

# A Model for Sharing of Confidential Provenance Information in a Query Based System

Meiyappan Nagappan and Mladen A.Vouk

North Carolina State University,  
Raleigh, NC 27695, USA  
{mnagapp, vouk}@ncsu.edu

**Abstract.** Workflow management systems are increasingly being used to automate scientific discovery. Provenance meta-data is collected about workflows, processes, simulations and data to add value. This meta-data and provenance information may have as much value as the raw data. Typically, sensitive information produced by a computational processes or experiments is well guarded. However, this may not necessarily be true when it comes to provenance information. The issue is how to appropriately share confidential provenance information. We present a model for sharing provenance information when the confidentiality level is dynamically decided by the user. The key characteristic of this model is the *Query Sharing* concept. We illustrate the model for workflows implemented using provenance enabled Kepler system.

**Key words:** Provenance, Confidentiality, Workflow management tools

## 1 Introduction

The provenance and meta-data collection in workflow support systems can be flow based, annotation based, or a combination of both [4]. Workflow systems execute scientific simulations and secure the output data in order to maintain confidentiality/privacy and ownership of the sensitive data. Most of them, however, do not have good mechanisms in place for maintaining the confidentiality of provenance information. Here we use the term "confidentiality" as defined in ISO/IEC-17799[11]:*ensuring that information is accessible only to those authorized to have access.* Building the provenance collection system with such a mechanism should be the priority from the very beginning [13]. Security and confidentiality must be considered in an integrated context. For example, one must implement security to ensure confidentiality of the information. Security is a process or tactics that ensures that the desired level of confidentiality can be an outcome [9].

While confidentiality of provenance information is important, so is the sharing of that information among collaborators. Most of the scientific projects are highly collaborative projects. Often the collaborators are in different research labs in the country, and sometimes even in different countries. They may be working together using the same approach, or may be taking different approaches,

to scientific discovery. Provenance data throws a lot of light onto a particular scientific problem, process, and the data it produces. It can help discover a knowledge nugget that needs to be shared among all collaborators, and some that need to be shared sparingly. Hence provenance information is unlike any other data-centric application. Assurance of confidentiality, should not restrict the sharing of provenance data with trusted collaborators. For example, let's assume that scientist A is the owner of runs 1,2, and 3 of a scientific simulation. Each of the runs produce a set of provenance information viz. P1, P2 and P3 respectively. User A wants to share subsets of P1, P2 and P3, with Scientists B and C. Each of these subsets can be different. An appropriate mechanism should enable easy sharing of this information either on a per run basis, or on a per user group basis, or individually. The goal of current work is to develop a model, in the context of provenance for scientific simulations, that

- Enables an easy sharing of provenance data.
- Does not compromise the confidentiality of the provenance data.
- Allows for dynamic changes in the confidentiality levels.

The specific focus is on systems that uses Kepler [1] for scientific workflow automation and management. Kepler workflows are composed of a set of actors (processes) forming, in more complex situations, generalized activity networks [3], [8]. The order of execution of these actors depends on the nature of the problem (workflow) being solved and the Model of Computation (MoC) used to execute the workflow [6], [12]. These MoC's are called directors in Kepler. One version of Kepler implements a provenance collection mechanism [3], [12]. We describe it in more detail in Section 2. In section 3 we describe an approach to provenance data collection that allows for appropriate sharing of confidential parts of that information. Section 4 discusses issues that remain open, and concludes the paper.

## 2 Provenance in Kepler

The workflow support system we use is Kepler [1]. There is a version of Kepler that directly supports provenance recording. The Kepler Provenance Recorder (PR) is described in [3], [6], [12]. PR implements the Read-Write-State reset (RWS) trace information (flow-based provenance), first introduced in [6]. PR captures the flow of data objects between the ports of actors. PR captures reads (consumer), and writes (emitter) of an actor as well as events such as the 'flushing' of the state of actors. The process generates a unique token id for every token consumed/emitted by an actor. Each execution of the workflow is assigned a unique id. Provenance capture model and implementation is described in section 3. The PR is a passive mechanism that is designed such that we need not edit any of the Kepler actors. It is similar to a Kepler director in that it is configured in the same way, but it is different in the fact that it does not control the workflow, rather it just listens to it (in that sense it is more akin to Kepler debugging facilities that allow detailed workflow tracing). When the Kepler PR

actor is included into a workflow, it automatically collects the provenance data by listening to the ports of all the actors in the workflow.

We propose to slightly modify the RWS relational tables [6] in order to achieve our goal. The relational tables in our model are: *usersTable* (*username, workflow name, run id, annotation*), *actorTable* (*run id, actor id, port id, annotation*), *traceTable* (*port id, token, event, annotation*), *tokenTable* (*token, object, annotation*), *objectTable* (*object, value, type, annotation*). This is very similar to the provenance relations described in [12]. The difference is that here we have introduced an annotation field in each relation and the *usersTable* relation for keeping track of the owner of a run. This enables capture of the provenance that was recorded by their RWS PR. However, the annotation field in each of the above relations will get its value from the user only, and not from Kepler.

### 3 Model and Implementation

In this section we describe a model that can accommodate the sharing of confidential provenance information in a scenario where the confidentiality level changes dynamically. We reach the goal stated previously through five sub goals: Ownership, Editing, Annotation, Sharing and Audit. The Model is implemented in a system being built by DOE Scientific Data Management Center [2], [17]. Fig. 1 illustrates the architecture. The details of the figure are discussed in the text that follows.

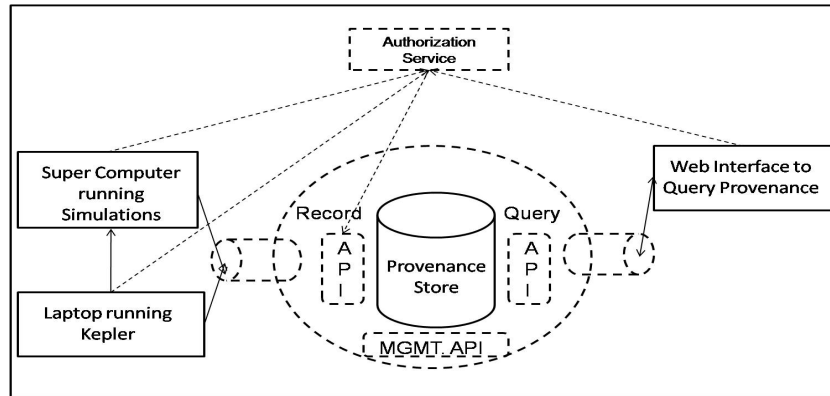


Fig. 1. Top-level architecture view of our Implementation Model.

#### 3.1 Sub Goal 1: Data Ownership

We need to ensure that the person who generates the simulation data would continue to be the owner of the provenance data generated at the time the

simulation results are generated. To achieve this we use a three tiered (Client-(Application Logic)-Database) approach, and we build role based access control at the application logic layer. Provenance is collected through an API. The client is either the workflow management system itself, or the scripts and simulation applications running in the super computer. The recording API is the application logic layer that makes sure that the information from the clients is stored in the appropriate tables of the schema. Also there is an authentication service that verifies the user. Thus the data in the provenance schema is indexed by the authenticated user and a particular run id.

In order to view this data, we use a Web Application (WA). The client is the Web Interface (WI), and the Query API is the application logic. Here too we have an authentication service that verifies the user. Since the data is indexed by user, the query API is able to fetch a particular user's data. Each user has multiple runs under their username [5]. They pick the one they want to see the provenance data for. The query API will execute some of the default queries only on the dataset available for this particular run and for this particular user. Then the user who launched this query can refine the dataset as they please. If the user annotates the data and generates additional provenance, the user becomes the owner of that provenance information. This is similar to the role based access available in many database applications.

### 3.2 Sub Goal 2: Editing and Audit Trail

The access privileges to the database are restricted to prevent any unwarranted use of provenance data. Once the data is created, users cannot delete or modify any of the provenance information. But, they can modify annotation fields of the records in the provenance relations and the *queryTable* relation. The prevention of edits to the data is important for audit purposes, but also to maintain consistency. A collaborator should get the same outputs when he executes shared queries at different times. This is possible only if the owner is not allowed to make any changes to the provenance data. Only an administrator may delete this data. But even there, all superuser actions must be logged. A trail must be maintained for audit purposes. A system that tracks the changes to database relations would suffice for this purpose. Even though users cannot delete the provenance data they can, through the annotation fields, comment on the accuracy of the provenance data, and they can always add additional provenance information. Access privileges found in modern database systems is sufficient for realization of this sub goal.

### 3.3 Sub Goal 3: Data Annotation

The annotation field in the *queryTable* relation, and the other relations may be updated from the web application. They can be used to annotate the datasets regarding any inaccuracies or interesting findings. When users share, they can choose not to share the annotations. Annotations, of course, are a form of user specified meta-data. Thus annotations help in differentiating useful provenance

information without deleting the inaccurate data, as well as in adding user level meta data to the results.

### 3.4 Sub Goal 4: Data Sharing

The ability of researchers to share their provenance data with their colleagues in an easy manner can be as important as the confidentiality of this data. We introduce the notion of **query sharing** for this purpose. An authenticated user will be able to see data in the web interface. If the scientist finds the provenance data that they currently see as interesting, and wants to share it, he/she can do so by saving the query that created that dataset and sharing it with collaborators. The approach is: What You See Is What You Want to Share(WYSIWYWTS): The relation *queryTable* (*Query ID*, *Saved by*, *Saved for*, *Query*, *Timestamp*, *Allow Cascading*, *Revoke Active*) is used to save the queries and the relation *annotTable* (*UserID*, *Query ID*, *Annotation*, *Viewable*) is used to store the annotation for the data set that is to be shared. The web application assures that the query that created the dataset is saved. The scientist who wants to share the data can choose to annotate the dataset. For example if User U1 wants to save the dataset of run Rx and share it with User U2, then the entry in the *queryTable* relation would look like this: (*QID*, *U1*, *U2*, *Select Query similar to the above one*, *timestamp*, *A binary 1/0 for whether cascading of should be allowed or not*, *A binary 0/1 if the collaborator has access currently or not*). They can also choose to annotate the dataset by making an entry in the *annotTable* relation. A typical entry would be (*UID*, *QID*, *Any relevant annotation to the dataset*, *A binary 1/0 if the collaborator should see the annotation or not*)

The collaborator, when logged in to the web application, will be able to see, given the right access, all the data in the hierarchical fashion discussed above. In a separate tab they will also be able to see the queries (possibly not the actual query itself, but rather the *Query ID*, *annotation*, *timestamp*, *saved by*, *and saved for* columns). This is done by the web application which will pass the user-id of the currently logged in user and one of the Query API's will fetch all the entries in the *queryTable* where the user id matches the user id in the saved by or saved for columns and the corresponding entries in the *annotTable* if the *viewable* attribute is set to 1.

A user selects a query to run. Then the user sees the data. If the user wants to refine the query, the user can do so. The queries are built for refining acts only on the dataset that was shared and not on the whole database. By abstracting the execution of the query to an API that manages data to be accessed, we restrict the user from seeing data that is not meant for the user. As an extra measure, we can encrypt the query attribute of the *queryTable* relation in the database to protect the data in the case the *queryTable* in the database gets compromised. The saved query is not shown to the users and can be use in an execute-only mode.

The *cascade* attribute in the *queryTable* is used to either allow or deny a collaborator from passing the shared information to another person. The *revoke* attribute is used to remove access to a particular query for a particular user by

the owner of that query. Users can save interesting subsets of the provenance information for themselves. In this case *saved by* and *saved for* attributes in the *queryTable* will be the same as the owner. A collaborator, in our model, is not able to annotate individual pieces in a shared dataset. Collaborators can only annotate the dataset as a whole. For this they would make an entry into the *annotTable*. If they want the owner to view their annotations then they would set the *viewable* attribute to 1.

A user can thus give others access to all his/her data, or to a particular dataset, or just a particular part of a particular dataset. Thus the granularity at which the user wants to share the data depends on the user. Each time the user can decide to share a different piece of the dataset without editing any rule set for access control. Also the entire chosen dataset can be shared by saving a single query instead of sharing each record in the dataset or saving a copy of the whole dataset.

The query sharing concept is very similar to stored procedures. Both have the same overhead. However, by saving queries in a table we extend the scope to add more meta data to the saved query which would not be possible with stored procedures. The salient features of data sharing concept given the constraints are:

- Dynamic sharing: This constraint prevents building the logic of the query into the application layer. Since applications are static, a user cannot easily add a new query for a subset of the dataset that the user wants to share.
- Sharing large data sets: Since the datasets are large and so are the subsets of information, it is difficult to share copies with the collaborators.
- Sharing subsets of data: The data that is shared are subsets. If we were to use a high level language to control access, then the users would have to individually pick the cells in the subset that they want to share. The large size, and the fact that they can be thought of as an atomic piece of information, would make the process of individually indexing cells unnecessary.

Thus given these constraints our query sharing solution would scale well, and is flexible.

### 3.5 Sub Goal 5: Data Audit and Verification

The inclusion of auditing capabilities into the model is to check on accuracy and integrity of the data. Auditors are users who can view the original data, provenance data, and annotations. They are provided with sufficient access privileges to see all information needed for auditing. The provenance relations have annotations in them for the auditors to verify the data against. Also, the *queryTable* relation has timestamps in it for the auditors to find any discrepancy in the sharing of the data. Finally, since all the edits to the database have a trail, any missing information can be accounted for by the auditors.

## 4 Discussion and Conclusions

The implementation model we have presented has some limitations. For example, because it is query-centric only provenance systems that store data in databases and fetch them using queries can use this approach. Another limitation is that the model requires automatic run-time provenance collection. The user is still allowed to annotate the data and create provenance manually, but the user is not allowed to modify original records. This assures a considerable level of integrity for the originally (and automatically) collected data, but perhaps less of integrity for manually added annotations. Also to be noted is that only owners of provenance data can freely annotate any part of the dataset. The collaborators can only annotate the entire shared subset and not any individual part in it. Even in automatic provenance collection systems we could face the security and privacy problems stated in [7].

With more emphasis being laid on provenance data collection in scientific workflow applications [14], [15], the issue of sharing provenance with varying levels of confidentiality becomes increasingly important. We believe that the simple model described in this paper is able to ensure considerable level of confidentiality in provenance data as well as sharing of it amongst trusted collaborators. The data sharing technique described in this paper allows users to change the level of collaboration dynamically. This model addresses the integrity, confidentiality, availability, and ownership responsibility issues raised in [10], [18]. But this model does not address the issues arising due to long term storage and scalability of provenance data discussed in [18]. There are some systems like PASS [16] that are used to provide security using provenance data. But there is not much research in the field of securing provenance data, providing confidentiality and enforcing privacy policies.

### Acknowledgments

This project is funded in part by the DOE SciDAC grant DE-FC02-01ER25809 and the IBM SUR program. We would like to thank the SPA group of the SDM Center for their collaboration. We would also like to thank Drs. Yu and Sherriff, and graduate students Vinod Arjun, Lucas Layman, Aaron Massey, and Andy Meneely for discussions and insights into the various aspects of this research.

### References

1. Kepler development and download site. <http://kepler-project.org/>
2. Scientific Data Management Center, <http://sdm.lbl.gov/sdmcenter/index.html>
3. Altintas, I., Barney, O. and Jaeger-Frank, E.: Provenance Collection Support in the Kepler Scientific Workflow System. LNCS, Volume 4145 (Provenance and Annotation of Data). pp. 118-132. Springer Berlin / Heidelberg (2006)
4. Barga, R.S. and Digiampietri, L.A.: Automatic Generation of Workflow Provenance. LNCS, Volume 4145 (Provenance and Annotation of Data). pp. 1-9. Springer Berlin / Heidelberg (2006)

5. Barreto, R., Critchlow, T., Khan, A., Klasky, S., Kora, L., Ligon, J., Mouallem, P., Nagappan, M., Podhorszki, N. and Vouk, M.: Managing and Monitoring Scientific Workflows through Dashboards. Poster # 93, at Microsoft eScience Workshop Friday Center, University of North Carolina, Chapel Hill, NC, October 13 - 15, (2007), pp. 108
6. Bowers, S., McPhillips, T., Ludeascher, B., Cohen, S. and Davidson, S.B.: A Model for User-Oriented Data Provenance in Pipelined Scientific Workflows. LNCS, Volume 4145 (Provenance and Annotation of Data). pp. 133-147. Springer Berlin / Heidelberg (2006)
7. Braun, U., Garfinkel, S., Holland, D.A., Muniswamy-Reddy, K.-K. and Seltzer, M.I.: Issues in Automatic Provenance Collection. LNCS, Volume 4145 (Provenance and Annotation of Data). pp. 171-183. Springer Berlin / Heidelberg (2006)
8. Elmaghraby, S.E.: Activity Networks: Project Planning and Congrol by Network Models, Wiley-Interscience, New York, NY,1977.
9. Griffiths, P.P. and Wade, B.W.: An authorization mechanism for a relational database system. ACM Transactions on Database Systems,(Sep 1976)., 1 (3). 242-255.
10. Hasan, R., Sion, R. and Winslett, M.: Introducing secure provenance: problems and challenges Proceedings of the 2007 ACM workshop on Storage security and survivability, ACM, Alexandria, Virginia, USA, (2007). pp 13-18
11. ISO/IEC 17799. Information technology Security techniques Code of practice for information security management. <http://www.iso.org/iso/en/prods-services/popstds/informationsecurity.html>, 2000. Rev. 2005.
12. Ludaescher, B., Podhorszki, N., Altintas, I., Bowers, S. and McPhillips, T.: From Computation Models to Models of Provenance: The RWS Approach. Concurrency and Computation: Practise and Experience, 20 (5). 507 - 518.
13. McGraw, G.: Building secure software: better than protecting bad software. Software, IEEE, 19 (6). 57-58.
14. Moreau, L. and Foster, I.: Intl. Provenance and Annotation Workshop (IPAW). in LNCS 4145, (Chicago, May 2006), Springer.
15. Moreau, L. and Ludaescher, B.: Concurrency and Computation: Practice & Experience Special Issue on the First Provenance Challenge. Wiley, 2007.
16. Muniswamy-Reddy, K.-K., Holland, D.A., Braun, U. and Seltzer, M.I.: Provenance Aware Storage Systems. in Proceedings of the 2006 USENIX Annual Technical Conference, (June 2006). pp 4-4
17. Nagappan, M., Altintas, I., Chin, G., Crawl, D., Critchlow, T., Koop, D., Ligon, J., Ludaescher, B., Mouallem, P., Podhorszki, N., Silva, C. and Vouk, M.: Provenance in Kepler-based Scientific Workflow Systems. Poster # 41, at Microsoft eScience Workshop Friday Center, University of North Carolina, Chapel Hill, NC, October 13 - 15, (2007), pp. 82
18. Tan, V., Groth, P., Miles, S., Jiang, S., Munroe, S., Tsasakou, S. and Moreau, L.: Security Issues in a SOA-Based Provenance System. LNCS, Volume 4145 (Provenance and Annotation of Data). pp. 203-211. Springer Berlin / Heidelberg (2006)