Practical Provenance Privacy Protection

Ashish Gehani
SRI

Raza Ahmad*
SRI

Hassaan Irshad
SRI

Abstract

Data provenance records may be rife with sensitive information. If such metadata is to be shared with others, it must be transformed to protect the privacy of parties whose activity is being reported. In general, this is a challenging task. It is further complicated by properties of provenance that facilitate drawing inferences about disparate portions of data sets. We consider aspects of the problem, describe strategies to address the identified issues, and share our implementation of configurable primitives for practical application of provenance privacy protection.

1 Introduction

When data sets are shared, the corresponding provenance metadata can help analysts ensure the integrity of workflows and experimental protocols. The information allows parameter regimes, the sets of inputs, and intermediate information to be accurately replicated. Provenance is of particular utility for accurately identifying dependencies when sharing data and procedures with others. This metadata also has a wide variety of related applications including performing scenario analyses without needing to gather new observations, and narrowing the range of hypotheses when diagnosing problems in complex experimental environments.

Since provenance metadata can be used for a variety of purposes, the systems used to report such information may be configured to collect a wide range of attributes about the data, the processes used to manipulate it, and the agents that control the processes. Among these attributes are some that may be particularly sensitive, such as identity details in filesystem and document metadata.

When data sets are shared, privacy-violating inferences can be drawn. We explore approaches for preventing such breaches. When provenance from multiples sources is integrated [10, 13], the issue is exacerbated. If ontologies relating the schema and semantics of different components [6] are available or easily constructed, it may be possible to determine which elements in the provenance must be sanitized to satisfy the dual constraints of allowing specific provenance queries and disallowing privacy-violating inferences about elements in the metadata. In situations where no solution can satisfy the competing constraints, an alternate approach can provide auxiliary protection for subsets of the metadata using directed granular encryption.

2 Background

Schemes for querying provenance data have received considerable attention. Harvard's PQL [19] describes a language for querying provenance and leverages the query optimization principles of semi-structured databases. IBM researchers have proposed a provenance index that improves the execution of forward and backward provenance queries [20]. Query optimization techniques on compressed provenance data have also been considered [18].

While a number of efforts have studied the question of how to secure provenance [17, 31] and developed access control models for it [3, 28], most do not address sanitizing the provenance metadata so that it may be safely shared while still supporting queries over it. A notable exception is the graph transformation approach [5, 23]. These efforts provide stronger assurance than the sanitization transformer we describe. Differential privacy-based approaches [7] offer even stronger privacy protection. However, this comes at the cost of being limited to statistics calculated over the provenance, precluding responses involving complete subgraphs.

Our approach for provenance sanitization assumes a framework similar to that of the *privacy-preserving data publishing* literature [4, 9], where data originates from individual record owners and is collected by a publisher that releases a compendium to the public or specific recipients [9]. Mechanisms to downgrade information date back several decades. Adams [1] described three classes: (i) restricting the number of queries that can be made, (ii) perturbing the data before executing the query, and (iii) running the query on unmod-

*While visiting.

ified data, but then perturbing the output. Over the years, a range of techniques has been proposed for sanitizing data, including *generalization*, which coarsens, abstracts, or collects multiple data into equivalence classes; *suppression*, which removes records from the sanitized output; *swapping*, which interchanges attributes from different records; *randomization*, which adds random noise to perturb the data; and *multi-views*, which applies different variants of the previous techniques [4].

If protecting *quasi-identifiers* was the only goal, k-anonymity [30] would suffice. If the statistical similarity of sensitive attributes was the only concern, then l-diversity [22] could be used. We do not consider the case where the sanitization must be effected by untrusted entities, necessitating the use of cryptographic protocols [32].

3 Practical Challenges

We decompose the problem into different classes based on the properties of the metadata and the approaches that can be applied.

Heterogeneous provenance arises when data sets from multiple sources that use the same provenance data model but only partially overlapping annotation schema are integrated. Information used for data integration can be used to identify associations that an adversary may leverage to relate disparate provenance elements.

In contrast, *homogeneous provenance* employs the same provenance data model and annotation schema throughout. This makes such provenance amenable to calculating statistics over it. An adversary may be able to use such priors to make inferences about missing elements in query responses.

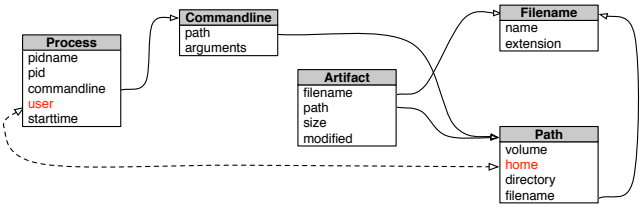

Figure 1: When provenance attributes or relationships are deemed to be privacy-sensitive, ontologies (such as those used for data integration) can be leveraged to infer other elements that must be sanitized as well. Above, the ontology relates the `user` whose `Process` is running to the `home` part of a filesystem `Path` that identifies the user.

3.1 Heterogeneous Provenance

Each provenance metadata set may be captured at a distinct level of abstraction, using overlapping or distinct sets of identifiers to refer to the same artifacts, processes, and agents. It is also possible the same annotations from different data sets may have dissimilar semantics. To facilitate analysis, schema that specify the relationships between provenance annotation components are assumed to be available. For example, the name of the user that a process executes as is closely related to a part of the path of files in that user's home directory (on many systems). This is depicted in Figure 1.

In the case that provenance from distinct data sets are combined, knowledge of the semantic relationships (that an adversary could use to breach privacy) can be used to relate components from different graphs. This can be used to compute the transitive closure of sensitive elements in the combined provenance graph, starting from a seed set of provenance attributes whose privacy is to be maintained. All elements in the computed set can then be sanitized, thereby preventing the privacy-violating inferences. Continuing with the example above, if either the `user` annotation in a `Process` vertex [24] or the `home` annotation in a `Path` vertex need to be sanitized, so should the other.

3.2 Homogeneous Provenance

When users collect and analyze provenance metadata at the operating system level, a single workload that runs in a few hours can generate tens of thousands of vertices in a provenance database. Some workflows may run for weeks, generating commensurately larger provenance graphs. This can give rise to a significant amount of structure in the provenance that derives from the repetitive nature of many tasks carried out, such as trying the same task with a range of input values to see which yields the best output.

Sanitizing the annotations of particular vertices in the provenance graph may not suffice in the above case because of topological relationships. Consider an example, where the name of each patient is used as the filename of the document in which the patient's confidential responses are recorded. To maintain patient privacy, these files are renamed before they are shared. All *Artifact* vertices [24] with annotations that match any of the patients' names are sanitized. However, as Figure 2 illustrates, file renaming operations result in a specific pattern in the provenance graph. (In Figure 2, the shape of the vertex indicates its Open Provenance Model (OPM) [24] type – oval for *Artifact* and rectangle for *Process* – while the color of the edge indicates the OPM relationship – green for *used*, red for *wasGeneratedBy*, and orange for *wasDerivedFrom*.) This structure can be used to determine the intermediate version of the file and obtain aspects such as the time-stamp and size of the file. Correlating these attributes with the arrival times and attachment sizes in email logs, for example, would allow the patient responses to be deanonymized.

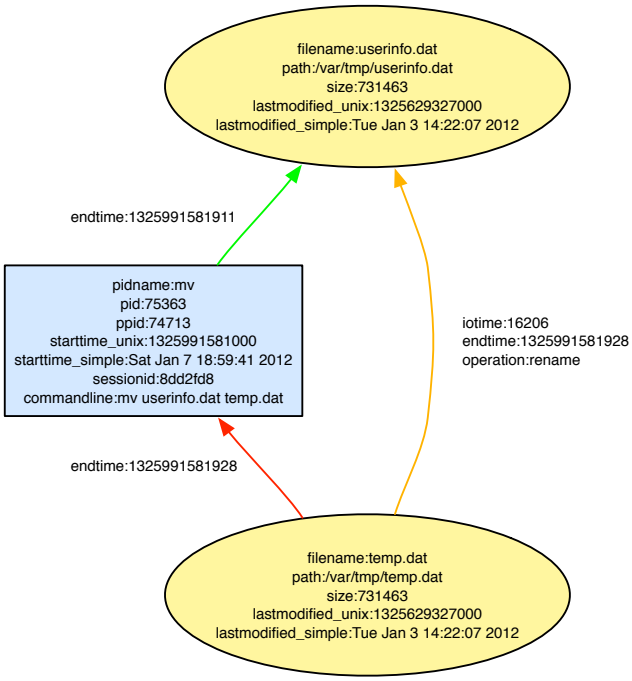

Figure 2: Structure in a provenance graph may leak information about vertices in non-obvious ways. In this simple example, annotations sanitized from the *Artifact* vertex of the file *userinfo.dat* can be inferred from those of the file *temp.dat* since these vertices refer to the same file before and after renaming.

3.3 Sanitization Residue

Provenance elements may need to be accessible to authorized parties in order to answer particular queries. However, the same elements may be sensitive enough that they needed to be protected from others. For example, electronic medical records may be augmented with Fast Healthcare Interoperability Resource (FHIR) [8] provenance metadata (in W3C PROV [27] format). A doctor may need to grant different subsets of these records to clinical trial administrators and patients.

This may be further complicated by the fact that subsets of a single data artifact's history may have been derived from multiple owners (since independent portions of the data may have originated from distinct individuals' information and activities). Each person may wish to apply a different access policy to their part of the provenance records.

At the time that provenance metadata is being created, the data's owner may not know who will need access to the metadata. In some settings, it may be desirable that the data owner does not know who will access the provenance, as is the case for a data set that is being submitted along with a paper to a journal for review. Instead, access may need to be granted to more abstract entities, such as individuals serving as referees or auditors.

In such situations, a subset of the sensitive provenance elements cannot be sanitized. Instead, a flexible cryptographic primitive, *attribute-based encryption* [2], can be utilized. The advantage of scoping access with encryption is that the metadata can be shared across trust boundaries, enabling its distribution through untrusted intermediaries such as brokers in publish-subscribe systems or cloud-based data storage services.

4 Protection Primitives

SPADE is an open source provenance middleware framework [12]. It supports inferring, collecting, filtering, storing, and querying records of provenance in multiple data models, including OPM [24], W3C PROV [27], and DARPA's CDM [21]. An instance of SPADE consists of a *Kernel* daemon to which components of various types can be added. This process, its children, and the operating system kernel and hardware that they run on are assumed to be trusted [11].

SPADE's default *Analyzer* component provides access to a rich query surface [15]. It can be used to locate specific vertices and edges, find paths between endpoints, compute ancestor or descendant lineage, and perform set operations on provenance subgraphs. Intermediate results are stored in variables without materializing the underlying subgraphs. This facilitates efficient faceted search on "big provenance". When a subgraph is materialized for export, it can be programmatically transformed. The latter functionality allows query responses to be transparently sanitized or encrypted as described below.

4.1 Response Rewriting

The system has multiple points of extensibility, including *transformers* [14] that allow responses to queries to be rewritten dynamically. An incoming query is sent to the selected storage, which computes and returns a response provenance graph. When a transformer is added, it operates on this graph and programmatically modifies it. This is illustrated in Figure 3.

Two transformers have been developed. Details on their usage are provided in SPADE's Wiki [26].

4.1.1 Sanitization Transformer

The first module implements schema-specific *provenance sanitization*. This is of particular utility in settings where there are no established trust relationships between the users making the queries and the service managing the provenance records. The absence of trust precludes users obtaining access credentials from an administrator.

The changes made to the provenance response are irreversible since the original annotation values are replaced with

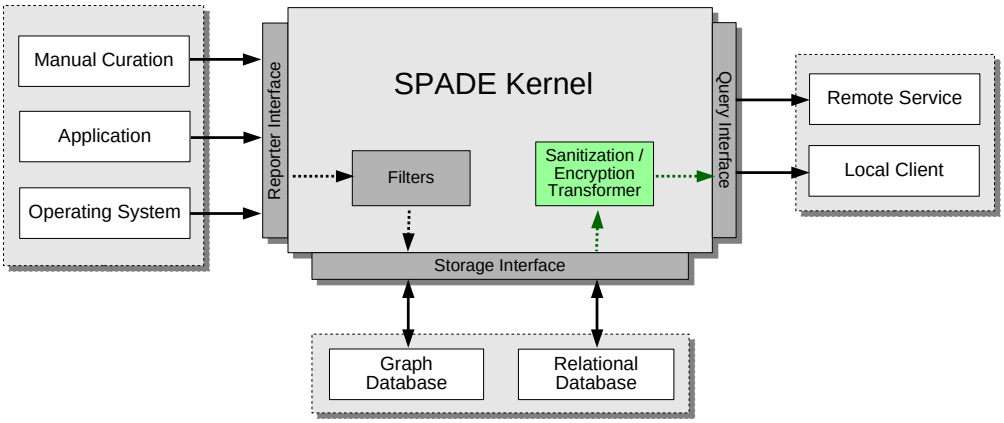

Figure 3: SPADE *transformers* rewrite the response graph returned after a provenance query. The two depicted here sanitize and encrypt the values of configured keys in the annotations on vertices and edges of outbound provenance graphs.

the sanitized variants before the graph is returned to the querying client. This form of provenance privacy protection is vulnerable to re-identification attacks. However, it may still be useful in situations where graphs (rather than aggregate statistics) need to be sent in response to queries.

As an example, the HIPAA[1] "safe harbor" approach for patient record de-identification stipulates removal or generalization of 18 elements [29]. The `Sanitization` Transformer can be used for this by editing its configuration to specify the annotation keys that are covered.

4.1.2 Encryption Transformer

In situations where the querying clients have prior trust relationships with the provenance service, a system administrator can arrange for each user to be provided with cryptographic credentials. Once the provenance service can assume that users have appropriate decryption keys, the provenance subgraphs materialized in response to queries can be appropriately encrypted prior to transmission to clients.

The second module provides support for fine-grained attribute-based *provenance encryption* using OpenABE [25]. When the `ABE` Transformer is applied, it produces an encrypted graph. Its configuration can be edited to specify which annotations need to be encrypted. This is a reversible transformation since the encrypted values can be decrypted if suitable attribute private keys are available at the client. Support has been added to SPADE's query engine to recognize graphs returned in this format and transparently handle decryption.

Both transformers can be customized. In particular, the set of annotations that they operate upon can be configured to match the schema of the provenance records in use. The default configurations contain support for operating upon provenance inferred from Linux Audit system call events using SPADE's corresponding Reporter module. The functionality of both transformers can be extended with *handlers* that provide custom treatment for specific annotations. This allows each part of a complex data type, such as a filesystem path, network address, or timestamp, to be sanitized or encrypted differently depending on the level of protection being applied.

To simplify use of the transformers, syntactic sugar is provided for selecting the annotations that will be modified. More specifically, the set of annotations that will be operated upon can be partitioned into three categories, *low*, *medium*, and *high*, in the configuration for each transformer. When the transformer is added to a running SPADE Kernel, a corresponding *level* argument can be provided to indicate which annotations will be sanitized. Use of this functionality is optional (since all annotations can be moved to a single level, if that is preferable).

4.2 Noisy Statistics

In some settings, external sources of information may be available about users whose activity is captured in the provenance records. Such auxiliary data may be leveraged for re-identification despite the protection provided by the `Sanitization` Transformer. Similarly, patterns in provenance graphs may be exploited to violate privacy, as illustrated in Figure 4.

At the same time, there may be no established trust relationships between the querying clients and the provenance service, precluding use of the `ABE` Transformer. Though of more limited utility, aggregate statistics computed over the provenance can be returned instead. Such results can be robustly protected using *differential privacy* [7].

SPADE's QuickGrail query surface [15] stores the response subgraph of a provenance query in a variable. This allows it to be reused in subsequent queries, manipulated by set

[1] United States Health Insurance Portability and Accountability Act

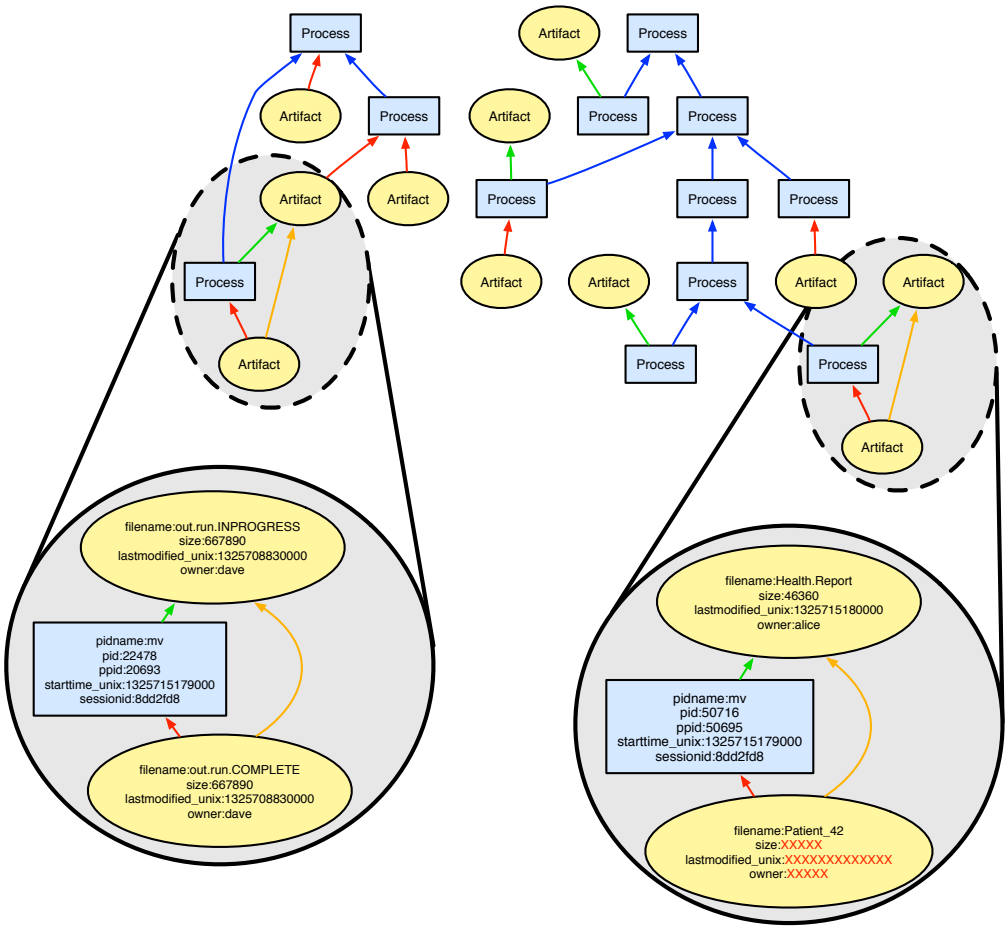

Figure 4: Topological patterns (such as isomorphic subgraphs with highly correlated annotations) can be exploited by an adversary to recover sanitized values. In this example, noting the structure of the *rename* operation (also described in Figure 2) on the left allows the reconstruction of the sanitized annotations (marked with red X's) as their values are unsanitized in the source of the rename operation on the right. In particular, this would reveal the patient's identity.

operations on "big provenance" (without incurring the cost of materializing the content), or exported for visualization or external processing. The query language contains a `stat` command, initially limited to reporting the number of vertices and edges in a graph variable.

Provenance is represented as a property graph in SPADE, with each vertex and edge associated with a set of annotations. Each annotation has a key and a value. Support is being added to allow a *stat* query to specify whether it should operate on the vertices or edges of a graph variable, and on which specific annotation key. Given this, the query can ask for the *mean*, *standard deviation*, *histogram*, or *distribution* of associated values. The *distribution* variant takes a bin count, divides the range of values from the minimum to the maximum into the specified number of sub-range bins, and then reports the number of values in each bin.

Differential Privacy in the Analyzer

A SPADE *Analyzer* provides an interface to the user for retrieving stored provenance records. It is responsible for receiving a query from a client, sending it to the appropriate storage, processing the information returned, and sharing the result with the user.

The default Analyzer is being extended with support for *differential privacy* [7]. In particular, if the underlying storage returns a response that is an aggregate statistic of the form described above, the value can be perturbed to preserve privacy. The level of noise added (determined by the differential privacy ε value used) can be specified in the Analyzer's configuration. This allows a tradeoff to be made between the utility of the response and the privacy of parties whose activity has been captured in the database of provenance records.

The implementation utilizes Google's differential privacy libraries [16].

5 Conclusion

We have described concerns with protecting the privacy of provenance in practice. Support has been added to SPADE for rewriting responses to provenance queries to either sanitize the subgraph by eliding elements or use attribute-based encryption to protect specific elements. An ongoing effort adds storage functionality for computing aggregate statistics about provenance subgraphs along with Analyzer support for adding differential privacy noise to the results.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant ACI-1547467. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Nabil Adam and John Worthmann, **Security-control methods for statistical databases: A comparative study**, *ACM Computing Surveys*, Vol. 21(4), 1989.

[2] John Bethencourt, Amit Sahai, and Brent Waters, **Ciphertext-Policy Attribute-Based Encryption**, IEEE Symposium on Security and Privacy, 2007.

[3] Uri Braun, Avraham Shinnar, and Margo Seltzer, **Securing provenance**, *3rd USENIX Conference on Hot Topics in Security*, 2008.

[4] Bee-Chung Chen, Daniel Kifer, Kristen LeFevre, and Ashwin Machanavajjhala, **Privacy-preserving data publishing**, *Foundations and Trends in Databases*, Vol. 2, 2009.

[5] Roxana Danger, Vasa Curcin, Paolo Missier, and Jeremy Bryans, **Access control and view generation for provenance graphs**, *Future Generation Computer Systems*, Vol. 49, 2015.

[6] Grit Denker, Ashish Gehani, Minyoung Kim, and David Hanz, **Policy-Based Data Downgrading: Toward A Semantic Framework and Automated Tools to Balance Need-To-Protect and Need-To-Share Policies**, *11th IEEE International Symposium on Policies for Distributed Systems and Networks*, 2010.

[7] Cynthia Dwork, **Differential privacy: A survey of results**, *Theory and Applications of Models of Computation, Lecture Notes in Computer Science, Springer-Verlag*, Vol. 4978, 2008.

[8] HL7 FHIR, https://www.hl7.org/fhir/provenance.html

[9] Benjamin Fung, Ke Wang, Rui Chen, and Philip Yu, **Privacy-preserving data publishing: A survey on recent developments**, *ACM Computing Surveys*, Vol. 42(4), 2010.

[10] Ashish Gehani, Dawood Tariq, Basim Baig, and Tanu Malik, **Policy-based integration of provenance metadata**, *12th IEEE International Symposium on Policies for Distributed Systems and Networks*, 2011.

[11] Ashish Gehani, David Hanz, John Rushby, Grit Denker, and Rance DeLong, **On the (F)utility of Untrusted Data Sanitization**, *30th IEEE Military Communications Conference*, 2011.

[12] Ashish Gehani and Dawood Tariq, **SPADE: Support for provenance auditing in distributed environments**, *13th ACM/IFIP/USENIX International Conference on Middleware*, 2012.

[13] Ashish Gehani and Dawood Tariq, **Provenance-Only Integration**, *6th USENIX Workshop on the Theory and Practice of Provenance*, 2014.

[14] Ashish Gehani, Hasanat Kazmi, and Hassaan Irshad, **Scaling SPADE to "Big Provenance"**, *8th USENIX Workshop on the Theory and Practice of Provenance*, 2016.

[15] Ashish Gehani, Raza Ahmad, Hassaan Irshad, Jianqiao Zhu, and Jignesh Patel, **Digging Into "Big Provenance" With SPADE**, *ACM Queue*, Vol. 19(3), 2021.

[16] Google Differential Privacy, https://github.com/google/differential-privacy

[17] Ragib Hasan, Radu Sion, and Marianne Winslett, **Introducing secure provenance: Problems and challenges**, *ACM Workshop on Storage Security and Survivability*, 2007.

[18] Thomas Heinis and Gustavo Alonso, **Efficient lineage tracking for scientific workflows**, *ACM SIGMOD International Conference on Management of Data*, 2008.

[19] David Holland, Uri Braun, Diana Maclean, Kiran-Kumar Muniswamy-Reddy, and Margo Seltzer, **Choosing a data model and query language for provenance**, *2nd International Provenance and Annotation Workshop*, 2008.

[20] Anastasios Kementsietsidis and Min Wang, **On the efficiency of provenance queries**, *25th International Conference on Data Engineering*, 2009.

[21] Joud Khoury, Timothy Upthegrove, Armando Caro, Brett Benyo, and Derrick Kong, **An event-based data model for granular information flow tracking**, *12th USENIX Workshop on the Theory and Practice of Provenance*, 2020.

[22] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam, **l-diversity: Privacy beyond k-anonymity**, *ACM Transactions on Knowledge Discovery from Data*, Vol. 1(1), 2007.

[23] Paolo Missier, Jeremy Bryans, Carl Gamble, Vasa Curcin, **Abstracting PROV provenance graphs: A validity-preserving approach**, *Future Generation Computer Systems*, Vol. 111, 2020.

[24] Luc Moreau, Ben Clifford, Juliana Freire, Yolanda Gil, Paul Groth, Joe Futrelle, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche, **The Open Provenance Model core specification (v1.1)**, *Future Generation Computer Systems*, 2010.

[25] OpenABE, https://github.com/zeutro/openabe

[26] SPADE Provenance Privacy, https://github.com/ashish-gehani/SPADE/wiki/Provenance-Privacy

[27] W3C PROV, http://www.w3.org/TR/prov-overview/

[28] Arnon Rosenthal, Len Seligman, Adriane Chapman, and Barbara Blaustein, **Scalable access controls for lineage**, *1st USENIX Workshop on the Theory and Practice of Provenance*, 2009.

[29] Safe Harbor, https://en.wikipedia.org/wiki/De-identification#Safe_harbor

[30] Latanya Sweeney, **k-anonymity: A model for protecting privacy**, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 10(5), 2002.

[31] Victor Tan, Paul Groth, Simon Miles, Sheng Jiang, Steve Munroe, Sofia Tsasakou, and Luc Moreau, **Security issues in a SOA-based provenance system**, *Provenance and Annotation of Data, Lecture Notes in Computer Science, Springer*, Vol. 4145, 2006.

[32] Zhiqiang Yang, Sheng Zhong, and Rebecca Wright, **Anonymity-preserving data collection**, *11th ACM International Conference on Knowledge Discovery in Data Mining*, 2005.