Provenance: Privacy and Security

Susan B. Davidson[1] and Sudeepa Roy[2]
[1]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA
[2]Department of Computer Science, Duke University, Durham, NC, USA

Synonyms

Origin, Lineage, Confidentiality, Integrity

Definition

Data provenance is information about the origins of data and its movement between databases and processes. It can be used to understand and debug the process by which data was obtained and transformed, to ensure reproducibility of results, and to establish trust. Provenance therefore has implications for both the security and privacy of the associated data. As metadata, there are also security and privacy concerns associated with provenance itself, including the integrity, confidentiality, and availability of provenance information.

Historical Background

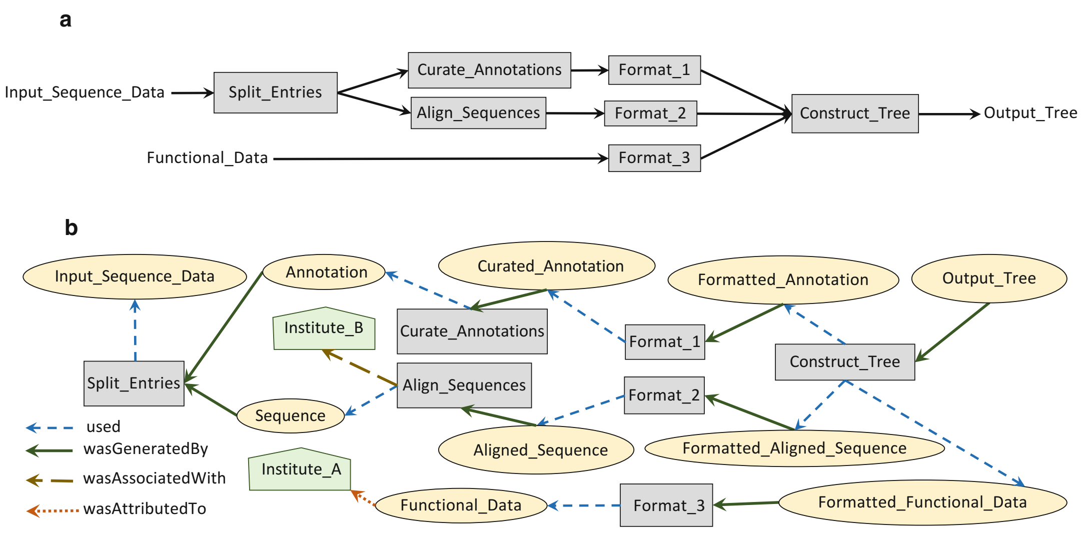
Tracking the provenance of data within a system includes (i) capturing metadata associated with raw data that is input to the system and (ii) details of computations that transform the raw data to create new information, e.g., the sequence of steps or processes, parameter settings (in a program), and inputs and outputs of each step. Queries over provenance typically answer a question of the form *What are the input data and/or processing steps that led to the creation or modification of a given data item?*.

Since provenance is itself data (metadata), at a first glance, it might seem that the security and privacy of provenance records are already addressed by the vast amount of work that has been done in these domains. However, the application of these techniques for provenance is not straightforward due to the interaction between the entities, agents and activities involved in the data generation process, and the types of questions that are asked of provenance data.

As an example, *differential privacy* [9] is a widely accepted notion of privacy which focuses on aggregate queries and obscures sensitive information about individuals in the aggregates by adding random noise. How to use these ideas in the context of data provenance is not clear: (i) queries over provenance information are typically *not* aggregate queries, and (ii) adding random noise to provenance makes it of little use for purposes of reproducibility and trust.

Another subtlety is that the privacy levels of different components of provenance may vary both within and across applications [3]. For example, the decision of whether to accept a paper for a refereed journal is based on a set

L. Liu, M.T. Özsu (eds.), *Encyclopedia of Database Systems*,
DOI 10.1007/978-1-4899-7993-3_80748-1



Provenance: Privacy and Security, Fig. 1 (**a**) A workflow specification, (**b**) the provenance of an execution of the workflow using PROV

of reviews. While the reviews can be accessed by the authors, the reviewers themselves should not be, making the provenance information more sensitive than data. Similarly, the decision of whether to accept a student to a graduate program is based on a set of letters of recommendation. While the identity of the recommenders is known to the student since they provide the names, the recommendations should not be revealed to the student. In this case, the data is more sensitive than provenance.

Note that in both of these applications, the processes by which the decision is made, including the activity nodes and connections between nodes, are not confidential. In other applications, however, an activity may represent proprietary code or the connections between activities (e.g., the sequence of steps taken in a biological/clinical experiment) may be required to remain confidential. Privacy and security concerns in provenance therefore require careful study at the level of data, agents, activities, and the connections between activities.

Scientific Fundamentals

Provenance is typically studied in the context of *data flow* applications, such as databases, workflows, or storage systems. Such applications can be modeled as a directed acyclic graph in which the nodes represent atomic processes (e.g., a program or a procedure), and the edges represent the data flow between processes. As an example, Fig. 1a shows the *specification* (i.e., the template) of a simplified phylogenetic tree construction workflow. Here the input sequence data flows into a process called `Split_Entries`, which splits the input into annotation and sequence information. The annotations are sent to process `Curate_Annotations` and sequences to `Align_Sequences`. The curated annotations, aligned sequences, as well as input functional data are then formatted and combined in `Construct_Tree` to create the final output tree.

In an *execution* of this data flow, data (e.g., in the form of files) will be passed from one process to others when the processes are executed. Provenance in such an execution can be modeled using PROV, a W3C standard that has been proposed to unify provenance representations and enable their interchanges on the Web (PROV Model Primer: http://www.w3.org/TR/prov-primer/.) [14]. There are three types of nodes in PROV: (i) *entity nodes*, denoting individual data items; (ii) *activity nodes*, denoting processes that consume one or more data items and produce one or more data items; and (iii) *agent nodes* that are *attributed to* one or more entity nodes or are *associated with* one or more activity nodes. Edges in this model include `used` (from an activity to an entity), `wasGeneratedBy` (from an entity to an activity), `wasAttributed to` (from an entity to an agent), and `wasAssociatedWith` (from an activity to an agent). Each data item is either an *initial input* (i.e., it was not generated by any activity, e.g., `Input_Sequence_Data`) or was generated by exactly one activity (e.g., `Curated_Annotation`). Similarly, if a data item is a final output (e.g., `Output_Tree`), it is not used in any activity; otherwise, it can be used in one or more activities.

A PROV graph for an execution of the workflow specification in Fig. 1a is shown in Fig. 1b. Its structure is similar to the specification, but additional nodes are present: ovals are used to represent data items as entities (e.g., `Annotation`), and pentagons represent agents (e.g., `Institute_A`). Edges also have different meanings and use the reverse orientation. Note that each execution of the workflow would generate a new provenance graph.

The provenance for a data item is the subgraph that is reachable from it and includes *provenance records* (nodes in the PROV graph) as well as information about *connectivity* between records (edges in the PROV graph). For example, Fig. 1b is the provenance of `Output_Tree`, and the provenance of `Annotation` is the path `wasGeneratedBy`, `Split_Entries`, `Used`, and `Input_Sequence_Data`.

Desiderata of Provenance Privacy and Security

Secure and privacy preserving provenance must ensure the following [5, 10]:

- **Confidentiality:** Unauthorized parties should not have access to provenance records or connectivity information.
- **Availability:** Authorized parties should have access to provenance records and connectivityinformation.
- **Integrity:** Adversaries cannot tamper with individual provenance records or connectivity information without being detected.

Provenance privacy requires ensuring confidentiality while maintaining availability, whereas provenance security requires ensuring availability while maintaining integrity.

Provenance Privacy

While capturing and publishing provenance enable transparency, trust, and reproducibility of results, it may expose sensitive information contained in the provenance graph. Provenance privacy therefore concerns data (entities and agents), activities (e.g., modules or processes), and structure (relationships between entities, agents, and activities) [7].

Data privacy. The first and foremost concern is ensuring the confidentiality of data items while publishing provenance. For example, in the refereed journal paper review example, the author should not be able to see information about agents who are reviewers, but should be able to see that of the agent who is the editor, the reviews, and the review process itself. All of the provenance information should be visible to the editor and his/her superiors. Assuming a double-blind review process, reviewers should be able to see the reviews, information about the editor and reviewer agents, as well as the review process, but not information about the author agent. Note that it is not necessary to disallow individuals from seeing the entire provenance graph, which would violate availability, as long as they are prevented from accessing confidential components. Most of such privacy concerns (involving entities and agents) can be managed using standard role-based access control techniques, i.e., by specifying categories of users and access privileges against the application specification.

Module privacy. Ensuring the confidentiality of certain activities contained within provenance is trickier. For example, if an activity node represents the execution of a proprietary software, it is not enough to disallow access to the provenance record for that activity in all executions. If a user is allowed to see provenance records of input (`used`) and output (`wasGeneratedBy`) entities to that activity, and the software is run over a large fraction of possible inputs, then the user might be able to emulate the proprietary software. To avoid this, some of the inputs and/or outputs to that activity across all executions must also be hidden, compromising availability to guarantee confidentiality of the activity.

Structural privacy. Publishing provenance information can also reveal confidential connectivity information among entities, agents, and activities. For example, suppose it is known that a set of modules is typically used to perform a particular analysis in a workflow (e.g., for functional annotation [7]). These modules can be executed in different orders, sequentially or in parallel, and some of them may be skipped under certain conditions. The order of processing may significantly influence the efficiency of the overall analysis, as well as the overall cost. In the published provenance information, the workflow owner may be willing to reveal which modules were used, but the order in which they were executed could be confidential.

Provenance Security

Provenance security involves ensuring the integrity and availability of provenance records and connectivity. The threat is that adversaries may attempt to subvert provenance by altering, adding, or deleting either data, provenance records, or connectivity. Ultimately, this means that provenance must be captured and stored using trusted hardware.

However, provenance is frequently captured by database or workflow systems without such support or may travel across software domains. The focus therefore becomes one of ensuring that provenance records and connectivity are *tamper evident* and that a *trusted auditor* can verify their authenticity. Note that if the auditor does not have access privileges for the provenance records, they must be able verify authenticity without actually seeing the contents.

Mechanisms for Provenance Privacy and Security

As mentioned earlier, role-based access control is widely used to ensure the privacy of entities and agents, in mechanisms for addressing module and structural privacy, as well as in ensuring the security of provenance records themselves.

A formal model for module privacy based on l-diversity was given in [6]. Given a desired privacy level l, they considered the problem of determining what input data to hide to ensure that for every input x to the confidential module, the output is be indistinguishable from at least $l - 1$ other possible outputs. However, the problem becomes much more complex in the setting considered with provenance graphs, in particular when the module is part of a workflow and interacts arbitrarily with other modules, some of which may be public with known functionality (e.g., a sorting or reformatting module).

A common technique for addressing structural privacy is to use *composite processes*. In contrast to an atomic process, such as `Split_Entries` and `Align_Sequences` in Fig. 1a, a composite process is one which itself has structure. For example, the entire workflow in Fig. 1a could be a single composite process (say, `Construct_All`), which takes sequence and functional data as input and generates a tree as output. Composite processes form a basis for *views* of provenance information [1, 4] in which details of the composite process execution (a subgraph) are grouped within a single, composite node. A user without sufficient privileges would not be allowed to expand the composite node and see details contained within it.

Although confidentiality of structural information can be addressed using composite nodes, availability may be compromised, and the user may be able to infer incorrect connectivity. For example, if the `Curate_Annotations` and `Align_Sequences` in the PROV graph are put in a composite module, the user could infer a path from `Curated_Annotation` to `Sequence`, which is not present in the fully expanded PROV graph. Initial ideas of how to use composite node expansion and contraction are being developed for PROV [14], but more work remains to be done. Other approaches for addressing structural privacy include using a declarative framework that allows owners to specify which parts of the provenance graph are to be hidden or abstracted and checking user requests against the specification (PROPUB [8]) and using *surrogate* nodes and edges to protect sensitive graph components while maximizing graph connectivity [2].

Security mechanisms used to secure data can also be used for provenance, e.g., signatures, checksums, and signed hashes. These techniques can be enforced at the kernel, file system, or application layers and at different stages in a workflow as the data is curated, annotated, and queried [11, 12, 15]. However, the problem becomes much more difficult when provenance can cross multiple domain boundaries and can be passed through untrusted domain owners.

Key Applications

In addition to data flow applications like document management and scientific workflows, provenance privacy and security are important for a wide range of applications. Three of these are discussed below.

1. **Cloud computing and service-oriented architectures:** In recent years, cloud services have gained popularity due to their affordability and ease of use. In a cloud platform, resources (storage and computation) are managed by the cloud service provider, accessed by multiple users, and may be used to process and store sensitive information about individuals. In addition to ensuring the confidentiality of sensitive information, the identity of users may be confidential. Data-intensive collaborations on cloud computing therefore require ensuring the privacy and security of provenance.
2. **Networks:** In networked applications, the provenance of exchanged information may be used to judge its credibility. For example, in a military command and control application, information about location status, intelligence reports, and operational plans may be communicated between different units, and provenance may be used to judge its trustworthiness. In a network routing application, provenance may be used to identify faulty or misbehaving nodes and to assess damage such nodes may cause to the network. In such systems, care must be taken to ensure that provenance information is secure, i.e., that a compromised node cannot forge or tamper with provenance, or reveal the identities of other nodes in the network to unauthorized agents.
3. **Healthcare:** Privacy is a critical concern in healthcare applications, where patients, practitioners, and other people involved in the system of care must have authorized access to patient or treatment information. Enabling provenance in this domain can significantly increase trust in the data and be used to improve the quality of service. However, since provenance may leak information about patients and their treatment to unauthorized agents, stringent measures must be taken to ensure its privacy and security.

Future Directions *

There are several important directions for future research in provenance privacy and security. First, there is a gap between theory and practice in this area. For example, PROV has been proposed as a standard for provenance representation and exchange and is starting to be used in provenance-enabled systems. However, it lacks certain abstractions, such as composite modules, which are important for mechanisms which enable provenance privacy and security. On the other hand, the practical effectiveness of provenance privacy techniques has not been successfully evaluated due to the lack of real datasets and well-defined measures on availability and confidentiality requirements. It is important to narrow this gap in future research. Second, existing research on provenance security has focused on identifying requirements and has proposed solutions using standard techniques used for securing data. It would be interesting to understand whether there are novel security problems and solutions arising from the interplay between data and provenance, as there are with privacy. Third, privacy notions currently being used for provenance are fairly weak and make strong assumptions. For example, module privacy uses ℓ-diversity [13] and assumes no background knowledge of the adversary; it is implemented by hiding portions of the provenance information using access control techniques. However, there are much stronger notions of privacy such as differential privacy [9], which rely on adding random noise to the output of aggregate queries. It is important to understand how these techniques can be used in the context of provenance, where queries are typically not aggregates and reproducibility is essential.

Cross-References

- ▸ Data Provenance
- ▸ Provenance in Databases
- ▸ Provenance in Workflows
- ▸ Provenance Standards
- ▸ Randomization Methods to Ensure Data Privacy
- ▸ Role Based Access Control

Recommended Reading*

1. Bao Z, Davidson SB, Milo T. Labeling workflow views with fine-grained dependencies. PVLDB. 2012;5(11):1208–19.
2. Blaustein B, Chapman A, Seligman L, Allen MD, Rosenthal A. Surrogate parenthood: protected and informative graphs. PVLDB. 2011;4(8):518–25.
3. Braun U, Shinnar A, Seltzer M. Securing provenance. In: HOTSEC; 2008. p. 4:1–5.
4. Chebotko A, Chang S, Lu S, Fotouhi F, Yang P. Scientific workflow provenance querying with security views. In: WAIM; 2008. p. 349–56.
5. Cheney J. A formal framework for provenance security. In: CSF; 2011. p. 281–93.
6. Davidson SB, Khanna S, Milo T, Panigrahi D, Roy S. Provenance views for module privacy. In: PODS; 2011. p. 175–86.
7. Davidson SB, Khanna S, Roy S, Stoyanovich J, Tannen V, Chen Y. On provenance and privacy. In: ICDT; 2011. p. 3–10.
8. Dey SC, Zinn D, Ludäscher B. PROPUB: Towards a declarative approach for publishing customized, policy-aware provenance. In: SSDBM; 2011. p. 225–43.
9. Dwork C. Differential privacy: a survey of results. In: TAMC; 2008. p. 1–9.
10. Hasan R, Sion R, Winslett M. Introducing secure provenance: problems and challenges. In: StorageSS; 2007. p. 13–8.
11. Hasan R, Sion R, Winslett M. Preventing history forgery with secure provenance. Trans Storage 2009;5(4):12:1–43.
12. Lu R, Lin X, Liang X, Shen XS. Secure provenance: the essential of bread and butter of data forensics in cloud computing. In: ASIACCS; 2010. p. 282–92.
13. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M. L-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data 2007;1(1):3.
14. Missier P, Bryans J, Gamble C, Curcin V, Dánger R. ProvAbs: model, policy, and tooling for abstracting PROV graphs. In: IPAW; 2014. p. 3–15.
15. Zhang J, Chapman A, LeFevre K. Do you know where your data's been? – tamper-evident database provenance. In: SDM; 2009. p. 17–32.