PITA: Privacy Through Provenance Abstraction

Daniel Deutch
Tel Aviv University
danielde@tauex.tau.ac.il

Ariel Frankenthal
Tel Aviv University
frankenthal@mail.tau.ac.il

Amir Gilad
Duke University
agilad@cs.duke.edu

Yuval Moskovitch
University of Michigan
yuvalm@umich.edu

2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00311

***Abstract*—Provenance is a valuable tool for explaining and validating query results. On the other hand, provenance also reveals much of the details about the query that generated it, which may include proprietary logic that the query owner does not wish to disclose. To this end, we propose to demonstrate PITA, a system designed to allow the release of provenance information, while hiding the properties of the underlying query. We formalize the trade-off between the level of information encoded in a provenance expression and the breach of privacy it incurs. Following this model, we design PITA to abstract the provenance so that it incurs minimum loss of information, while keeping privacy above a given threshold, namely protecting details of the original query from being revealed.**

***Index Terms*—provenance, privacy, explanations, k-anonymity**

I. Introduction

Data provenance is commonly used for explaining and justifying query results [1], [2]. Organizations, as data owners, can therefore use provenance to explain some of their query-based decisions by showing (a user friendly form of) provenance to users. However, publishing the provenance of specific query results may reveal the properties of the proprietary query that generated them [3]. This, in turn, may discourage data owners from providing provenance-based explanations.

Example 1.1: Consider an online advertising company that wishes to match ads to people. Their database contains information about people, their hobbies and interests, a sample of which appears in Figure 1. Each tuple has an identifier, appearing to its left. The company may run queries such as Q_{real} appearing in Table I looking for people that like dancing and music. The query output includes James and Brenda, and relevant advertisements may then be presented to them. Upon request, Brenda may receive an explanation of why the advertisement was shown to her (see, e.g., [4], [5]). However, the company may wish to avoid disclosing the general criteria (i.e., the query Q_{real}), since these criteria are part of the company's confidential business strategy.

TABLE I: Queries for the running example. Q_{real} is the original, Q_{false1} and Q_{false2} are similar but not identical

Name	Query
Q_{real}	Q(id) :- Person(id,name,age), Hobbies(id,'Dance',src1), Interests(id,'Music',src2)
Q_{false1}	Q(id) :- Person(id,name,age), Hobbies(id,'Trips',src1), Interests(id,'Music',src2)
Q_{false2}	Q(id) :- Person(id,name,age), Hobbies(id,'Dance',src1), Interests(id,'Parties',src2)

Interests

	PID	Interest	Source
i_1	1	Music	WikiLeaks
i_2	2	Music	Facebook
i_3	3	Music	LinkedIn
i_4	1	Parties	WikiLeaks
i_5	2	Parties	Facebook
i_6	4	Movies	WikiLeaks

Hobbies

	PID	Hobby	Source
h_1	1	Dance	Facebook
h_2	2	Dance	LinkedIn
h_3	4	Dance	Facebook
h_4	1	Trips	Facebook
h_5	2	Trips	LinkedIn
h_6	3	Trips	WikiLeaks

Persons

	PID	Name	Age
p_1	1	James T	27
p_2	2	Brenda P	31

Fig. 1: Partial database instance of hobbies and interests of people collected from different sources

Output	Provenance
1	$p_1 \cdot h_1 \cdot i_1$
2	$p_2 \cdot h_2 \cdot i_2$

(a) Ex_{real}

Output	Provenance
1	$p_1 \cdot h_4 \cdot i_1$
2	$p_2 \cdot h_5 \cdot i_2$

(b) Ex_{false1}

Output	Provenance
1	$p_1 \cdot h_1 \cdot i_4$
2	$p_2 \cdot h_2 \cdot i_5$

(c) Ex_{false2}

Fig. 2: $\mathbb{N}[X]$-examples. Ex_{real}, Ex_{false1} and Ex_{false2} are the outputs of Q_{real}, Q_{false1} and Q_{false2}, respectively

We propose to demonstrate PITA*, a novel system for obfuscating the provenance of query results so that it remains useful and informative, while hiding the underlying query.*

The provenance of a given query result describes the tuples used by the query to derive the result and the manner in which they were used. We use here the well-established model of *provenance semirings* [6], and specifically that of provenance polynomials ($\mathbb{N}[X]$, using the notation of [6]).

Example 1.2: The provenance of the output tuple (1) according to Q_{real} (Table I) is presented in the first row of Figure 2a. The expression, formulated as a product of the annotations p_1, h_1, i_1, intuitively means that the three tuples with these annotations in the database (Figure 1) have jointly participated in an assignment to Q_{real} that yielded this result.

Provenance obfuscation is done by abstracting it. Namely, abstracting the tuple annotations in it. We do so by replacing some tuple annotations in the provenance expression with meta-annotations. To ensure that the replacement is logical, we rely on the concept of an *abstraction tree* - a tree whose leaves correspond to actual tuple annotations and ancestors can be used as abstractions of their descendants. This idea is adapted from [7] where trees were used to compress provenance.

When the original annotations in the provenance are abstracted, the specifics of the tuples that they represent are lost. We quantify this loss using entropy [8]. Information entropy expresses the level of uncertainty of the given data. In our context, we wish to measure "how uncertain" is a

978-1-7281-9184-3/21/$31.00 ©2021 IEEE DOI
10.1109/ICDE51399.2021.00311

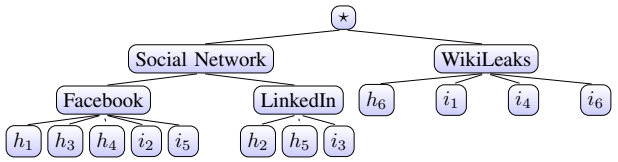


Fig. 3: Abstraction tree containing a subset of the tuple annotations in the database in Figure 1 as leaves, and inner nodes that are abstractions of the leaves

viewer of the abstracted provenance expression, with respect to the actual one (each possibility for the actual provenance, given an abstraction, is called a concretization). We assume a given distribution over the concretizations. Lacking additional knowledge, this distribution may simply be taken as uniform. The entropy for an abstraction is then defined with respect to a tree and a distribution.

The privacy given by a certain abstraction of the provenance is then measured as the number of 'likely' queries that can yield it when generating the output. Specifically, we define a 'likely' query as a connected inclusion-minimal (CIM) query [3], i.e., queries whose join graph is connected and are not included in any other query in this set. This idea draws on the well founded concept of k-anonymity [9], originally developed for data rather than queries.

Given a subset of the query results and their provenance, an abstraction tree, and a privacy threshold k, PITA finds an optimal abstraction, i.e., an abstraction that has at least k CIM queries that 'can fit' it, and minimizes the loss of information among all such abstractions.

Example 1.3: Consider Ex_{real} presented in Figure 2a showing two outputs of the query Q_{real} and their provenance. The allowed abstractions are defined based on the tree T depicted in Figure 3. The leaves of T are annotations (identifiers) of the tuples in Figure 1, and its inner nodes are abstracted forms of these annotations. An abstraction of the provenance in Ex_{real} w.r.t. T may, e.g., replace the annotation h_1 with its ancestors *Facebook* or *Social Network*. Other tuple annotations may be abstracted as well. A choice of abstraction dictates a certain amount of information loss since the annotation *Facebook* can stand for any one of the annotations h_1, h_3, h_4, i_2, i_5, and when viewing the annotation *Facebook* we cannot be sure which annotation is the original. At the same time, it may obfuscate the underlying query Q_{real}, as more queries can possibly fit the observable provenance information.

Finding an optimal abstraction is NP-hard. Hence, we provide novel heuristic algorithms for computing optimal abstractions in practically efficient ways.

Query owners can therefore upload to PITA a set of results along with their provenance, an abstraction tree, and a privacy threshold. PITA will then compute an optimal abstraction of the provenance according to the tree and output this abstraction. This allows query owners to publish a subset of the results along with their (abstracted) provenance as explanations, while guaranteeing the privacy of the query itself. We will demonstrate PITA using real-world data from IMDB, showing its usefulness and effectiveness.

To the best of our knowledge, PITA is the first system to demonstrate the approach of obfuscating fine-grained provenance through abstraction, guaranteeing a privacy threshold for queries.

II. Technical Details

We (informally) introduce the model underlying PITA, through a running example.

Queries and Provenance: We assume that the reader is familiar with Conjunctive Queries, and refer to [6], [3] for formal definitions of provenance, and $\mathbb{N}[X]$-examples. We denote by $\mathbb{N}[X]$-*example* ("example") a row composed of a query result and its provenance.

Example 2.1: An $\mathbb{N}[X]$-example is depicted in Figure 2a where the left column shows two output examples, and the right column shows the provenance of each, respectively.

Obfuscating Provenance through Abstraction: We propose a simple way to obfuscate provenance, based on a *provenance abstraction tree*. The general idea is to replace tuple annotations with "meta-annotations", e.g., in Figure 3, the tuples h_2, h_5, i_3 can be abstracted to $LinkedIn$.

An *abstraction function* maps the annotations found in the provenance of an $\mathbb{N}[X]$-example to their abstraction according to the structure of the abstraction tree. Essentially, an abstraction function converts the explicit provenance in the $\mathbb{N}[X]$-example into an implicit one by replacing some of the original tuple annotations with their ancestors.

Example 2.2: Reconsider the $\mathbb{N}[X]$-example Ex_{real} in Figure 2a and the abstraction function A^1_T depicted in Figure 4. Using A^1_T on Ex_{real} will create the abstracted $\mathbb{N}[X]$-example $\widetilde{Ex}_{abs1}$ shown in Figure 5. Formally, $A^1_T(Ex_{real}) = \widetilde{Ex}_{abs1}$.

$$A^1_T(v) = \begin{cases} Facebook, & \text{if } v = h_1, h_4 \\ LinkedIn, & \text{if } v = h_2, h_5 \\ v, & \text{otherwise} \end{cases}$$

Fig. 4: Abstraction Functions

$\widetilde{Ex}_{abs1} = A^1_T(Ex_{real}) =$
$A^1_T(Ex_{false1})=$

Output	Provenance
1	$p_1 \cdot Facebook \cdot i_1$
2	$p_2 \cdot LinkedIn \cdot i_2$

$\widetilde{Ex}_{abs2} = A^2_T(Ex_{real}) =$
$A^2_T(Ex_{false2})=$

Output	Provenance
1	$p_1 \cdot h_1 \cdot WikiLeaks$
2	$p_2 \cdot h_2 \cdot Facebook$

Fig. 5: Abstracted $\mathbb{N}[X]$-examples

$$A^2_T(v) = \begin{cases} WikiLeaks, & \text{if } v = i_1, i_4 \\ Facebook, & \text{if } v = i_2, i_5 \\ v, & \text{otherwise} \end{cases}$$

A concretization is the 'reverse' operation of abstraction. The concretization set of $\widetilde{Ex}$ is defined by $C(\widetilde{Ex}) = \{Ex \mid \exists A_T.\ A_T(Ex) = \widetilde{Ex}\}$.

Example 2.3: Reconsider the abstracted $\mathbb{N}[X]$-example $\widetilde{Ex}_{abs1}$ presented in Figure 5, the $\mathbb{N}[X]$-example Ex_{real} shown in Figure 2a and the abstraction function A^1_T given in Figure 4. From Example 2.2, we have $Ex_{real} \in C(\widetilde{Ex}_{abs1})$ since $A^1_T(Ex_{real}) = \widetilde{Ex}_{abs1}$. Now consider the $\mathbb{N}[X]$-example Ex_{false1} shown in Figure 2b. It also holds that $A^1_T(Ex_{false1}) = \widetilde{Ex}_{abs1}$, and thus $Ex_{false1} \in C(\widetilde{Ex}_{abs1})$, i.e., Ex_{false1} is also in the concretization set of $\widetilde{Ex}_{abs1}$.

Quantifying Loss of Information: Each abstraction entails a loss of information. We measure the loss of information of an abstracted $\mathbb{N}[X]$-example $\widetilde{Ex}$ via the notion of *Entropy*. Entropy is the average level of "information" or "uncertainty" inherent in the possible outcomes of a random variable [8]. Given a random variable X, with possible outcomes x_i, each with probability $P_X(x_i)$, the entropy $H(X)$ of X is as follows: $H(X) = -\sum_i P_X(x_i) \ln P_X(x_i)$. The entropy quantifies how 'informative' or 'surprising' the random variable is, averaged over all of its possible outcomes. The loss of information of $A_T(Ex)$ is then defined by the entropy on the concretization set: $-\sum_{i=1}^{n} P_X(x_i) \ln P_X(x_i)$ where $X = C(A_T(Ex)) = \{x_1, \ldots, x_n\}$ and $P_X(x_i)$ is the probability of the concretization x_i. The probabilities may be determined using statistical properties of the database or external information.

Example 2.4: Reconsider the abstracted $\mathbb{N}[X]$-example Ex_{real}, the abstracted tree T and the abstraction function A_T^1 (shown in Figures 2a, 3 and 4 resp.). The output of $A_T^1(Ex_{real})$ is the abstracted $\mathbb{N}[X]$-example $\widetilde{Ex}_{abs1}$ shown in Figure 5. Let $c_1 \ldots, c_{15}$ be the concretizations of $\widetilde{Ex}_{abs1}$. Assuming the probabilities of the concretizations are $P_X(c_i) = 0.05$ if $i \in (1, ..., 10)$ and $P_X(c_i) = 0.1$ if $i \in (11, ..., 15)$. The loss of information of $\widetilde{Ex}_{abs1}$ with those probabilities is then $-\sum_{i=1}^{15} P_X(c_i) \ln P_X(c_i) \approx 2.649$.

Note that for a finite probability space X with a discrete uniform distribution over n states, the entropy is $H(X) = \ln(n)$. Since $C(A_T(Ex))$ is a finite set, if the probabilities of all concretizations in $C(A_T(Ex))$ are equal then the loss of information of $A_T(Ex)$ is $\ln(|C(A_T(Ex))|)$.

Provenance Privacy: Recall that our goal is to show an abstraction of a given $\mathbb{N}[X]$-example, while hiding the query that yielded the $\mathbb{N}[X]$-example. To measure the privacy of an abstraction, we can consider the set of its possible concretizations. For each concretization, we look at the set of queries that, when given the provenance of each row in $\mathbb{N}[X]$-example would generate the corresponding output tuple (previously referred to as *consistent* [3]). For example, Q_{real} (Table I) is consistent w.r.t. Ex_{real} (Figure 2a) since when given the tuple annotated by p_1, h_1, i_1, it will output (1) and this is also the case for the second row. However, not all such queries are "interesting". Therefore, we may restrict attention to CIM (connected inclusion-minimal) queries., i.e., queries whose join graph is connected and are not contained in any other query in this set. These queries are representative of the viable options for the hidden query. We define the privacy incurred by an abstraction as the cardinality of this set (i.e., the number of CIM queries that match some concretization).

Example 2.5: Reconsider the abstracted $\mathbb{N}[X]$-example $\widetilde{Ex}_{abs1}$ shown in Figure 5. There are only 2 CIM queries, Q_{real} and Q_{false1}, shown in Table I. Both are (1) consistent, (2) connected, and (3) are not subsumed by other consistent queries (the rest of the consistent queries are either disconnected or are not inclusion-minimal since they contain less joins or fewer constants). Thus, the privacy of $\widetilde{Ex}_{abs1}$ is 2.

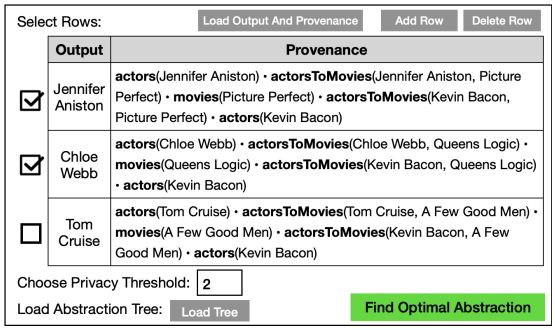
The Problem of Optimizing Abstractions: These two components (privacy and loss of information) are then combined to define the problem of optimal provenance abstraction: given an $\mathbb{N}[X]$-example and a privacy threshold, we want to find an abstraction that satisfies this threshold but also minimizes the loss of information. We call this abstraction an *optimal abstraction*.

Example 2.6: Reconsider the database depicted in Figure 1, the query Q_{real} shown in Table I, its output Ex_{real} given in Figure 2a and the abstraction tree T presented in Figure 3. Assume that the privacy threshold is 2 (i.e., we want our privacy to be at least 2) and the loss of information is entropy with discrete uniform distribution, therefore the loss of information is $\ln|C(\widetilde{Ex})|$ where $C(\widetilde{Ex})$ is the set of all concretization of $\widetilde{Ex}$. We can use the abstraction function A_T^2 (detailed in Figure 4) so that $A_T^2(Ex_{real})$ yields $\widetilde{Ex}_{abs2}$ (depicted in Figure 5). Since the queries Q_{real} and Q_{false2} (shown in Table I) are CIM w.r.t. $\widetilde{Ex}_{abs2}$, its privacy is 2. In addition, $\ln|C(\widetilde{Ex}_{abs2})| = \ln 20 \approx 2.996$, thus the loss of information incurred by $A_T^2(Ex_{real})$ is 2.996. On the other hand, we can use the abstraction function A_T^1 (detailed in Figure 4) so that $A_T^1(Ex_{real})$ yields $\widetilde{Ex}_{abs1}$ (depicted in Figure 5). In Example 2.5 we have seen that the privacy of $\widetilde{Ex}_{abs1}$ is 2. In addition, $\ln|C(\widetilde{Ex}_{abs1})| = \ln 15 \approx 2.708$, thus the loss of information incurred by $A_T^1(Ex_{real})$ is 2.708. Since the loss of information of A_T^1 is smaller than all possible abstraction functions that guarantee privacy ≥ 2 (in particular, A_T^2), it is an optimal abstraction.

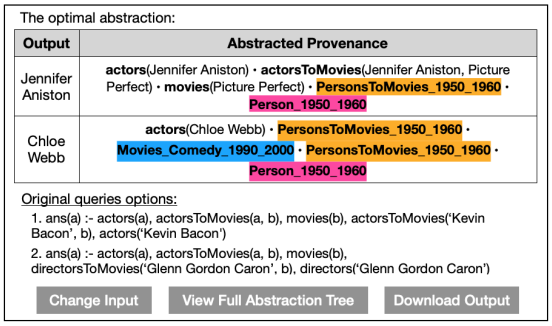
Solution: The problem of finding an optimal abstraction is NP-hard. Bearing this bound in mind, we provide novel heuristic algorithms for computing optimal abstractions in practically efficient ways. Our approach revolves around several key ideas. First, we optimize the order of traversal over the possible abstractions, by examining "simpler" abstractions first. We further prioritize the computation of loss of information over privacy, as the former can be done significantly more efficiently. Additionally, privacy computation is performed in a greedy fashion, relying on the properties of the $\mathbb{N}[X]$-example. Namely, we check the connectedness and consistency of candidate queries in a row-by-row fashion. Finally, caching is used in order to avoid repetitive computations. Our heuristics and optimizations render our approach scalable even for large databases and complex queries.

III. System Overview

`PITA`'s back-end side is implemented in Java 13. Its web-based GUI was built using JavaScript, CSS and HTML. The general architecture of `PITA` is shown in Figure 7. In order to provide explanations for a query without revealing it, users may employ any provenance-aware query engine (such as `SelP` [10]) to get the query results alongside their provenance. These $\mathbb{N}[X]$-example, together with an abstraction tree (provided as a JSON file) and a privacy threshold are then fed to `PITA`. The system contains three components. The main algorithm is implemented in the *Abstraction Factory*, which iterates over all possible abstractions. For each abstraction, it utilizes the *Privacy Computation Engine* and the *LOI Computation Engine* to evaluate the privacy and the



(a) Input Screen



(b) Output Screen

Fig. 6: PITA User Interface

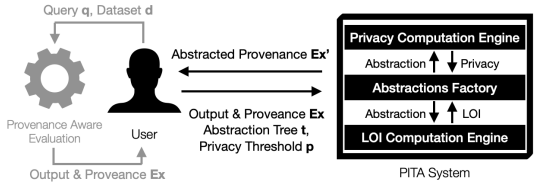


Fig. 7: PITA Architecture

loss of information for each abstraction. Finally, the optimal abstraction is returned to the user. The input and output screens of the system are shown in Figure 6.

Input Screen (Figure 6a): PITA presents the given $\mathbb{N}[X]$-example. The user can insert rows manually (using 'Add Row' and 'Delete Row' buttons), or load the $\mathbb{N}[X]$-example (generated using a provenance-aware engine) from a JSON file (using 'Load Output And Provenance' button). Using a check-mark, the user may choose a subset of rows. The user can then load the abstraction tree JSON file (using 'Load Tree' button) and input the desired privacy threshold (in the 'Choose Privacy Threshold' field). Clicking on the 'Find Optimal Abstraction' button will invoke PITA to compute the optimal abstraction that satisfies the privacy threshold for the chosen rows, using the abstraction tree.

Output Screen (Figure 6b): Once the optimal abstraction is computed, PITA presents the computed abstraction of the chosen rows, with the abstracted tuples highlighted in different colors. Hovering over an abstracted tuple shows all options for replacing this abstracted tuple with a concrete tuple (based on the abstraction tree). In addition, the system list all possible CIM queries, i.e., all queries that might be the original one, so users can validate that these are reasonable queries and that their original query is properly obfuscated.

IV. Demonstration Scenario

We will first walk the audience through the process of obfuscating provenance using real-world data from the IMDB dataset [11]. We will start by presenting the underlying database to the audience. The participants will then be asked to select a query from a set of pre-defined queries or formulate one on their own. In this demonstration, we will use the SelP system [10] to generate the $\mathbb{N}[X]$-example. We will invite participants to use PITA allowing them to change the privacy threshold, the selected rows and the abstraction tree. For the demo purpose, we will have several abstraction trees at hand allowing users to experiment with different tree structures.

In the second phase, we will demonstrate how the abstracted $\mathbb{N}[X]$-example provides the desired privacy while still being informative and useful. We will generate an abstracted $\mathbb{N}[X]$-example from an unrevealed query and let them interactively try to reverse-engineer the original query, showing that our system successfully hides the original query. To demonstrate usefulness, we will ask them several hypothetical questions about the data, showing that most of the questions can be answered successfully using only the abstracted $\mathbb{N}[X]$-example (e.g., we could ask "would the output be affected if we were to remove all comedy movies?", referring to Figure 6b).

Finally, we will allow the audience to look "under the hood". In particular, we will show the audience intermediate results of the algorithm (e.g., the privacy and loss of information for representative abstraction) and the computational sequence that lead to the resulting abstraction.

Acknowledgements

This research has been funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 804302) and the Israeli Science Foundation (ISF) Grant No. 978/17.

References

[1] P. Buneman, S. Khanna, and W. Tan, "Why and where: A characterization of data provenance," in *ICDT*, 2001, pp. 316–330.
[2] A. Chapman and H. V. Jagadish, "Why not?" in *SIGMOD*, 2009, pp. 523–534.
[3] D. Deutch and A. Gilad, "Reverse-engineering conjunctive queries from provenance examples," in *EDBT*, 2019, pp. 277–288.
[4] Google, "Why you're seeing an ad," https://support.google.com/accounts/answer/1634057.
[5] Facebook, "Understand why you're seeing certain ads and how you can adjust your ad experience," https://about.fb.com/news/2019/07/understand-why-youre-seeing-ads/.
[6] T. J. Green, G. Karvounarakis, and V. Tannen, "Provenance semirings," in *PODS*, 2007, pp. 31–40.
[7] D. Deutch, Y. Moskovitch, and N. Rinetzky, "Hypothetical reasoning via provenance abstraction," in *SIGMOD*, 2019, pp. 537–554.
[8] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
[9] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, p. 557–570, 2002.
[10] D. Deutch, A. Gilad, and Y. Moskovitch, "Selective provenance for datalog programs using top-k queries," *PVLDB*, vol. 8, no. 12, pp. 1394–1405, 2015.
[11] IMDB, http://www.imdb.com/interfaces.