# Enabling Privacy in Provenance-Aware Workflow Systems

Susan B. Davidson, Sanjeev Khanna, Sudeepa Roy, Julia Stoyanovich, Val Tannen
University of Pennsylvania, Philadelphia, USA
{susan, sanjeev, sudeepa, jstoy, val}@seas.upenn.edu

Yi Chen
Arizona State University, Tempe, USA
yi@asu.edu

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

## 1. INTRODUCTION

A new paradigm for creating and correcting scientific analyses is emerging, that of *provenance-aware* workflow systems. In such systems, repositories of workflow specifications and of provenance graphs that represent their executions will be made available as part of scientific information sharing. This will allow users to search and query both workflow specifications and their provenance graphs: Scientists who wish to perform new analyses may search workflow repositories to find specifications of interest to reuse or modify. They may also search provenance information to understand the meaning of a workflow, or to debug a specification. Finding erroneous or suspect data, a user may then ask provenance queries to determine what downstream data might have been affected, or to understand how the process failed that led to creating the data. With the increased amount of available provenance information, there is a need to efficiently *search* and *query* scientific workflows and their executions.

However, workflow authors or owners may wish to keep some information in the repository confidential. For example, intermediate *data* within an execution may contain sensitive information, such as a social security number, a medical record, or financial information about an individual. Although users with the appropriate access level may be allowed to see such confidential data, making it available to all users, even for scientific purposes, 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 access level should not be able to *infer* its behavior if they are allowed to see the inputs and outputs of the module. Finally, details of how certain modules in the workflow are connected may be proprietary, and so 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 the information provided in response to a search/query and the privacy guarantees that authors/owners desire.**

Scientific workflows are gaining wide-spread use in life sciences applications, a domain in which privacy concerns are particularly acute. We now illustrate three types of privacy using an example from this domain. Consider a personalized disease susceptibility workflow in Fig. 1. Information such as an individual's genetic make-up and family history of disorders, which this workflow takes as input, is highly sensitive and should not be revealed to an unauthorized user, placing stringent requirements on data privacy. Further, a workflow module may compare an individual's genetic make-up to profiles of other patients and controls. The manner in which such historical data is aggregated and the comparison is made, is highly sensitive, pointing to the need for module privacy. Finally, the fact that disease susceptibility predictions are generated by "calibrating" an individual's profile against profiles of others may need to be hidden, requiring that workflow structure be kept private.

As recently noted in [8], "You are better off designing in security and privacy ... from the start, rather than trying to add them later."[1] We apply this principle by proposing that privacy guarantees should be *integrated* in the design of the search and query engines that access provenance-aware workflow repositories. Indeed, the alternative would be to create multiple repositories corresponding to different levels of access, which would lead to inconsistencies, inefficiency, and a lack of flexibility, affecting the desired techniques.

This paper focuses on *privacy-preserving* management of *provenance-aware* workflow systems. We consider the formalization of privacy concerns, as well as query processing in this context. Specifically, we address issues associated with *keyword-based search* as well as with *querying* such repositories for *structural patterns*.

To give some background on provenance-aware workflow systems, we first describe the common model for workflow specifications and their executions (Sec. 2). We then enumerate privacy concerns (Sec. 3), consider their effect on query processing, and discuss the challenges (Sec. 4).

## 2. MODEL

Workflow *specifications* are typically represented by graphs, with nodes denoting modules and edges indicating dataflow between modules. Workflow specifications may be *hierarchical*, in the sense that a module may be *composite* and itself

[1]While the context for this statement was the use of full body scanning in airports (where the privacy issues are obvious), it is equally valid in provenance systems!

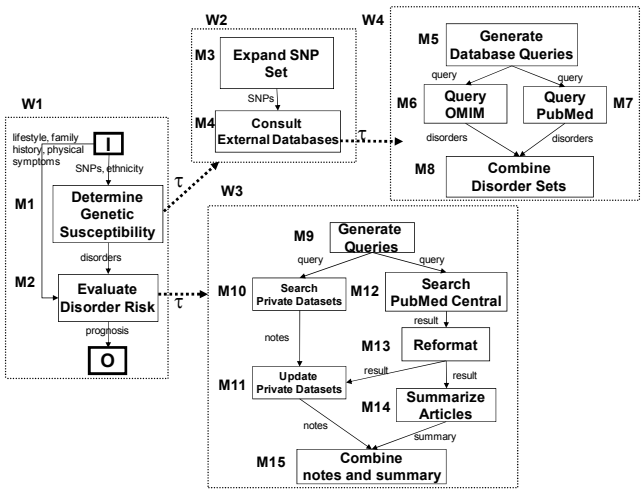


**Figure 1: Disease Susceptibility Workflow Specification**

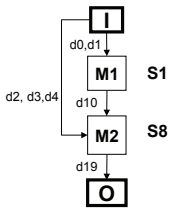


**Figure 2: View of Provenance Graph**

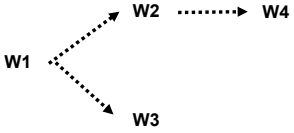


**Figure 3: Expansion Hierarchy**

contain a workflow. Composite modules are frequently used to simplify workflow design and allow component reuse.

For example, the workflow in Fig. 1 estimates disease susceptibility based on genome-wide SNP array data. The input to the workflow, whose top-most level is given by the dotted box labeled $W1$, is a set of SNPs, ethnicity information, lifestyle, family history, and physical symptoms. The first module in $W1$, $M_1$, determines a set of disorders the patient is genetically susceptible to based on the input SNPs and ethnicity information. The second module, $M_2$, refines the set of disorders for which the patient is at risk, based on lifestyle, family history, and physical symptoms.

Fig. 1 also contains $\tau$-labeled edges that give the definitions of composite modules, which we call *expansions*. For example, $M_1$ is defined by the workflow $W2$, $M_2$ by the workflow $W3$, and $M4$ by the workflow $W4$. Hence $W2$ and $W4$ are *subworkflows* of $W1$, and $W3$ is a subworkflow of $W2$. The $\tau$ expansions (subworkflow relationships) naturally yield an *expansion hierarchy* as shown in Fig. 3.

*Prefixes* of the expansion hierarchy can be used to define views of a workflow specification.[2] Given a prefix, the *view* that it defines is given by expanding the root workflow so that composite modules in the prefix are replaced by their expansions. For example, consider the expansion hierarchy in Fig. 3 and its prefix consisting of $\{W1, W2\}$. This prefix determines a view of the specification in Fig. 1, which is the

[2]Recall that a *prefix* of a rooted tree $T$ is a tree obtained from $T$ by deleting some of its subtrees (i.e., some nodes and all their descendants).

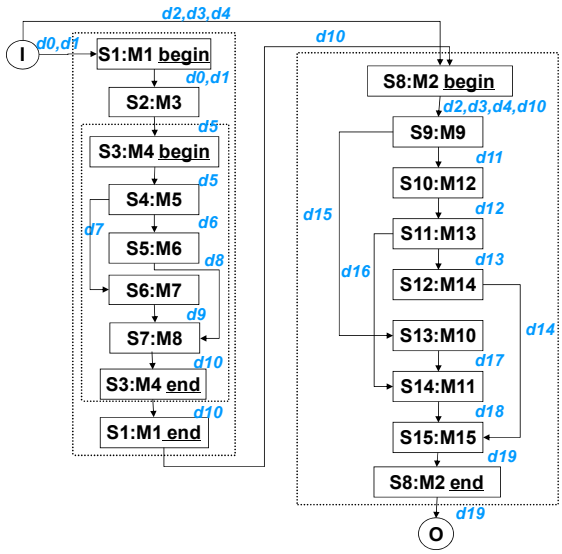


**Figure 4: Disease Susceptibility Workflow Execution**

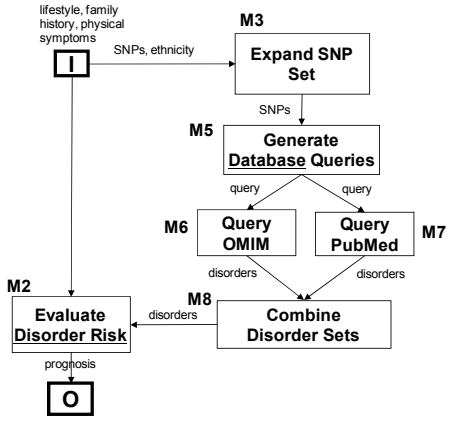


**Figure 5: Result of Query "Database, Disorder Risks"**

simple workflow obtained from $W1$ by replacing $M1$ with $W2$. Another view is the full expansion, which yields a workflow with module names $I, O, M3$,and $M5 - M15$ and whose edges include one from $M3$ to $M5$ and another from $M8$ to $M9$. We will shortly discuss the benefit of views.

A workflow specification describes the possible run-time *executions*. Executions are modeled similarly to simple workflow graphs, but additionally associate a unique process id with a module execution, and data with edges. When execution reaches a composite module, it continues in the corresponding subworkflow and eventually returns (like a procedure call). For example, an execution of the workflow specification in Fig. 1 is shown in Fig. 4. In this example, for clarity we show the process id appended to the name of the module being executed, e.g. S1:M1. Following common practice [1], each composite module execution is represented by two nodes, the first standing for its activation and the second for its completion, e.g. S1:M1-begin and S1:M1-end.

In an execution, data flows over the edges. We assume that each data item is the output of exactly one module execution and has a unique id. We can therefore annotate each edge $M \rightarrow N$ in the execution with the set of data

items that flow as the output of $M$ to the input of $N$. For example, in Fig. 4 the set $\{d0, d1\}$ flows from I to S1:M1.

The *provenance* of a data item $d$ in an execution $E$ is therefore the subgraph induced by the set of paths from the start node to the end node of $E$ that produced $d$ as output. In the sequel, we blur the distinction between the provenance of data items and the executions that produce them.

As introduced in [2], we can use views to simplify what is seen of an execution. Using the view defined by prefix $\{W1\}$, the execution of Fig. 4 would be simplified to that in Fig. 2. Thus views can be used to define access control to address privacy concerns. Specifically, we can define a user's access privilege as the finest grained view that s/he can access, called an *access view*.

# 3. PRIVACY

Privacy concerns are tied to the workflow components: data, modules, and the structure of a workflow. To illustrate them, consider again the sample workflow in Fig. 1. *Data privacy* requires that the output of M1, i.e., the genetic disorders the patient is susceptible to, should not be revealed with high probability, in any execution, to users without the required access privilege. Such data masking is a fairly standard requirement in privacy-aware database systems. *Module privacy* is more particular: It requires that the functionality of a private module – that is, the mapping it defines between inputs and outputs – is not revealed to users without the required access privilege. Returning to our example, assuming that M1 implements a function $f_1$, module privacy with respect to M1 requires that no adversarial user should be able to guess the output $f_1$(`SNP`, `ethnicity`) with high probability for any `SNP` and `ethnicity` input. From a patient's perspective, this is important because they do not want someone who may happen to have access to their `SNP` and `ethnicity` information to be able to determine what `disorders` they are susceptible to. From the module owner's perspective, they do not want the module to be simulated by competitors who capture all input-output relationships. Finally, *structural privacy* refers to hiding structure of the information flow in the given execution. In this example it might mean that users without the required access privilege should not know whether or not `lifestyle` was used to calculate the `disorders` output by M1.

Broadly speaking, the fundamental privacy question to be addressed is: **How do we provide provable guarantees on privacy of components in a workflow while maximizing utility with respect to provenance queries?**

In doing so, we must understand 1) how to measure privacy; 2) what information can be hidden; 3) how to measure utility; and 4) how to efficiently find solutions that *simultaneously* provide provably good guarantees on privacy and utility. It is worth highlighting an important characteristic, namely, that all privacy guarantees are required to hold over repeated executions of a workflow with varied inputs.

We discuss module and structural privacy in more depth before turning to the impact of privacy on search and query mechanisms.

**Module Privacy.** 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. The approach that we take in [4] is to *hide a carefully chosen subset of intermediate data*, thereby limiting the amount of provenance data shown to the user and guaranteeing some desired level of privacy. Ignoring for now structural privacy, one may assume that users can see the connections (edges) between modules in the workflow; only the *values* of selected intermediate data are hidden, in *all* executions of the workflow. Since there may be several different subsets of intermediate data whose hiding yields the desired level of privacy, and certain data may have higher utility to users than other data (i.e., data may be weighted), this becomes an interesting optimization problem.

**Structural Privacy.** The goal of structural privacy is to keep private the information that some module $M$ contributes to the generation of a data item $d$, output by another module $M'$. For instance, in the execution of the workflow $W3$, we may wish to hide the fact that the reformatted data from PubMed Central (module $M13$) contributes to updating the private DB, and hence to the output of module $M11$. One possible approach is to delete edges and vertices so as to eliminate all paths from $M$ to $M'$, e.g., in this example to delete the edge $M13 \rightarrow M11$. However, by doing so, we may hide additional provenance information that does not need be hidden (e.g., the existence of a path from $M12$ to $M11$). Another approach is to use *clustering*, where certain modules are hidden in a composite module $P$ so that the reachability of any pair $(u, v)$ in $P$ is no longer externally visible. For example, we could cluster $M11$ and $M13$ into a single composite module. However, we may now infer incorrect provenance information, e.g., that there is a path from $M10$ to $M14$. This is called an *unsound view* in [3, 9]. Once again, one faces a challenging optimization problem: guaranteeing an adequate level of privacy while preserving soundness and minimizing unnecessary loss of information.

# 4. PRIVACY-PRESERVING QUERY EVALUATION

Query languages for workflow specifications/executions support two main types of queries: *structural queries* that allow users to select sub-workflows based on structural properties (e.g., "find executions where `Expand SNP Set` was executed before `Query OMIM` and return the provenance information for the latter") and *keyword queries* that retrieve sub-workflows that match the input keywords (e.g., "find workflows that include 'disorder risk' and 'database'", result shown in Fig. 5). In both cases the query answer is given as a minimal view of the flow that satisfies the query criteria and includes the keywords (see [1, 7] for formal definitions). Much research was recently devoted to developing efficient query evaluation techniques in this context. Unfortunately, none of this works addresses the privacy issues mentioned in the previous section. We consider below the main challenges in enabling such privacy-preserving query evaluation.

**Privacy-controlled Semantics for Queries.** Before one can consider efficient query evaluation, there is a need to formally define the semantics of queries in this context. What is the correct answer to a given query, assuming privacy and access control settings that are guided by the hierarchical structure of workflow specifications? Notably, the three different kinds of privacy we consider may have different impacts on what information should be available to a given user, and therefore on the semantics of queries and on the definition of search results. For example, consider struc-

tural privacy, which may be achieved by clustering nodes to form a composite module. Such an approach may introduce extraneous paths, causing misleading provenance information [9]. A challenging question to investigate is then the following: In the presence of unsound views, how can we define search results that maximize utility (defined to be some function of both the number of correct node connectivity relationships captured and the number of modules disclosed in a result), while guaranteeing privacy?

**Efficient Search with Privacy Guarantees.** There is clearly a distinction between defining the formal semantics of relevance and computing the answers efficiently. It may be infeasible to create variants of the workflow repository, one for each privilege/privacy setting, due to high space overhead. Instead, the information must be hidden on-the-fly, which usually leads to processing overhead. A challenge is then to develop algorithms for addressing these computational problems.

First, standard, non-privacy preserving workflow management systems use various indexing structures or materialized views to speed up query processing. With data privacy, we must manage an index with "different user views", as users often have different privileges on data accesses. A promising direction is to consider representing the specification and execution graphs using advanced data structures that classify and group their elements based on privacy settings. Another promising direction is to consider user groups when utilizing cached information during query processing.

Second, to achieve privacy, one needs to generate query results with respect to user access privileges (view). One approach would be to first construct a full answer, oblivious to the privacy requirement. If the result reveals sensitive information, we may gradually "zoom-out" the view by hiding details of composite modules and sensitive data, until privacy is achieved. However, this can be expensive as each zoom-out may involve a disk access. Techniques must be developed to efficiently construct user-specific answers.

**Impact of Ranking on Privacy Preservation.** Sometimes a user query is ambiguous and the results can have varying degrees of relevance. Ranking is therefore an important function, especially for a keyword-based search engine. One typical metric in ranking is to consider term frequency (TF) and inverse document frequency (IDF). A highly ranked result is likely to have more occurrences of an input keyword than a lowly ranked result. Thus, a user might be able to infer the range of value occurrences in a result even though s/he is unable to see the values due to privacy preservation. Such inference may cause information leakage, and affect module and value privacy. A challenge is to design sophisticated ranking schemes that not only rank results in the order of relevance but are also privacy-aware.

# 5. RELATED WORK AND CONCLUSIONS

Extensive work has been done on privacy in various settings, e.g., *data mining*, *social networks*, *auditing queries*, and *statistical databases*. The problem of defining a consistent set of *access controls* to preserve privacy in a workflow has also been considered, as well as the problem of ensuring the *lawful use* of data according to some specified privacy policies. However, a formal study of privacy issues specific to workflows, with provable privacy guarantees, has not yet been done. It will be particularly interesting to see if ideas from *differential privacy* [5, 6] can be used in this setting. Although it is the strongest notion of privacy known to date, it is also known that *no* deterministic algorithm can guarantee differential privacy. This may limit the applicability of differential privacy in our setting — provenance in scientific workflows is used to ensure reproducibility of experiments, and adding random noise to provenance information may render it useless.

Keyword search has been extensively studied, e.g., for *graph-structured* and *tree-structured* (e.g., XML) data. There has also been work on *graph query languages*. Nonetheless, prior work does not adequately address the requirements of privacy-aware workflow management systems, for two reasons. First, workflow specifications involve both dataflow and expansion ($\tau$) edges, and the difference between them cannot be ignored [1, 7]. Second, prior work does not consider search and query processing in the face of privacy requirements.

In summary, there are significant novel research challenges that must be addressed in developing the next generation of privacy-enabled provenance-aware workflow systems. These range from formalizing privacy requirements and notions of utility, and developing algorithms for associated optimization problems, to designing efficient systems that integrate privacy with query processing mechanisms.

# 6. ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation grants IIS-0803524, IIS-0629846, and IIS-0915438, by NSF CAREER award IIS-0845647, and by grant 0937060 to the Computing Research Association for the CIFellows Project. This work was also supported in part by an IBM faculty award. This work was also supported in part by the Israel Science Foundation, and by the US-Israel Binational Science Foundation.

# 7. REFERENCES

[1] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes with BP-QL. *Information Systems*, 33(6):477–507, 2008.

[2] O. Biton, S. C. Boulakia, S. B. Davidson, and C. S. Hara. Querying and managing provenance through user views in scientific workflows. In *ICDE*, pages 1072–1081, 2008.

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

[4] S. B. Davidson, S. Khanna, D. Panigrahi, and S. Roy. Preserving module privacy in workflow provenance. *Manuscript available at http://arxiv.org/abs/1005.5543.*

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

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

[7] Z. Liu, Q. Shao, and Y. Chen. Searching workflows with hierarchical views. *PVLDB*, 3(1), 2010.

[8] S. S. Shapiro. Privacy by design: moving from art to practice. *Commun. ACM*, 53(6):27–29, 2010.

[9] P. Sun, Z. Liu, S. B. Davidson, and Y. Chen. Detecting and resolving unsound workflow views for correct provenance analysis. In *SIGMOD*, pages 549–562. ACM, 2009.