

Scientific Workflow Provenance Querying with Security Views

Artem Chebotko[#], Seunghan Chang[#], Shiyong Lu, Farshad Fotouhi
Department of Computer Science, Wayne State University, Detroit, MI, 48202, USA
{artem,chang,shiyong,fotouhi}@wayne.edu

[#] Authors contributed equally.

Ping Yang
Department of Computer Science, Binghamton University, Binghamton, NY, 13902, USA
pyang@cs.binghamton.edu

Abstract

Provenance, the metadata that pertains to the derivation history of a data product, has become increasingly important in scientific workflow environments. In many cases, both data products and their provenance can be sensitive and effective access control mechanisms are essential to protect their confidentiality. In this paper, we propose i) a formalization of scientific workflow provenance as the basis for querying and access control; ii) a security specification mechanism for provenance at various granularity levels and the derivation of a full security specification based on inheritance, overriding, and conflict resolution rules; iii) a formalization of security views that are derived from a scientific workflow run provenance for different roles of users; and iv) a framework that integrates abstraction views and security views such that a user can examine provenance at different abstraction levels while respecting the security policy prescribed for her. We have developed the SECPROV prototype to validate the effectiveness of our approach.

1 Introduction

In recent years, more and more scientists start to use workflow technologies to automate the steps they need to go through from raw datasets to potential scientific discovery. As a result, scientific workflows have emerged as a new field to address the new requirements from scientists [30, 32]. A scientific workflow is a formal specification of a scientific process, which represents, streamlines, and automates the steps from dataset selection and integration, computation and analysis, to final data product presentation and visualization. A scientific workflow management system supports the specification, execution, re-run, and monitoring of scientific processes [30, 33, 15, 21, 18, 41, 20].

Provenance management is essential for scientific workflows to support scientific discovery reproducibility, result interpretation, and problem diagnosis [36, 9]; such a facility is usually not necessary for business workflows. Provenance metadata captures the derivation history of a data product, including the original data sources, intermediate data products, and the steps that were applied to produce the data product. In many cases, both data products and their provenance can be sensitive and effective access control mechanisms are essential to protect their confidentiality.

As an example, consider an intragenomic recombination analysis scientific workflow shown in Fig. 1, which is simplified from our original workflow that consists of over 50 workflow tasks [3]. For a given genome, this workflow takes its protein sequences and identifies all its multi-gene families (T_1). A particular multi-gene family is then selected by the user and its associated DNA sequences are retrieved (T_2). Then a recombination analysis is performed on the retrieved sequences (T_3), which consists of two steps: a multiple DNA sequence alignment step (T_4) and a gene conversion detection step (T_5); the latter is implemented by an off-the-shelf program GENECONV with an input data file preparation step (T_6). As shown in the figure, a scientific workflow consists of a set of workflow tasks, workflow inputs, workflow outputs, and data channels that connect them. Each task represents a computational or analytical step of a scientific process. A task has input ports and output ports that provide the communication interface to other tasks. Tasks are linked together into a workflow as an acyclic graph via data channels. During workflow execution, tasks communicate with each other by passing data via their ports through data channels. Finally, a task can have an arbitrary number of input parameters (special kind of input ports), which are used by a scientist to configure its dynamic execution behavior. In the workflow, p_1, \dots, p_8 are input parameters whose meanings are described in the

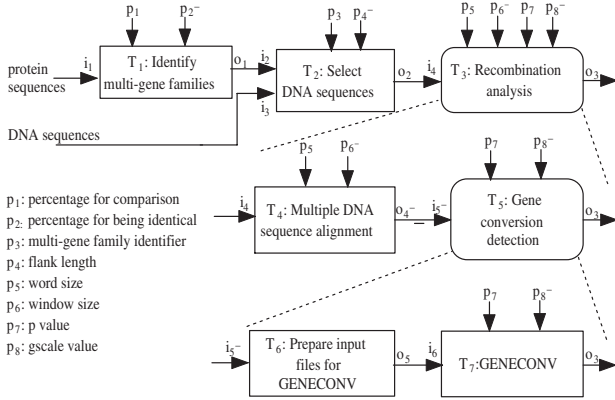


Figure 1. *W*: an intragenomic recombination analysis scientific workflow.

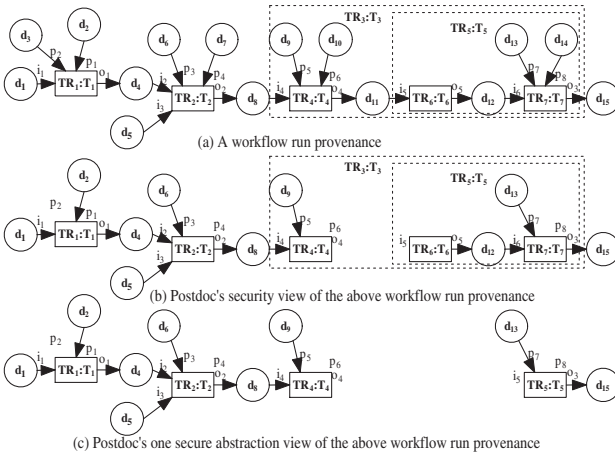


Figure 2. Security view and secure abstraction view of a scientific workflow run.

figure. The workflow is hierarchical: composite task T_3 consists of atomic task T_4 and composite task T_5 , which in turn consists of atomic tasks T_6 and T_7 . Suppose both data products and their provenance are sensitive, and we define a role *Postdoc*. *Postdoc* can access everything except $p_2, p_4, p_6, p_8, o_4, i_5$, and the dependency induced by data channel from o_4 to i_5 ; this is specified by a “-” annotation on them.

This workflow can be executed many times for different genomes or for the same genome but with different parameter settings, resulting in vast amounts of data products and provenance information. Fig. 2.(a) shows a sample scientific workflow run of the workflow in Fig. 1. There are two kinds of nodes: circles represent data products and rectangles represent workflow task runs, which are labeled their workflow task run identifier in the form of $TR_i : T$ where T is the identifier of the task.

An edge from a data product to a task run represents a *consume* relationship, while an edge from a workflow task to a data product represents a *produce* relationship. Fig. 2.(a) shows the most detailed workflow run provenance information that will be recorded by a scientific workflow provenance system. However, following our example, since *Postdoc* cannot access $p_2, p_4, p_6, p_8, o_4, i_5$, and the data channel from o_4 to i_5 , the most detailed provenance information *Postdoc* can access is shown in Fig. 2.(b), in which data products $d_3, d_7, d_{10}, d_{14}, d_{11}$ and d_{11} 's incoming and outgoing edges are eliminated. Finally, since a user typically browses a workflow run provenance at a particular abstraction level at a time, a secure abstraction view of the above provenance is shown in Fig. 2.(c) for *Postdoc*, in which $TR_5 : T_5$ is viewed as a blackbox but $TR_3 : T_3$ is viewed as a composition of $TR_4 : T_4$ and $TR_5 : T_5$.

Although many access control mechanisms have been proposed for business workflows [6, 19, 11, 2, 23, 1, 10, 4], they are not suitable for scientific workflows since business workflows are control-flow oriented while scientific workflows tend to be dataflow oriented and provenance enabled [30]. In particular, existing access control mechanisms for business workflows cannot be used to restrict the access to the dependency relationships between data products in scientific workflow provenance; see more discussion in the related work section.

The main contributions of this paper are: i) a formalization of scientific workflow provenance as the basis for querying and access control; ii) a security specification mechanism for provenance at various granularity levels and the derivation of a full security specification based on inheritance, overriding, and conflict resolution rules; iii) a formalization of security views that are derived from a scientific workflow run provenance for different roles of users; and iv) a framework that integrates abstraction views and security views such that a user can examine provenance at different abstraction levels while respecting the security policy prescribed for her. We have developed the SEC PROV prototype to validate the effectiveness of our approach.

2 Related work

The importance and requirements of security have been well understood in business workflows [6, 19, 11, 2, 23, 1, 10, 4]. Much work has been done in authentication [31], authorization [39, 7, 24, 34, 40, 28], data privacy, and secure workflow models [26, 27, 5]. While process integrity is ensured by constrained planning [7, 38, 17], data confidentiality is often supported by integrating Role-Based Access Control [35] in the enactment system [28, 22, 26]. Security requirements can be either managed by the workflow system itself [25, 29], or enforced outside of the workflow engine [14].

While execution logs are maintained in business workflows, a richer set of provenance information is collected and maintained in a scientific workflow management system for the purpose of supporting scientific discovery reproducibility, result interpretation, and problem diagnosis [36, 9]. Provenance metadata captures the derivation history of a data product, including the original data sources, intermediate data products, and the steps that were applied to produce the data product. The provenance management problem concerns about the efficiency and effectiveness of collecting, storing, browsing, querying, and visualization of scientific workflow provenance metadata [13].

Although security issues for provenance have been identified by a couple of researchers [37, 12], these issues are still open problems. While several access control mechanisms have been proposed for business workflows [6, 19, 11, 2, 23, 1, 10, 4], they are insufficient for scientific workflows provenance for the following reasons: 1) they do not support the restriction of access to the dependency relationships between data products in workflow provenance; 2) they have not considered different levels of workflow provenance, including workflows, tasks, ports, data channels, and their containment and inheritance relationships; 3) they have not considered the interaction of access control and abstraction. The latter is used for viewing provenance at different abstraction level as scientific workflows can be hierarchical; and 4) they have not considered the data channel constraint introduced by a scientific workflow specification.

Our notion of abstraction views is similar to the notion of user views introduced in [16, 8]. However, we developed the notion of security views, and based on abstraction views and security views, we developed the notion of secure abstraction views. Secure abstraction views enable a user to access and query provenance information at different abstraction levels, returning the intersection of the abstraction view requested by the user and the security view corresponding to the access control policy prescribed for the user.

Finally, access controls for XML data cannot be applied to provenance due to the differences in data models (XML tree vs. provenance acyclic graph), abstraction mechanisms (XML tree structure vs. workflow structure, rather than provenance structure), and data relationships (XML parent-child relationships vs. provenance data dependencies).

3 Scientific Workflow Provenance Model

In this section, we formalize a model for scientific workflow provenance by defining the notions of atomic task, composite task, task run, and workflow run provenance.

Definition 3.1 (Atomic task) An *atomic task* is a tuple $(tid, \mathcal{IP}, \mathcal{OP})$, where tid is the unique identifier of the task,

$\mathcal{IP} = \{i_1, i_2, \dots, i_m\}$ is the set of input ports of the task, and $\mathcal{OP} = \{o_1, o_2, \dots, o_n\}$ is the set of output ports of the task. We use $tid.i_j$ and $tid.o_k$ to denote the input port i_j and the output port o_k of the task tid , respectively. \diamond

Definition 3.2 (Composite task) A *composite task* (or *sub-workflow*) is a tuple $(wid, \mathcal{IP}, \mathcal{OP}, \mathcal{T}, \mathcal{F})$, where wid is the unique identifier of the composite task, $\mathcal{IP} = \{i_1, i_2, \dots, i_m\}$ is the set of input ports of the composite task, $\mathcal{OP} = \{o_1, o_2, \dots, o_n\}$ is the set of output ports of the composite task, \mathcal{T} is the set of constituent tasks of the composite task, each of which is either atomic or composite, and \mathcal{F} is the set of data channel of the composite task with each $(t_1.o_j, t_2.i_k) \in \mathcal{F}$ representing the data channel from output port $o_j \in t_1.\mathcal{OP}$ of some task $t_1 \in \mathcal{T}$ to input port $i_k \in t_2.\mathcal{IP}$ of some other task $t_2 \in \mathcal{T}$. \diamond

At the top level, a scientific workflow is also considered as a composite task. A task might be used in several parts of a scientific workflow or in an iteration construct. Such a task might get executed multiple times in a particular workflow execution. Each execution of a task T is called a *task run* and is assigned with a unique task run identifier in the form of $TR_i : T$; see Fig. 2 for examples. At the top level, a scientific workflow is considered as a composite task. Therefore, a workflow run is a special case of a task run. Each execution of a scientific workflow produces a *workflow run provenance*, which archives the derivation history of data products, including the task runs that have contributed to the data products. We formalize the notion of workflow run provenance as follows.

Definition 3.3 (Workflow run provenance) A *workflow run provenance* is a tuple $(wrid, wid, \mathcal{D}, \mathcal{TR}, Consume, Produce)$, where $wrid$ is the unique identifier of the workflow run, wid is the identifier of the workflow that $wrid$ executes, \mathcal{D} is the set of all the data products consumed or produced by the workflow run, \mathcal{TR} is the set of all the task runs executed in the workflow run with $wrid \subseteq \mathcal{TR}$, $Consume$ is the relationship set with each $(d, tr.i_j) \in Consume$ representing that input port i_j of task run $tr \in \mathcal{TR}$ consumed data product $d \in \mathcal{D}$ during the workflow run, $Produce$ is the relationship set with each $(tr.o_k, d) \in Produce$ representing that output port o_k of task run $tr \in \mathcal{TR}$ produced data product $d \in \mathcal{D}$ during the workflow run. \diamond

Our workflow run provenance model captures provenance at various levels of abstraction and granularity: *Consume* and *Produce* dependency information are collected for all levels of a composite task or workflow, and provenance is collected for task runs, ports, and data channels (by *Consume* and *Produce*). Such a scientific workflow provenance model provides the basis for querying and access control of provenance at different levels of abstraction and granularity.

4 Security Model

In this section, we propose a Role-Based Access Control for scientific workflow run provenance. Using our access control, one can not only impose restriction on the access to data products consumed and produced during a workflow execution, but also impose restriction on the access to the dependency relationships among the data products. When a workflow is designed, a system security administrator provides a security specification for each semantic role of users in the system. We propose three levels of security specification, namely, *task level*, *port level*, and *data channel level*.

Task level security specification. At this level, an atomic task, composite task, or a whole workflow, can be annotated to be accessible (+) or inaccessible (−), meaning that all the data products consumed and produced by an execution of the task are accessible or inaccessible, respectively. For example, the administrator may specify security annotations for all or some of the tasks in the following way: $\langle \text{Task } t, \text{Role } r, \text{security annotation } a \rangle$, where a can be either + or −. We call a set of such security annotations as a *task level security specification* and denote it as TL . Any task that has no security annotation in TL inherits the annotation of its nearest ancestor. At the top level, the annotation of a whole workflow can be set to a default annotation, either + or −; we use + as the default annotation for a workflow in this paper. The annotation of a task can be calculated by function $getTaskSecAnnot$ defined as follows:

```

01 Function getTaskSecAnnot
02 Input: Task  $t$ , Role  $r$ , workflow specification  $W$ , security specification  $TL$ 
03 Output: Task security annotation  $\langle t, r, a \rangle$ 
04 Begin
05 If there exists  $\langle t, r, a \rangle \in TL$ , then Return  $\langle t, r, a \rangle$ ;
06 If  $t$  is a workflow  $W$ , then Return  $\langle t, r, + \rangle$ ; /*accessible by default*/
07 Let  $t_p$  be a composite task that directly contains  $t$ ;
08  $\langle t_p, r, a_p \rangle = getTaskSecAnnot(t_p, r, W, TL)$ ;
09 Return  $\langle t, r, a_p \rangle$ ; /*inheritance from a parent in a task hierarchy*/
10 End Function

```

Intuitively, input and output ports inherit security annotations of tasks that they belong to. We provide more details on port security annotations in the following.

Port level security specification. At this level, an individual port can be assigned to be accessible (+) or inaccessible (−), meaning that all the data products consumed or produced by this port from all workflow runs of the workflow are accessible or inaccessible, respectively. For example, the administrator may specify security annotations for all or some of the ports in the following way: $\langle \text{Port } p, \text{Role } r, \text{security annotation } a \rangle$, where a can be either + or −. We call a set of such security annotations as a *port level security specification* and denote it as PL . Any port that has no security annotation in PL inherits the annotation of its owner task. For an unannotated port that simultaneously belongs to a hierarchy of tasks, its annotation is set to − if one of its owner task has an annotation of − and set to +

otherwise; other derivation rules can be used or user intervention can be incorporated from an interface. The explicit annotation of a port in PL always overrides the implicit security annotation inherited from a task that the port belongs to. In summary, the annotation of a port can be calculated by function $getPortSecAnnot$ defined as follows:

```

01 Function getPortSecAnnot
02 Input: Port  $p$ , Role  $r$ , workflow spec.  $W$ , sec. specifications  $TL$  and  $PL$ 
03 Output: Port security annotation  $\langle p, r, a \rangle$ 
04 Begin
05 If there exists  $\langle p, r, a \rangle \in PL$ , then Return  $\langle p, r, a \rangle$ ;
06 Let  $t_1, t_2, \dots, t_n$  be tasks that have port  $p$ ;
07  $\langle t_1, r, a_1 \rangle = getTaskSecAnnot(t_1, r, W, TL)$ ,
08  $\langle t_2, r, a_2 \rangle = getTaskSecAnnot(t_2, r, W, TL), \dots$ ,
09  $\langle t_n, r, a_n \rangle = getTaskSecAnnot(t_n, r, W, TL)$ ;
10 If  $a_1 = -$  or  $a_2 = -$  or ... or  $a_n = -$ , then Return  $\langle p, r, - \rangle$ ;
11 Else Return  $\langle p, r, + \rangle$ ;
12 End Function

```

Data channel level security specification. At this level, a data channel between two ports can be assigned to be accessible (+) or inaccessible (−), meaning that a user is revealed or unrevealed, respectively, that there exists a dependency between a data product produced by one port and a data product consumed by another port. For example, the administrator may specify security annotations for all or some of the data channels in the following way: $\langle \text{Data channel } (p_1, p_2), \text{Role } r, \text{security annotation } a \rangle$, where a can be either + or −. We call a set of such security annotations as a *data channel level security specification* and denote it as DL . In addition, we propose to use rules to derive security annotations for data channels which have no assignments in DL . Such rules are specified in a table that we denote DL^T . DL^T is searched for the first applicable rule that is used to derive a security annotation for a data channel; if none of the rules is applicable, either a default annotation, + or −, or the user can be prompted for choosing an annotation for the unannotated data channel.

A sample table DL^T with two data channel security annotation derivation rules is shown in Table 1. The first rule derives the + annotation for any data channel (p_1, p_2) with both p_1 and p_2 having the annotation of +. The second rule derives the − annotation for any data channel (p_1, p_2) with both p_1 and p_2 having the annotation of −. One can also change the second rule to a derivation of a + to allow the access of the dependency induced by the data channel even though the ports are not accessible. Note that DL^T is optional (can be empty) for our security specification model and the presented rules are only examples. In summary, the annotation of a data channel can be calculated by function $getDataChannelAnnot$ defined as follows:

```

01 Function getDataChannelSecAnnot
02 Input: Data channel  $(p_1, p_2)$ , Role  $r$ , workflow spec.  $W$ ,
03     sec. specifications  $TL, PL, DL$ , and table  $DL^T$ 
04 Output: Data channel security annotation  $\langle (p_1, p_2), r, a \rangle$ 
05 Begin
06 If there exists  $\langle (p_1, p_2), r, a \rangle \in DL$ , then Return  $\langle (p_1, p_2), r, a \rangle$ ;
07 Find the first (top-down search) rule  $R$  in  $DL^T$  that is applicable to  $(p_1, p_2)$ 
08 If  $R$  is found, then apply it and Return the result of  $R$  application;
09 Else Return  $\langle (p_1, p_2), r, defaultvalue \rangle$ ; /*customizable default val.*/
10 End Function

```

Table 1. Sample table DL^T

#	Data channel security annotation derivation rules DL^T
1	$\forall (p_1, p_2), \langle (p_1, p_2), r, + \rangle \notin DL, \langle (p_1, p_2), r, - \rangle \notin DL,$ $\langle p_1, r, a_1 \rangle = \text{getPortSecAnnot}(p_1, \dots),$ $\langle p_2, r, a_2 \rangle = \text{getPortSecAnnot}(p_2, \dots),$ $a_1 = a_2 = + \Rightarrow \langle (p_1, p_2), r, + \rangle$
2	$\forall (p_1, p_2), \langle (p_1, p_2), r, + \rangle \notin DL, \langle (p_1, p_2), r, - \rangle \notin DL,$ $\langle p_1, r, a_1 \rangle = \text{getPortSecAnnot}(p_1, \dots),$ $\langle p_2, r, a_2 \rangle = \text{getPortSecAnnot}(p_2, \dots),$ $a_1 = a_2 = - \Rightarrow \langle (p_1, p_2), r, - \rangle$

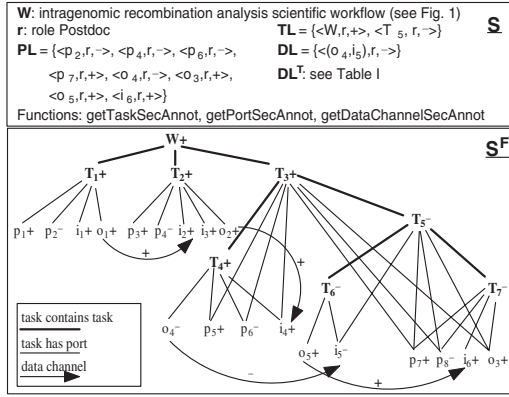


Figure 3. Security spec. S and full security spec. S^F for workflow W and role $Postdoc$.

The three defined functions, $getTaskSecAnnot$, $getPortSecAnnot$, and $getDataChannelSecAnnot$, answer the question of how full security specifications for a workflow W can be derived from partial security specifications TL , PL , and DL for W , and rule table DL^T . In the following, we refer a tuple that includes W , TL , PL , DL , DL^T , and the three functions as a *security specification* S . We denote *full security specification*, that contains explicit security annotations for all tasks, ports and data channels in a given workflow, as S^F and S^F can be easily derived from S . In Fig. 3, we show security specification S and full security specification S^F for our sample intragenomic recombination analysis scientific workflow W (see Fig. 1) and role $Postdoc$. S^F is represented as a graph and is computed by calling the corresponding functions on each task, port and data channel of W . Next, we address the important consistency problem of security specifications.

Security specification consistency. We define the notion of a security specification consistency in terms of consistency constraints, such that the security specification is *consistent* if it does not violate any of the consistency constraints in the system. The first consistency constraint that our model supports is the *data channel constraint*, which restricts that the two ports connected by any data channel must have the same security annotations, i.e., $\forall (p_1, p_2), \langle p_1, r, a_1 \rangle \in S^F, \langle p_2, r, a_2 \rangle \in S^F, a_1 = a_2$. The

rationale for this constraint is to avoid the unintentional permission of accessing a sensitive data product in the situation in which the data product is accessible via one port (with + annotation) and inaccessible via another port (with - annotation) of some data channel. It is important to report such a situation to the system security administrator to prevent the unauthorized data access. This is the only mandatory constraint in our provenance access control model.

Another kind of consistency constraints supported by our model is the so called *separation of duty constraints*, which restrict the exclusive choice of accessing two different ports. For example, for some scientific workflow, it may be important to ensure that if a user can access either port $t_i.o$ or port $t_j.o$, but never both.

Our sample security specification (see Fig. 3) for the intragenomic recombination analysis scientific workflow is consistent, since the data channel constraint holds for each data channel and we do not define any other constraints at this point. When the system encounters an inconsistent security specification that violates one or more of the consistency constraints, the administrator is required to either change the security specification or relax the constraints.

5 Security Views of Provenance

Our approach for enforcing security specification for scientific workflow run provenance is based on the innovative notion of *security views*. A security view of provenance is a restricted view of the recorded scientific workflow run provenance consisting of all and only the information that the users are authorized to access.

Before we formalize and incorporate the security view notion into our provenance model, consider the implication of security annotations of data channels and their associated ports on the accessibility of provenance. A consistent specification has four cases of security annotations for a data channel and its associated ports as shown in Fig. 4; other cases lead to inconsistent specifications due to the mandatory data channel constraint. In the first case, both ports and the data channel are annotated with + (e.g., $\langle o, r, + \rangle, \langle i, r, + \rangle, \langle (o, i), r, + \rangle \in S^F$), and therefore, corresponding task runs, data product, and their dependency should all be accessible. In the second case, both ports are accessible but the data channel is inaccessible. Therefore, we need to enforce that although the data product is accessible, the dependency is not. To achieve this, we introduce a copy d^c of data product d but with a new unique data product ID. In this way, the user is able to access the content of the product but not the dependency; our approach does not prevent a user to infer the dependency information by comparing the value of d and d^c . In the third case, both ports are inaccessible while the data channel is accessible; this implies that the data product is not accessible but

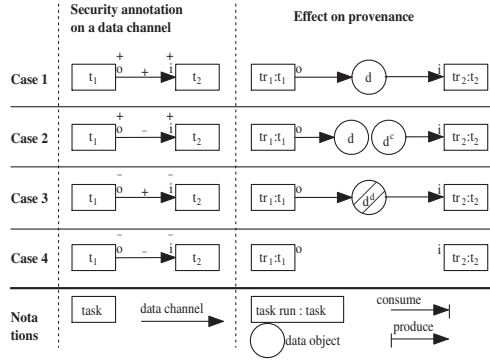


Figure 4. Implication of security annotations of a data channel and its associated ports on provenance accessibility.

the dependency is. As our current model does not permit a task run to be connected directly to another task run, we replace the data product with a dummy data product with a new unique product ID to maintain the dependency without authorizing the access to the data product itself. Finally, in the fourth case, both the data channel and its associated ports are inaccessible; this implies that both the data product and dependencies are inaccessible. Therefore, both the data product and its associated dependency edges are deleted in the provenance to be returned to a user.

The security view of a workflow run provenance ($wrid$, wid , \mathcal{D} , \mathcal{TR} , $Consume$, $Produce$) only includes a subset of data products in \mathcal{D} , as well as some data product copies and some dummy data products. Similarly, subsets of $Consume$ and $Produce$ are preserved and augmented with relationships for newly introduced data products (copies and dummies). The security view definition is as follows.

Definition 5.1 (Security view) A security view of a workflow run provenance is a tuple $(wrid, wid, \mathcal{D}', \mathcal{TR}, Consume', Produce')$, that is derived from a workflow run provenance $(wrid, wid, \mathcal{D}, \mathcal{TR}, Consume, Produce)$ and a consistent full security specification S^F for a role r , where

- (i) $\mathcal{D}' = \mathcal{D}^a \cup \mathcal{D}^c \cup \mathcal{D}^d$ is the set consisting of:
 - (1) all the data products $\mathcal{D}^a \subseteq \mathcal{D}$ consumed or produced by the workflow run and each $d \in \mathcal{D}^a$ is accessible to r via an accessible input or output port p , i.e., it is true that $\langle p, r, + \rangle \in S^F$ and $(d, tr.p) \in Consume$ or $(tr.p, d) \in Produce$ for some $tr \in (\mathcal{TR} \cup \{wrid\})$,
 - (2) data products \mathcal{D}^c and each $d \in \mathcal{D}^c$ is a copy of some $d' \in \mathcal{D}^a$, such that d and d' have the same values but different identifiers, and d' is consumed and produced by accessible ports that are connected by an inaccessible channel (see Case 2 in Fig. 4), i.e., $(d', tr_i.i) \in Consume$, $(tr_j.o, d') \in Produce$, $(tr_j.o, tr_i.i)$ is a data channel, $\langle i, r, + \rangle \in S^F$, $\langle o, r, + \rangle \in S^F$, and $\langle o, i, r, - \rangle \in S^F$, and

(3) data products \mathcal{D}^d and each $d \in \mathcal{D}^d$ is a dummy data product with a unique data product ID and each d corresponds to $d' \in \mathcal{D}$ that is consumed and produced by inaccessible ports connected by an accessible channel (see Case 3 in Fig. 4), i.e., $(d', tr_i.i) \in Consume$, $(tr_j.o, d') \in Produce$, $(tr_j.o, tr_i.i)$ is a data channel, $\langle i, r, - \rangle \in S^F$, $\langle o, r, - \rangle \in S^F$, and $\langle o, i, r, + \rangle \in S^F$.

(ii) $Consume' = Consume^a \cup Consume^c \cup Consume^d$ is the relationship set consisting of (1) set $Consume^a$ which is the projection of $Consume$ over $(\mathcal{D}^a - \mathcal{D}^c)$, where $\mathcal{D}^c \subseteq \mathcal{D}^a$ and each $d' \in \mathcal{D}^c$ has a copy $d \in \mathcal{D}^c$, (2) set $Consume^c$ which is the projection of $Consume$ over \mathcal{D}^c and data products in $Consume^c$ are substituted with their copies from \mathcal{D}^c , and (3) set $Consume^d$ which is the projection of $Consume$ over all the data products that have corresponding dummy data products in \mathcal{D}^d and data products in $Consume^d$ are substituted with their dummy versions.

(iii) $Produce' = Produce^a \cup Produce^d$ is the relationship set consisting of (1) set $Produce^a$ which is the projection of $Produce$ over \mathcal{D}^a and (2) set $Produce^d$ which is the projection of $Produce$ over all the data products that have corresponding dummy data products in \mathcal{D}^d and data products in $Produce^d$ are substituted with their dummy versions. \diamond

A sample security view for the intragenomic recombination analysis workflow run provenance and *Postdoc*'s security specification (see Fig. 3) is shown in Fig. 2.(b).

6 Provenance Querying with Security Views and Abstraction Views

Composite tasks, or subworkflows, serve as an important mechanism for abstraction. While exploring a workflow run provenance, a user may be interested in data products that have been produced or consumed by only certain task runs. Therefore, an abstraction mechanism is needed to enable a user to focus on only relevant provenance information. In this section, we briefly outline the notion of abstraction views and introduce a framework that integrates abstraction views and security views, such that a user can examine provenance information at different abstraction levels while respecting the security specification prescribed for her.

We define an *abstraction view specification* for a scientific workflow as a set of atomic and/or composite tasks of the workflow that a user chooses as relevant. Then, an *abstraction view* of provenance is a restricted view of the recorded scientific workflow run provenance consisting of all and only the information that is recorded for task runs of tasks chosen relevant in an abstraction view specification.

Thus, both security and abstraction views are restricted views (like filters) of a workflow run provenance that include restricted sets of data products, consume relationships, produce relationships, and so forth. Let $sv_r^S(wr)$ and

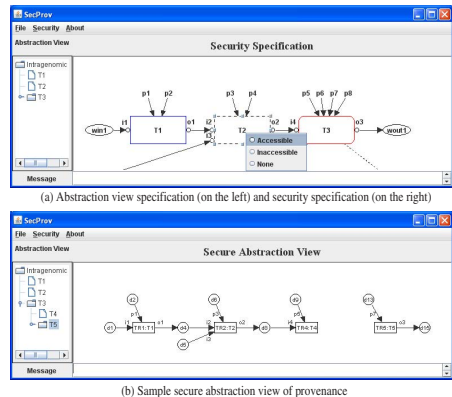


Figure 5. Screenshots of SEC PROV.

$av_u^A(wr)$ denote operations that compute a security view of workflow run provenance wr for role r and an abstraction view of workflow run provenance wr for user u of role r , respectively, where a security specification S for r and an abstraction view specification A for u are given. Then, in our integration framework, a *secure abstraction view* for user u with role r , is defined as $sv_r^S(av_u^A(wr))$ or $av_u^A(sv_r^S(wr))$.

We outline three approaches to provenance querying with security views and abstraction views. In the first, the most natural one, a provenance query q is evaluated over a secure abstraction view $sv_r^S(av_u^A(wr))$ of provenance. In the second approach, q is evaluated over a provenance wr and the result is filtered out based on security and abstraction view specifications S and A . In the last approach, q is rewritten into a ‘security and abstraction view aware’ query q' , and q' is evaluated over wr . For example, consider the following query q issued by a *Postdoc* user: return task runs that produced data product d_{15} (see Fig. 2). Using the first approach, we can retrieve $TR_5 : T_5$ (see Fig. 2.(c)) directly as the result of the query. Using the second approach, we can retrieve $TR_5 : T_5$, $TR_7 : T_7$, and $TR_3 : T_3$ (see Fig. 2.(a)) and filter out $TR_7 : T_7$ and $TR_3 : T_3$, since the tasks T_7 and T_3 are not part of the *Postdoc*’s abstraction view specification. Finally, using the third approach, we can rewrite q to return task run tr that produced data product d_{15} , such that tr does not execute T_7 or T_3 and tr ’s output port is accessible with respect to the security specification.

7 SEC PROV Prototype

We developed the SEC PROV prototype to validate the effectiveness of our approach to secure provenance querying with integrated security and abstraction views. We used Java and JGraph to implement a GUI for assigning security and abstraction specifications and XSB Prolog to implement our algorithms. Fig. 5 shows two screenshots of SEC PROV. In the upper one, an abstraction view specifica-

tion is selected (on the left) based on the task hierarchy of a workflow. A workflow (on the right) is annotated at the levels of task, port, and data channel to create a security specification. The lower screenshot shows a secure abstraction view of a workflow run provenance (on the right). A user can select different abstraction levels from the left panel to examine different abstraction views of the same workflow run provenance. Each abstraction view is *secure*.

8 Conclusions and Future Work

In this work, we studied the problem of protecting scientific workflow provenance, including both data products and their provenance. First, we formalized scientific workflow provenance that builds the basis for querying and access control. Second, we proposed a security specification mechanism for provenance at various granularity levels and the derivation of a full security specification based on inheritance, overriding, and conflict resolution rules. Third, we proposed the notion of security views of provenance to serve as the mechanism for enforcing security specification for scientific workflow provenance. Fourth, we studied a framework that integrates abstraction views and security views such that a user can examine provenance at different abstraction levels while respecting the security policy. Finally, we developed the SEC PROV prototype to validate the effectiveness of our approach. Currently, we are working on the design of algorithms to efficiently derive security views of provenance and would like to incorporate our findings into our provenance management system RDFPROV.

References

- [1] Workflow security consideration - white paper. In *Workflow Management Coalition*, February 1998. WPMC-TC-1019.
- [2] G. Ahn, R. Sandhu, M. Kang, and J. Park. Injecting RBAC to secure a web-based workflow system. In *Proc. of the fifth ACM Workshop on RBAC*, pages 1–10, 2000.
- [3] J. Alhiyafi, C. Sabesan, S. Lu, and J. Ram. RE-COMBFLOW: A scientific workflow environment for intragenomic gene conversion analysis. *Int. J. of Bioinformatics Research and Applications*, 2008. In press.
- [4] V. Atluri and W. Huang. An authorization model for workflows. In *Proc. of the fourth European Symposium on Research in Computer Security*, pages 44–64, 1996.
- [5] V. Atluri, W. Huang, and E. Bertino. A semantic-based execution model for multilevel secure workflows. *Journal of Computer Security*, 8(1), 2000.
- [6] V. Atluri and J. Warner. Security for workflow systems. *Handbook of Database Security Applications and Trends*, pages 213–230, 2007.
- [7] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999.

- [8] O. Biton, S. Cohen-Boulakia, S. Davidson, and C. Hara. Querying and managing provenance through user views in scientific workflows. In *Proc. of the Twenty-Fourth IEEE Int. Conf. on Data Engineering*, 2008. To appear.
- [9] R. Bose and J. Frew. Lineage retrieval for scientific data processing: a survey. *ACM Comput. Surv.*, 37(1):1–28, 2005.
- [10] R. Botha and J. Eloff. A security interpretation of the workflow reference model. In *Proc. of the Inf. Secur. - from Small Syst. to Management of Secure Infrastr.*, pages 43–51, 1998.
- [11] R. A. Botha and J. H. P. Eloff. Separation of duties for access control enforcement in workflow environments. *End-to-end Security*, 40(3), 2001.
- [12] U. Braun and A. Shinna. A security model for provenance. Technical Report TR-04-06, Harvard University, 2006.
- [13] A. Chebotko, X. Fei, C. Lin, S. Lu, and F. Fotouhi. Storing and querying scientific workflow provenance metadata using an RDBMS. In *Proc. of the IEEE International Workshop on Scientific Workflows and Business Workflow Standards in e-Science*, pages 611–618, 2007.
- [14] H. Chivers and J. McDermid. Refactoring service-based systems: how to avoid trusting a workflow service. *Concurr. Comput. : Pract. Exper.*, 18(10):1255–1275, 2006.
- [15] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang. Programming scientific and distributed workflow with Triana services. *Concurrency and Computation: Practice and Experience*, 18(10):1021–1037, 2006.
- [16] S. Cohen, S. Cohen-Boulakia, and S. Davidson. Towards a model of provenance and user views in scientific workflows. In *Proc. of the Third International Workshop on Data Integration in the Life Sciences (DILS)*, pages 264–279, 2006.
- [17] H. Davulcu, M. Kifer, L. Pokorny, C. Ramakrishnan, I. Ramakrishnan, and S. Dawson. Modeling and analysis of interactions in virtual enterprises. In *Proc. of the Ninth Int. Workshop on Research Issues on Data Engineering*, 1999.
- [18] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming J.*, 13(3):219–237, 2005.
- [19] D. Domingos, A. Silva, and P. Veiga. Workflow access control from a business perspective. In *Proc. of the Int. Conf. on Enterprise Information Systems*, pages 18–25, 2004.
- [20] I. Foster, J. Vöckler, M. Wilde, and Y. Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *Proc. of the Int. Conf. on Scientific and Statistical Database Management (SSDBM)*, 2002.
- [21] J. Freire, C. T. Silva, S. P. Callahan, E. Santos, C. E. Scheidegger, and H. T. Vo. Managing rapidly-evolving scientific workflows. In *Proc. of the International Provenance and Annotation Workshop (IPAW)*, 2006.
- [22] E. Gudes, M. Olivier, and R. Riet. Modelling, specifying and implementing workflow security in cyberspace. *Journal of Computer Security*, 7(4):287–315, 1999.
- [23] G. Herrmann and G. Pernul. Toward security semantics in workflow management. In *Proc. of the Thirty-First Annual Hawaii International Conference on System Sciences*, 1998.
- [24] W. Huang and V. Atluri. Analysing the safety of workflow authorization models. In *Proc. of the IFIP TC11 WG 11.3 Int. Working Conf. on Database Secur.*, pages 43–57, 1999.
- [25] W. Huang and V. Atluri. SecureFlow: a secure Web-enabled workflow management system. In *Proc. of the fourth ACM Workshop on Role-based Access Control*, pages 83–94, 1999.
- [26] P. Hung and K. Karlapalem. A secure workflow model. In *Proc. of the Australasian Information Security Workshop Conference on ACSW Frontiers*, 2003.
- [27] S. Kandala and R. Sandhu. Secure role-based workflow models. In *Proc. of the Fifteenth Annual Working Conf. on Database and Application Security*, pages 45–58, 2001.
- [28] M. Kang, J. Park, and J. Froscher. Access control mechanisms for inter-organizational workflow. In *Proc. of the sixth ACM Symposium on Access Control Models and Technologies*, pages 66–74, 2001.
- [29] D. Long, J. Baker, and F. Fung. A prototype secure workflow server. In *Proc. of the Fifteenth Annual Computer Security Applications Conference*, page 129, 1999.
- [30] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.
- [31] R. Martinho, D. Domingos, and A. Rito-Silvas. Supporting authentication requirements in workflows. In *Proc. of the Eighth Int. Conf. on Enterprise Inf. Syst.: Databases and Information Systems Integration*, pages 181–188, 2006.
- [32] S. Miles, P. Groth, M. Branco, and L. Moreau. The requirements of recording and using provenance in e-science experiments. *Journal of Grid Computing*, 2006.
- [33] T. M. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, R. M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.
- [34] R. Sandhu. Transaction control expressions for separation of duties. In *Proc. of the Fourth Computer Security Applications Conference*, pages 282–286, 1988.
- [35] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.
- [36] Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, 2005.
- [37] V. Tan, P. Groth, S. Miles, S. Jiang, S. Munroe, S. Tsasakou, and L. Moreau. Security issues in a SOA-based provenance system. In *Proc. of the third Int. Provenance and Annotation Workshop (IPAW)*, 2006.
- [38] J. Wainer, P. Barthelmess, and A. Kumar. W_RBAC - a workflow security model incorporating controlled overriding of constraints. *Int. J. Cooperative Inf. Syst.*, 12(4):455–485, 2003.
- [39] J. Warner and V. Atluri. Inter-instance authorization constraints for secure workflow management. In *Proc. of the eleventh ACM Symposium on Access Control Models and Technologies*, pages 190–199, 2006.
- [40] S. Wu, A. Sheth, J. Miller, and Z. Luo. Authorization and access control of application data in workflow systems. *Journal of Intelligent Information Systems*, 18(1):71–94, 2002.
- [41] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, I. Raicu, T. Stef-Praun, and M. Wilde. Swift: Fast, reliable, loosely coupled parallel computation. In *Proc. of the Int. Workshop on Scientific Workflows (SWF)*, 2007.