm to compute the function f.

always be able to see input and output data for the workflow as a whole, since they are able to run the workflow. An alternative approach commonly used in privacy preserving data mining techniques is to add noise to the answer of a query to meet the required privacy level. While adding noise is reasonable for scenarios like statistical databases, where the goal is to preserve the answer to the query within some precision, adding noise to output data for workflow modules destroys the integrity of the data and undermines repeatability of the experiment. We therefore choose to hide data rather than add noise.

We also assume that the user can see all connections or edges between modules in the workflow; only the *values* (data) associated with certain edges are hidden. Note that they will be hidden in *all* executions of the workflow.

We start by formalizing the notion of Γ-privacy of a module when it is not part of a workflow (*standalone module privacy*) as well as when it is a component of a workflow (*in-network module privacy*). Although it would seem that standalone module privacy should be enough, the interactions between modules in a network make the in-network setting more complex.

Informally, we say that a standalone module is Γ-private for some parameter $\Gamma > 0$ w.r.t the information visible to the user, if given any input x to the module, the actual value of $f(x)$ is indistinguishable from at least $\Gamma - 1$ other potential values of $f(x)$ (this also implies that the user can guess the correct output $f(x)$ with probability at most $\frac{1}{\Gamma}$). This definition is similar to the concept of ℓ-diversity [21], which resolves some problems of k-anonymity [27][3]. For example, consider module v_1 with $f_1(d_1,d_2) = (d_3,d_4)$, where $d_3 = d_1, d_4 = d_1 \vee d_2$; i.e. $f_1(0,0) = (0,0), f_1(0,1) = (0,1)$, $f_1(1,0) = (1,1)$ and $f_1(1,1) = (1,1)$. If we hide the output data d_3 or d_4, without any apriori knowledge of the function f_1, any input can be mapped to at least two different outputs. For example, when d_3 is hidden then $f_1(0,0)$ can be mapped to either $(0,0)$ or to $(1,0)$) and the user can guess the correct output only with probability $\frac{1}{2}$.

More precisely, a *consistent* function for f_1 with respect to a set of *hidden* input and output data preserves the input-output mapping with respect to the set of *visible data*. An example of a consistent function g for f_1 with respect to the hidden data $\{d_3\}$ is $g(d_1,d_2) = (d_3 = 1 - d_1, d_4 = d_1 \vee d_2)$. We say that hiding d_3 guarantees Γ-privacy for standalone module v_1 where $\Gamma = 2$, since for any assignments of input data d_1,d_2, the range $\bigcup_g g(d_1,d_2) \geq \Gamma = 2$, where the union is taken over all consistent functions g with respect to the hidden data d_3 for f_1. It can be shown that even if the user is allowed to see all executions of f_1 projected to the visible data, the actual value of $f(x)$ is indistinguishable from at least one more value (i.e. he cannot guess the correct output of any input with probability more than $\frac{1}{2}$).

The *standalone privacy requirement* of each module in a workflow can be specified in the most general form as a list of pairs of input-output data sets (called *set constraints*) that need to be hidden to ensure Γ-privacy. In our example, the sets $\{d_3\}$ or $\{d_4\}$ ensures 2-privacy for v_1. A more succinct, but less expressive, representation is to specify a list of pairs of numbers called *cardinality constraints* for each module. Hiding at least as many input and output data as specified in any one of the cardinality constraint pairs guarantees standalone privacy for that module . For example, hiding any 1 output data guarantees 2-privacy for v_1.

We assume that the standalone privacy requirement of a module is given by the module designer, who has complete knowledge of its functionality. However, note that in many real life scenarios the number of input and output data of a module is typically small. Therefore, given a table completely specifying the the inputs and outputs of the module, it may be possible to calculate the the standalone privacy requirement of a module by considering all possible combinations of input and output data.

The in-network Γ-privacy of a module is defined similarly: For *any* input to *any* module in the workflow, the user should not be able to guess the correct output of the module with probability $> \Gamma$. The notion of consistent function is extended to a sequence of consistent functions for the modules in the workflow. Note that we still consider privacy of individual modules in the workflow, but that interactions between modules in the workflow can potentially reveal more information than in the standalone case.

For example, suppose d_4 is hidden in the workflow depicted in Figure 1. Consider functions $g_1 = (d_1, \overline{d_1 \vee d_2})$, $g_2 = \overline{d_3 \oplus d_4}$ and $g_3 = d_4$. It can be verified that the visible bits produced by these functions are identical to those produced by f_1, f_2, f_3, in any execution of the workflow. Therefore, $\mathbf{g} = \langle g_1, g_2, g_3 \rangle$ is a consistent sequence of functions for the workflow, with respect to the hidden data d_4.

Although a sequence of consistent functions for the network is always consistent for the individual standalone modules with respect to the same set of hidden bits, the reverse is not true. In particular, a sequence of consistent functions for a set of standalone modules with respect to a set of hidden bits does not guarantee that the same sequence is consistent in a network setting, when the same set of bits is hidden.

Consider the following example – we define $\mathbf{g} = \langle g_1, g_2, g_3 \rangle$, where $g_1 = f_1, g_2 = \overline{d_3 \oplus d_4}$ and $g_3 = f_3$. Note that g_1, g_2, g_3 are consistent functions for f_1, f_2, f_3 respectively with respect to the hidden data d_4. However, g_1, g_2, g_3 is *not* a consistent sequence of functions for the workflow with respect to the hidden data d_4. In this case, if $d_1 = 0, d_2 = 0$, then $f_1(d_1,d_2) = (d_3 = 0, d_4 = 1)$, $f_2(0,1) = d_5 = 1$, whereas $g_1(d_1,d_2) = (d_3 = 0, d_4 = 1), g_2(0,1) = d_5 = 0$. But d_1, d_2 and d_5 are parts of initial input and final output data, and hence are always visible. This makes $\mathbf{g} = \langle g_1, g_2, g_3 \rangle$ inconsistent with the workflow with respect to the hidden data d_4.

Returning to the notion of in-network module privacy, let $v_1, \cdots, v_n$ be modules in a workflow. We say that a module v_i has in-network Γ-privacy with respect to a set of hidden bits S if $\bigcup_{\mathbf{g}=\langle g_1,\ldots,g_n \rangle} g_i(x) \geq \Gamma$, where the union is over all consistent sequence of functions for the workflow with respect to a hidden set of bits S. While standalone module privacy can be naturally captured in terms of hiding input and output bits at the module level, the notion of in-network privacy of a module is inherently linked to the topology of the network representing the workflow as well as the functionality of the modules.

In general, it appears that in-network privacy of modules requires us to consider the input-output behavior of all modules given all possible initial inputs. This may be computationally infeasible if the individual data items come from an unbounded or infinite domain, even if the number of data items in the initial input is small.

The first question we address in [11] is whether this global privacy requirement can be captured in terms of the local standalone privacy requirements of individual modules. Then we define a natural optimization problem, which we call *the secure-view* problem, as follows: *what is the minimum cost of intermediate data that can be hidden while guaranteeing all modules in the workflow have in-network Γ-privacy for some given parameter $\Gamma > 0$?* In this problem the *cost* of a data item indicates the utility lost to the user in terms of provenance information when the data item is hidden. The

[3]The notion of privacy according to k-anonymity is that, for each x, there are $\Gamma - 1$ other x' such that $(x, f(x))$ and $(x', f(x'))$ are indistinguishable w.r.t the visible information. However, it may not be sufficient to hide the value of $f(x)$ if there is not enough diversity in the $f(x')$ values, for example, if $f(x') = f(x)$ for all such x'

versions of secure-view problem, where the requirements for individual modules are either specified as explicit set constraints or as cardinality constraints, are shown to be NP-hard, even assuming unit cost and no data sharing. This motivates us to give poly-time *approximation algorithms*[4] for these problems.

5. FUTURE WORK

There are numerous promising directions for future exploration. The first question that arises is what happens when both public and private modules can constitute the workflow. If a module is *public* with known underlying function f, then given any input x to the module $f(x)$ can be computed. We can easily construct workflows where the presence of such modules provably compromises privacy of private modules.

For example, suppose in Figure 2 that modules v_1, v_3 are private with unknown functionality but that module v_2 is public, with underlying function f_2 which is a constant function (eg. $f_2 = (0,0)$ for all assignments of its inputs). Since the final output data bits d_3, d_4 are always visible, this causes a privacy breach for the private module v_3 on input $d_1 = 0$ and $d_2 = 0$.

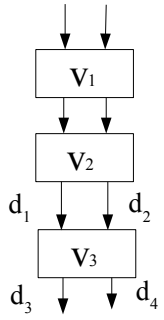

Figure 2: An example workflow where v_1, v_3 are private modules and v_2 is a public module

Thus handling privacy in this more general setting requires either identifying natural restrictions on the behavior of public modules (for example, one-one and/or onto functions), or introducing a parameter in the privacy requirement of the private modules which depends on some property of the public modules.

A more ambitious question will be to generalize the notion of private and public modules. For example, how we can guarantee that the prior knowledge of a user on the functionality of each module in the workflow does not change significantly given access to executions of the workflow on different inputs?

Another interesting open problem is to study whether a privacy definition similar to *differential privacy*[13, 14] is possible in the settings of workflows. A differentially private mechanism answering queries from a statistical database guarantees that the output distribution of the mechanism is *almost* the same on two *neighboring* databases that differ in at most one record, thereby ensuring that the privacy of any single record remains the same whether or not it participates in the database. However, in the settings of workflows and the privacy issues we are considering, there is no obvious notion of "neighborhood" since excluding any single module or data from a workflow can change the entire semantics of the workflow. A suitable query mechanism must also be explored since just hiding or revealing a single data value may not suffice.

So far we have only focused on preserving module privacy. However, exploring the mathematical models for data-privacy in workflow is another open problem. Note that existing techniques for p-reserving data privacy in statistical and relational databases do not directly extend to preserving data privacy in workflows, since standard provenance queries in workflows require revealing exact data values. As discussed earlier, adding noise reduces the repeatability of experiments involving scientific workflows. Also, just hiding a data value or denying a particular query may not be sufficient to ensure data privacy due to the dependence of data values on other modules and data in the workflow.

There may also be mechanisms other than hiding individual data items that can be used to guarantee privacy, for example, clustering portions of the workflow into *composite modules* [5]. It will be interesting to create a formal privacy model using clustering, and study issues such as if a privacy-preserving clustering preserves the mutual data dependency among the intermediate data not hidden inside clusters [5, 6, 26].

Other examples of future research directions include studying privacy in the presence of loops and forks (i.e. parallel module executions over elements of an input set), and handling multiple users with different privacy components corresponding to different permissions with possible collusions between users.

6. CONCLUSIONS

In this paper we discussed privacy issues in workflows, and briefly described a formalization of module privacy which gives high-quality solutions to the optimization problem that tries to maximize the utility of provenance data while preserving module privacy in the workflow. We also suggested several promising directions for future exploration in this very new area of workflow privacy.

Acknowledgments: We thank the anonymous reviewers for their detailed comments which helped to improve the presentation of this paper. This research was supported in part by NSF grant: IIS-0803524.

7. REFERENCES

[1] Privacy-preserving data mining: Models and algorithms. *www.charuaggarwal.net/toc.pdf*, 33(1):50–57, 2004.

[2] N. R. Adam and J. C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *PODS*, pages 153–162, New York, NY, USA, 2006. ACM.

[4] 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.

[5] O. Biton, S. Cohen-Boulakia, S. Davidson, and C. Hara. Querying and managing provenance through user views in scientific workflows. In *ICDE*, 2008.

[4] An algorithm is said to be a $\mu(n)$-*approximation algorithm* for some non-decreasing function $\mu(n) : \mathbb{N}^+ \to \mathbb{N}$ if on every input of size n it computes a solution where the value is within a factor of $\mu(n)$ of the optimal algorithm.

[6] O. Biton, S. B. Davidson, S. Khanna, and S. Roy. Optimizing user views for workflows. In *ICDT*, pages 310–323, 2009.

[7] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In *ER*, 2005.

[8] U. Braun, A. Shinnar, and M. Seltzer. Securing provenance. In *USENIX HotSec*.

[9] A. Campan and T. M. Truta. A clustering approach for data and structural anonymity in social networks. In *PinKDD*, 2008.

[10] A. Chebotko, S. Chang, S. Lu, F. Fotouhi, and P. Yang. Scientific workflow provenance querying with security views. *WAIM*, pages 349–356, July 2008.

[11] S. B. Davidson, S. Khanna, D. Panigrahi, and S. Roy. Preserving module privacy in workflow provenance. Technical Report MS-CIS-10-22, Dept. of Computer and Information Science, University of Pennsylvania, May 2010.

[12] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, New York, NY, USA, 2003. ACM.

[13] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

[14] C. Dwork. The differential privacy frontier (extended abstract). In *TCC*, pages 496–502, 2009.

[15] J. Freire, C. T. Silva, S. P. Callahan, E. Santos, C. E. Scheidegger, and H. T. Vo. Managing rapidly-evolving scientific workflows. In *IPAW*, 2006.

[16] A. Gil, W. K. Cheung, V. Ratnakar, and K. kin Chan. Privacy enforcement in data analysis workflows. In *PEAS*, 2007.

[17] Y. Gil and C. Fritz. Reasoning about the appropriate use of private data through computational workflows. In *Intelligent Information Privacy Management, Papers from the AAAI Spring Symposium*, pages 69–74, March 2010.

[18] R. Hasan, R. Sion, and M. Winslett. Introducing secure provenance: problems and challenges. In *StorageSS*, pages 13–18, New York, NY, USA, 2007. ACM.

[19] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu. Link privacy in social networks. In *CIKM*, pages 289–298, New York, NY, USA, 2008. ACM.

[20] J. Lyle and A. Martin. Trusted computing and provenance: Better together. In *TaPP '10: 2nd Workshop on the Theory and Practice of Provenance*, 2010.

[21] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.

[22] R. Motwani, S. U. Nabar, and D. Thomas. Auditing sql queries. In *ICDE*, pages 287–296, 2008.

[23] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards robustness in query auditing. In *VLDB*, pages 151–162. VLDB Endowment, 2006.

[24] T. Oinn *et al*. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(1), 2003.

[25] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. In *PODS*, pages 107–116, 2009.

[26] P. Sun, Z. Liu, S. B. Davidson, and Y. Chen. Detecting and resolving unsound workflow views for correct provenance analysis. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 549–562, New York, NY, USA, 2009. ACM.

[27] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

[28] 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.