Addressing the Limitations of Γ-privacy

Isuru Suriarachchi, School of Informatics and Computing, Indiana University

Abstract—**Collection of provenance information is an important aspect of any scientific workflow system. Workflow provenance generally captures lot of information about individual modules in the workflow including input parameters, input and output data products, intermediate data products, module invocation times etc. Therefore, a complete provenance graph contains enough information for someone to have a clear picture about the workflow structure, individual modules and data flow within the workflow. This can cause privacy issues in certain workflows which consume sensitive information. To address these issues, workflow owners may want to keep some provenance information confidential and make sure those are not published with provenance data. Davidson et al. [1] presents Γ-privacy which quantifies the module privacy requirements of scientific workflow provenance data. It ensures the privacy of all modules in the workflow by hiding some information from the original provenance data. And also, Γ-privacy tries to minimize the cost of hidden data to make sure the maximum amount of provenance information is published. However, Cheney and Perera [2] points out some limitations of Γ-privacy including the difficulty of deciding an appropriate value for Γ in a complex workflow. In this paper, we discuss those limitations in more details and present a solution to address the main limitations of Γ-privacy including the difficulty of selecting a value for Γ using the ideas from differential privacy [3] and ProPub [4].**

Keywords—Workflow provenance, Module privacy, Differential privacy, Noise.

I. Introduction

PROVENANCE capture in scientific workflows has been an interesting topic during the last decade. Importance of provenance information is well recognized when it comes to derivation of ownership, assessment of quality and trustworthiness, reproducibility, validation and failure tracing. Most well known workflow engines like Kepler [5] and Taverna [6] capture provenance information automatically. And also, there are tools like Karma [7] and Komadu [8] using which, any workflow system can easily capture and store provenance information. Workflow provenance captures lot of information about individual modules in the workflow including input parameters, input and output data products, intermediate data products, module invocation times etc. Workflow provenance traces are normally represented by directed acyclic graphs in which the nodes are the modules in the workflow and edges are the invocations or data flows among modules. A complete provenance graph contains enough information for someone to have a clear picture about the workflow structure, individual modules and data flow within the workflow.

But there are cases where the owner of the workflow cannot publish the complete provenance graph. In some cases, disclosing certain information in a provenance graph may violate security or privacy. For a simple example, sometimes information like email addresses of human agents in the provenance graph need to be hidden. If some internal module of the workflow takes the social security number from a person's record as a parameter and the set of parameters is been captured as provenance information, those must be hidden from the published provenance trace.

All above scenarios are related to data privacy. But that is not the only reason for not publishing the full provenance trace. In some workflow systems, there can be modules which are proprietary. Even though the complete description about such a module will not be available for an observer, sometimes it is possible to infer its functionality by analysing details about input and output data products and parameter values available in the provenance graph. In some other systems, the workflow structure and invocation sequences may be proprietary. Then again it can be possible to infer those information using the data flow details available in a provenance graph. Davidson et al. [1] [9] presents Γ-privacy which tries to solve the problem of preserving the privacy of workflow modules [10]. The Γ-privacy model quantifies the module privacy and provides a way of preserving it by hiding input and/or output data items (attributes).

In Γ-privacy, a module is treated as a relation which takes a set of input attributes and produces a set of output attributes. A workflow is thought of as a relation which is the input-output join of the individual module relations. Each row in this relation represents a workflow execution. They call this relation the *provenance relation*. A module with functionality m in a workflow is said to be Γ-private if for every input x, the actual value of the output $m(x)$ is indistinguishable from Γ - 1 other possible values with respect to the visible data values in the provenance relation. By extending the this notion, a workflow provenance relation is said to be Γ-private if all the individual modules of the relation are Γ-private. The idea behind this definition is that Γ-privacy ensures that an attacker cannot guess the correct value of $m(x)$ for any module in the workflow for any execution with probability greater than $\frac{1}{\Gamma}$ by looking at a Γ-private provenance relation. Given a complete workflow provenance relation, the way to make it Γ-private is by hiding input-output attributes of internal modules. Sometimes it is possible to achieve the same level of privacy by hiding different subsets of attributes in the provenance relation. However, the amount of useful provenance information conveyed to the user may be different for each subset. Therefore, they assign a cost for each attribute in the relation and try to minimize the cost of hidden attributes. This makes sure that the maximum possible amount of useful information is exposed to the user while achieving Γ-privacy.

Even though Γ-privacy is an interesting way of quantifying the required level of privacy for workflow provenance, it has few limitations as pointed out by Cheney and Perera [2]. The most visible limitation in a practical point of view is that it is not clear how to choose a suitable value for Γ. Selecting a value for Γ can depend on the application and it should be done by analysing different aspects related to the exact use case in hand. If a good value is not chosen for Γ, Γ-privacy may not make much sense. The Γ-privacy model uses a single Γ value for all the modules in a workflow. This is not realistic for a complex workflow which contains different kinds of modules with different privacy requirements. In addition to that, while hiding attributes to preserve Γ-privacy, all attributes are assumed to be equally contributing to the output of the module. In other words, owner of the workflow cannot control the attributes which are selected to be hidden by the system. And also, the polynomial time approximation algorithm proposed in [1] using cardinality constraints assumes that all attributes in a module have the same domain size. That is a very restrictive assumption for a real workflow. Cheney and Perera [2] points out that it might be possible to address some of these by using techniques from quantitative opacity [11], differential privacy [3] and quantitative information flow security [12].

Even though it is hard to figure out a proper value for Γ just by looking at a complex workflow provenance relation, selecting the set of sensitive attributes that must be hidden in the published provenance data is a far more practical way of looking at the same problem. Therefore, we use the idea of allowing the user to select the attributes to be hidden to build our solution. For example, ProPub [4] is a system which uses this idea to preserve security in provenance graphs. In Γ-privacy model, the set of attributes to be hidden is selected by the system for a Γ value provided by the user. In the modified Γ-privacy model that we present, we allow the user to select the attributes that must be hidden and use that input to derive the Γ value for the rest of the workflow. When this model is applied to a workflow which contains modules with different domain sizes, there can be situations where a complete module must be hidden to satisfy the selected Γ value. In such cases, we use the technique of adding noise attributes for those modules without completely hiding the modules in the published provenance data. This idea of adding noise to improve security is used in differential privacy [3].

The organization of this paper is as follows. In section II we describe Γ-privacy using simple examples. In section III we discuss the limitations of Γ-privacy in more details. Then in section IV we evaluate quantitative opacity, differential privacy and information flow security. As the main contribution of this paper, in section V we present a solution to address the above issues in Γ-privacy and discuss the applicability of our solution for more realistic workflows. Finally in section VI we present the conclusion and possible future directions.

II. Γ-PRIVACY

Γ-privacy [1] tries to quantify the level of security required for scientific workflow provenance. A workflow is identified as a set of modules. A module takes a set of attributes as the input and generates another set of attributes as the output. The Γ-privacy model is built by considering the privacy of a single module (*standalone module privacy*) first. Based on that, the privacy of a connected set of modules in a workflow (*workflow module privacy*) is defined. Relational algebra is used to build the Γ-privacy model.

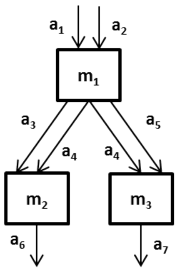

Fig. 1. A simple workflow

TABLE I. *R*: WORKFLOW RELATION

a_1	a_2	a_3	a_4	a_5	a_6	a_7
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	1	0	0	1
1	1	1	0	1	1	1

TABLE II. R_1: MODULE RELATION FOR m_1

a_1	a_2	a_3	a_4	a_5
0	0	0	1	1
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

A. Standalone Module Privacy

A module m in a workflow takes a set I of input attributes and produces a set O of output attributes. Module m is modelled as a relation R over a set of attributes $A = I \cup O$ that satisfies the functional dependency $I \rightarrow O$. The domain of each attribute $a \in A$ is denoted by Δ_a. Then the *domain* of module m is given by $\prod_{a \in I} \Delta_a$ and the *range* is given by $\prod_{a \in O} \Delta_a$. Therefore the relation R represents the function m which is defined from that domain to range. A tuple in relation R represents a single execution of m where for every tuple $t \in R$, $\Pi_O(t) = m(\Pi_I(t))$ (Π denotes the projection for a single tuple). Figure 1 shows a simple workflow example taken from [1], which has modules m_1, m_2 and m_3 with boolean input and output attributes. Table I shows the relation R which represents the above workflow. Each tuple in R shows an execution of the workflow. Table II shows the relation R_1 which represents the module m_1 and shows the input and output attributes. Module m_1 computes $a_3 = a_1 \vee a_2$, $a_4 = \neg(a_1 \wedge a_2)$ and $a_5 = \neg(a_1 \oplus a_2)$ using inputs a_1 and a_2.

To ensure standalone module privacy, Davidson et al. comes up with an approach in which some attributes of the provenance relation are hidden. They call the final reduced relation a *Provenance View*. The subset of attributes to be hidden is carefully selected to minimize the cost of hidden information. In formal terms, R is projected on a restricted subset V of

TABLE III. R_V: A PROJECTION OF R_1

a_1	a_3	a_5
0	0	1
0	1	0
1	1	0
1	1	1

attributes to compute the final provenance view $R_V = \Pi_V(R)$. To make sure that R_V is secure enough, the required level of privacy is quantified by Γ-Privacy. The definition of *possible worlds* which leads to the definition of Γ-Privacy is as follows.

"The set of possible worlds for R w.r.t. a restricted subset of attributes V, denoted Worlds(R,V), consists of all relations R′ over the same schema as R where $\Pi_V(R') = \Pi_V(R)$*"*.

Table III shows a projection of R_1 with respect to the set of visible attributes $V = \{a_1, a_3, a_5\}$. There are many relations over the same schema as R_1 which belongs to Worlds(R_1, V). Obviously, R_1 is one of those and we can come up with lot more by changing values of a_2 and a_4 in R_1 as those attributes are not in V. Based on the notion of *possible worlds*, Γ-Privacy is defined for a given parameter $\Gamma \geq 1$. A provenance view R_V is Γ-standalone-private if for every $t \in R$, the possible worlds *Worlds(R,V)* contain at least Γ distinct output values that could be the result of $m(\Pi_I(t))$. In other words Γ-standalone-privacy means that for any input, the observer cannot guess *m*'s output with probability greater than $\frac{1}{\Gamma}$. For example, consider the Γ-privacy of relation R_1 with respect to the visible set $V = \{a_1, a_3, a_5\}$. It can be shown that R_1 is Γ-private for Γ = 4. As an example tuple, if we take *x* = (0,0), the set of distinct output values is $\{(0,\underline{0},1), (0,\underline{1},1), (1,\underline{0},0), (1,\underline{1},0)\}$. Underlined is the hidden attribute a_4. This can be shown for all cases of *x*.

B. Workflow Module Privacy

A workflow which consists of *n* modules is modelled as a relation *R* over the set of attributes $A = \cup_{i=1}^{n} (I_i \cup O_i)$, where each tuple in *R* represents an execution of the workflow and for every $t \in R$, and every $i \in [1,n]$, $\Pi_{Oi}(t) = m_i(\Pi_{Ii}(t))$. A sample workflow relation is given in Table I. Like in module privacy, a workflow relation *R* is projected on a restricted subset *V* of attributes to compute the final provenance view $R_V = \Pi_V(R)$. A workflow can contain both private modules and public modules. Therefore, the provenance view R_V must preserve the behaviour of public modules while preserving Γ-privacy for private modules. Based on that, the *possible worlds* for a workflow relation is defined as follows.

"The set of possible worlds for the workflow relation R w.r.t. a restricted subset of attributes V, denoted Worlds(R,V), consists of all relations R′ over the same schema as R where $\Pi_V(R') = \Pi_V(R)$ *and for every public module* m_i *and every tuple* $t' \in R'$, $\Pi_{Oi}(t') = m_i(\Pi_{Ii}(t'))$*"*.

This notion of *possible worlds* leads to the definition of Γ-workflow-privacy for a given parameter $\Gamma \geq 1$. A view R_V is Γ-workflow-private if for every tuple $t \in R$, and every private module m_i in the workflow, the possible worlds *Worlds(R,V)* contain at least Γ distinct output values that could be the result of $m_i(\Pi_{Ii}(t))$.

Γ-workflow-privacy for each module in a workflow can be achieved by hiding different subsets of attributes in workflow relation *R*. But the best approach is to maximize the amount of provenance information revealed to the user. Therefore, a cost is associated with each attribute in the relation and the safe subset *V* of attributes is selected such that the total cost of hidden attributes is minimized. The problem of computing the minimum cost subset is called the `Secure-View` problem.

III. LIMITATIONS OF Γ-PRIVACY

Before moving onto limitations within Γ-privacy model, here we discuss its limited applicability first. Davidson et al. argue that the functionality of a proprietary module in a workflow can be understood by observing the input and output attributes available in the provenance trace of the module. This argument is valid for a simple workflow module like the Boolean functions used in Figure 1. In that case all inputs and outputs for the module are captured as provenance. However, in real scientific workflows, input and output data are more complex. For example, most modules consume input data files with a set of parameters and produce some output data files. In such cases, provenance can only capture metadata about input and output data, but not the actual data. Therefore, in most real world workflows module privacy is not challenged by exposing provenance information. That can only happen in workflows where actual input and output data are simple enough to be captured as provenance. And also, Γ-privacy model works only for modules in which all input and output attributes have finite domains. This is also a very restrictive assumption for majority of the workflows. Therefore, we think that the applicability of Γ-privacy is limited when it comes to real workflows. But we do not try to address this limitation in this paper and concentrate only on limitations within the model.

As pointed out by Cheney and Perera [2], it is not clear how to choose a good value for Γ. User has to decide the value of Γ depending on the level of security required by the application. According to the definition of Γ-privacy, it makes sure that the correct output cannot be guessed by an attacker with a probability more than $\frac{1}{\Gamma}$. However, this probability varies from module to module. For example, for boolean modules shown in Figure 1, it might be sufficient to set Γ = 2 because hiding only one attribute will be enough to prevent the observer from guessing the output. But for a module which has five input attributes and five output attributes with varying domain sizes, it is not easy to decide a Γ value. And also, in Γ-workflow-privacy, same Γ value is used for all the modules in the workflow. For a complex workflow with modules with varying domain sizes, this can create lot of issues. If the Γ value is too low, complex modules in the workflow will not be sufficiently secured. If the Γ values is too high, simple modules have to be completely hidden to preserve Γ-privacy. Therefore, setting a Γ value for a complex workflow is even harder.

It is a very common use case to have a set of specific attributes that must be hidden in the published provenance data. In Γ-privacy, owner of the provenance relation cannot control the attributes selected to be hidden after setting some value

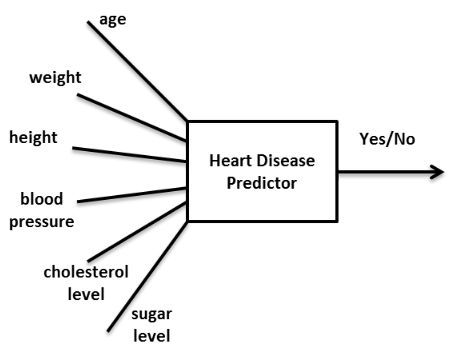

Fig. 2. A Heart Disease Predictor Module

for Γ. Except for considering cost of hiding, all attributes are treated equally by the Γ-privacy model. Cost of each attribute is measured from the end users perspective to minimize the value of hidden data. That cannot be used by the provenance owner to guarantee that a particular attribute is in the published provenance relation. For example, consider the *Heart Disease Predictor* module shown in Figure 2 which takes a set of attributes from a person's health record and produces a decision. Height of a person is a minor factor when deciding whether that person has got a heart disease. But attributes like blood pressure and cholesterol Level are major factors which contribute most to the output. Γ-privacy model does not take such differences into account in hiding attributes. It may achieve same Γ by hiding either height or blood pressure as it assumes both attributes contribute equally to the output. In this case, if the owner wants to hide blood pressure values in the published provenance data, there is no way of specifying that.

The Γ-privacy model assumes a fixed domain size for the module relation *R* when analysing the complexity of the algorithm which decides whether a given set of visible attributes is safe for a particular module. Cheney and Perera [2] highlights this as another limitation in Γ-privacy. However, even with this assumption, the algorithm is NP-hard in the number of attributes of the module and the complexity is $O(2^k N^2)$ where *k* is the number of attributes and *N* is the assumed constant size of the relation. If we try to avoid using the fixed domain size assumption, the upper bound of *N* can be calculated as $N = d^j$ where *d* is the maximum domain size out of input attributes and *j* is the number of input attributes. When applied this value, the complexity of the algorithm becomes $O(2^k d^{2j})$. Now if we assume that the number of input attributes is roughly half of the total number of attributes, this can be simplified as $O(2^k d^k) = O((2d)^k)$. This shows that even without the fixed domain size assumption, the algorithm still takes exponential time in the number of attributes. Therefore, this assumption does not make much difference to the complexity analysis. And also, Davidson et al. argue that the exponential time algorithm is not going to affect most workflows because the number of attributes for most modules is fairly small. As that is a fair assumption for most cases, we are not trying to address the complexity issue in this paper.

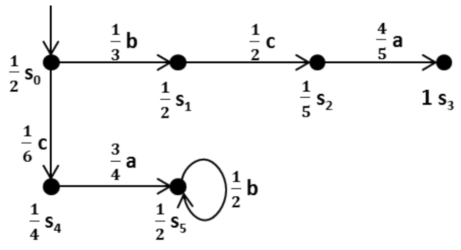

Fig. 3. A Probabilistic Labelled Transition System

IV. Evaluation of Similar Techniques

A. Quantitative Opacity

Opacity [13] has been identified as an important property that can be used to measure security of a system. Bryans et al. [14] applies opacity into transition systems following a qualitative approach. Quantitative opacity [11] is an extension of that work which quantifies opacity in probabilistic labelled transition systems (PLTS). When applying opacity for transition systems, a single execution of the system is called a run (denoted by λ) and the output is called the observation which is given by an observation function *obs*. Certain secure properties of the system which should not be determined by the observer are called predicates. A predicate is said to be opaque if an observer of the system cannot determine the satisfaction of the predicate using the observations of a given run of the system. When measuring the opacity of a predicate, a set ϕ of runs is identified for which the predicate holds.

Qualitative Opacity. A set of runs ϕ is *opaque* w.r.t. *obs* if, for every run $\lambda \in \phi$, there is another run $\lambda' \notin \phi$ such that $obs(\lambda) = obs(\lambda')$, i.e., λ' covers λ.

The idea behind this definition of qualitative opacity from [14] is that, for a single run to be opaque, there must be another run which can produce the same observations (cover). A sensitive predicate is associated with a set of runs ϕ and if all runs in ϕ are covered, the predicate is opaque. According to this definition, having a single covering run for each run in ϕ is enough for the predicate to be opaque.

Quantitative opacity is defined for probabilistic labelled transition systems and it extends the above definition by taking probability of runs into account. Figure 3 shows a sample PLTS adopted from [14]. A PLTS consists of a set of states and a set of labels between states. Each state is associated with a probability distribution which assigns probabilities for each connected label. First two notions of quantitative opacity are similar to qualitative opacity. A set of runs ϕ is said to be π-opaque when the probability of having an uncovered run is zero and π_ξ-opaque when the probability of having an uncovered run is ξ.

$\overline{\pi}_\gamma$**-opacity.** A set of runs ϕ is said to be $\overline{\pi}_\gamma$-opaque if the probability of ϕ^{cov} is γ times larger than the probability of ϕ where ϕ^{cov} is the set of runs which cover at least one run in ϕ and γ is much greater than one.

Quantitative opacity follows a similar idea to Γ-privacy. Both techniques measures the level of uncertainty in publishing sensitive data. Higher uncertainty leads to better privacy in both techniques. But Γ-privacy is a technique which is capable of making a given workflow provenance relation Γ-private for a given value of Γ by hiding attributes. But quantitative opacity is only capable of measuring the level of privacy of a given system. If π-opacity is used to find a value for Γ in Γ-privacy for a single workflow module, π-opacity is equivalent to Γ = 2 case. When considering a workflow, two models can be compared by treating a module in a workflow as a state in a PLTS. Even in that case, Γ-privacy is a much stronger notion compared to π-opacity. To achieve π-opacity, the system needs only one covering run for each sensitive run. A covering run can be created by changing only one module in the original run. But to achieve Γ-workflow-privacy, each module in the workflow must be Γ-private. Therefore, Γ-privacy for even Γ = 2 ensures much better privacy for a system compared to π-opacity. However, $\overline{\pi}_\gamma$-opacity can provide better security for higher γ values as it increases the number of covering runs for a given sensitive run. Even in that case, if a similar value is selected for Γ and γ, Γ-privacy provides much stronger security. Here we highlight that selecting a suitable value for γ in $\overline{\pi}_\gamma$-opacity is also not straightforward and depends on the application like in selecting a value for Γ in Γ-privacy. Therefore, we think it is hard to use the techniques from $\overline{\pi}_\gamma$-opacity to address the limitations of Γ-privacy.

B. *Differential Privacy*

Differential privacy [3] is a technique used to preserve the privacy of individuals involved in statistical databases. The exact idea captured by differential privacy is that the risk of breaching one's privacy should not statistically increase as a result of participating in a statistical database. Differential privacy is an interactive privacy mechanism where the data collector provides an interface through which the users can execute queries about the data and get answers. Underlying query processor adds noise to original answers to preserve differential privacy. A noise function K_f is associated with each query f and that decides the amount of noise to be added in the result of query f. The noise function K_f depends on Δf which is the maximum difference between $f(\mathrm{D}_1)$ and $f(\mathrm{D}_2)$ for all sub datasets D_1, D_2 in the dataset differing in at most one element. The idea here is to add an amount of noise which is proportional to the largest difference that can be made to the answer by a single data entry. The query processor first computes the correct answer and uses the noise function K_f to add noise to it.

Like in Γ-privacy, the intention behind differential privacy is also to increase the uncertainty to preserve data privacy. To increase the amount of uncertainty, Γ-privacy uses the technique of hiding attributes while differential privacy adds noise. The idea of adding noise can also be used in Γ-privacy to increase the value of Γ for a module. We use this technique in our solution presented in the next section. However, provenance information are mostly used for purposes like reproducing and failure tracing in workflows. Excessive noise can ruin the purpose of collecting provenance. Therefore, the technique of adding noise must be used carefully only when there is no better option.

C. *Quantitative Information Flow Security*

Information flow security [15] is a concept used to assess the amount of private information leaked from a high security variable to a low security variable during a process within a system. Values of low security variables are always visible to the users. The *non-interference* [16] property enforces a system to have zero information leakage from high security variables to low security variables. This property is too strict to be practical for most programs. For example, a password checker program reveals some information about the password to an attacker when it rejects a wrong password. Therefore, measuring the information leakage is important to enforce bounds within programs. Quantitative information flow security [12] is an attempt to automatically determine the leakage of information from high security variables to low security variables in a program using syntax-directed rules. In other words, it measures the amount of information initially in high security variables which an attacker can learn by observing a run of a program.

Information flow security concept is mostly related to programming languages. However, the idea behind it can be applied to Γ-privacy as well. The Γ-privacy model identifies a set of high security attributes H for a given Γ value and a attribute cost vector. The set of other attributes in the workflow provenance relation which is published to the users become the low security attributes L. It completely hides all attributes in H and exposes all attributes in L. If the user can learn some information about attributes in H by observing the attributes in L, Γ-privacy will not be preserved. There are two types of workflows to be considered here. For workflows in which all modules are private, no information leakage is possible. However, for mixed workflows which contains both private and public modules, values of high security attributes can be leaked through low security attributes in public modules during data sharing. To prevent that, Davidson et al. use a privatization technique for public modules in [1]. In their latest work [9], they propose a propagation technique to address information leakage more precisely. Therefore, Γ-privacy always preserves the *non-interference* property.

V. Solution for the Limitations of Γ-privacy

As discussed above, there are number of limitations in the Γ-privacy model. Here we present a solution which addresses three major limitations: (1) difficulty in selecting a suitable value for Γ, (2) using the same Γ value for complex workflows which consists of modules with varying domain sizes and (3) inability to specify the attributes that must be hidden. The power of Γ-privacy model is that it always guarantees that the correct output for any input in any module in the workflow cannot be guessed by an attacker with a probability more than $\frac{1}{\Gamma}$. Therefore, the solution to address the above limitations must always preserve this property.

A. *Proposed Solution*

From the provenance owners perspective, it is far more easier to select a set of sensitive attributes in the workflow provenance relation than directly deciding a suitable Γ value for the entire workflow. Therefore, in our solution, first we allow the user to select the attributes that must be hidden in the provenance relation to be published. The attributes selected by the user can belong to different modules. Let the set of modules on which the user selected at least one attribute to be hidden be *M*. However, for most complex workflows which consists of a large number of modules, there will be lot of modules which do not belong to *M*. But still all the private modules in the system must be secured using a suitable Γ value. Therefore, the Γ values for modules in *M* are used to select a suitable Γ value for other private modules which are not in *M*.

Once the user selects the set of attributes to be hidden, system internally calculates the Γ values for the modules in *M*. For a given module relation *R* and a set of attributes to be hidden *H*, Γ value can be calculated by first calculating *Worlds(R,* $\overline{H}$*)* and then finding the input with minimum number of unique outputs in *Worlds(R,* $\overline{H}$*)*. As shown for `Safe-View` problem in [1], this computation takes exponential time in the number of attributes. However, following Γ-privacy analysis, we assume that the number of attributes on a given module is fairly small and the computation is feasible. For a simple example, again consider the workflow in Figure 1 and its provenance relation *R* in Table I. If the user selects attribute a_2 to be hidden, *M* = $\{m_1\}$ and *H* = $\{a_2\}$. Now the system has to calculate Γ for m_1. Module relation R_1 for m_1 is given in Table II and *Worlds(*R_1*,* $\overline{H}$*)* can be calculated by changing the value of a_2 for all tuples. Now the unique outputs in *Worlds(*R_1*,* $\overline{H}$*)* for each input *x* have to be counted to find Γ. Set of unique outputs for *x* = (0,0) is {(0,1,1), (1,1,0)}, for *x* = (0,1) is {(0,1,1), (1,1,0)}, for *x* = (1,0) is {(1,1,0), (1,0,1)}, for *x* = (1,1) is {(1,1,0), (1,0,1)}. Therefore, Γ = 2 for module m_1 w.r.t. *H* = $\{a_2\}$.

The calculated Γ values for modules in *M* are shown to the user so that she can have an idea about the relationship between the hidden attributes and related Γ values. User is given the option to change the set of hidden attributes according to the Γ value of each module in *M*. For example, some modules may show Γ = 1 even though one attribute is already hidden. In such cases, user may hide more attributes to increase Γ as Γ = 1 does not indicate any uncertainty. If changed, the new Γ values are calculated according to the changes. Once the user agrees on the set of Γ values for modules in *M*, next step is to assign a proper Γ value for the private modules which are in $\overline{M}$. The user is allowed to select a suitable Γ value for those modules based on the values finalized for modules in *M*. For example, if the user thinks that the most important private modules are already covered by *M*, a sensible value for other private modules would be the minimum Γ value out of the values calculated for modules in *M*. However, the user is allowed to select any value according to her requirements.

However, selecting a fixed Γ value for modules in $\overline{M}$ can create problems in complex workflows which includes modules with small domain sizes as well as large domain sizes. If the attributes selected to be hidden by the user are having fairly large domain sizes, the calculated Γ values for the modules in *M* can also be large. And then if the user selects the minimum of those values as the Γ value for modules in $\overline{M}$, that can completely hide any modules in $\overline{M}$ with very small domain sizes. As a simple example, consider the provenance relation for an AND module *m* in Table IV. Suppose $m \in \overline{M}$ in a workflow provenance relation and Γ = 4. Here, the domain size of the output attribute a_3 is only 2. So the maximum Γ value that can be achieved by hiding only 2 attributes is 2. Therefore, Γ = 4 cannot be achieved for *m* without hiding all attributes a_1, a_2 and a_3. If all attributes are hidden that can be considered as the Γ = ∞ case as no information about the module will be revealed. The Γ-privacy model quantifies the cost of hiding all attributes of *m* as c(*Q*) = $\Sigma_{a \in Q}$c(*a*) where *Q* = $\{a_1, a_2, a_3\}$. But we argue that the cost of hiding all attributes of a module is greater than the total cost of individual modules because the provenance user will not know even the existence of such a module.

TABLE IV. A SAMPLE RELATION FOR AN AND MODULE

a_1	a_2	a_3
0	0	0
0	1	0
1	0	0
1	1	1

TABLE V. RELATION FOR THE AND MODULE WITH NOISE

a_1	a_2	a_3	a_4	a_5
0	0	0	1	0
0	1	0	1	1
1	0	0	0	1
1	1	1	0	0

To address this issue, we use the concept of adding noise to increase uncertainty of exposed data from differential privacy [3]. In case of the domain size of a particular module in $\overline{M}$ is not enough to satisfy the selected Γ value, new noise attributes are added to the provenance relation to increase the size of the domain. Domain sizes of the noise attributes must be carefully selected to sufficiently increase the domain size of the relation to satisfy the given Γ value. Adding a noise attribute is equal to hiding an actual attribute in Γ-privacy as it increases the number of possible *Worlds*. However, the noise attributes must be added to the set of visible attributes *V* in the final solution. Values for noise attributes have to be randomly generated for each tuple in the relation. Table V shows how the scenario in the above example can be handled by adding two noise attributes a_4 and a_5 into the provenance relation. Here a domain size of 2 is selected for both noise attributes a_4 and a_5 to match the other attributes in the module. Possible *Worlds* for this new relation can be expanded by assigning all possible values for a_4 and a_5. Now lets consider the set of unique outputs in *Worlds* for each possible input *x*. For *x* = (0,0), set of unique outputs is {(0,$\underline{0}$,$\underline{0}$), (0,$\underline{0}$,$\underline{1}$), (0,$\underline{1}$,$\underline{0}$), (0,$\underline{1}$,$\underline{1}$)} where the noise attribute values are underlined. Likewise, it can be shown that for each input, there are 4 unique outputs and hence Γ = 4. Displaying this relation is far more useful than completely hiding the entire relation.

B. `New-Secure-View` Problem

The involvement of the user to select a set of attributes to be hidden and to select Γ values for modules in *M*, introduces few changes into the original Γ-privacy model. As the Γ value is no longer constant for the entire workflow, new model maintains an array of Γ values for each private module in the workflow and $\gamma(m_i)$ returns the Γ value for module m_i. To preserve Γ-privacy of a workflow *W*, all modules in *W* must be Γ-private w.r.t. their own Γ values. Now the system has a set of attributes selected to be hidden even before applying the `Secure-View` algorithm given in [1]. Each module in *M* already has a fixed set of attributes to be hidden. There is no need to consider the cost of hiding attributes for the modules in *M* as those are selected by the user. So the `Secure-View` problem is reduced to finding the minimum cost safe subsets of attributes for modules which are not in *M* and combining them to find the safe subset for the entire workflow. In formal terms, we define the `New-Secure-View` problem for a workflow as follows.

`New-Secure-View` Problem. Given a workflow *W*, an array of Γ values for each private module in *W*, and a set of fixed hidden attributes *H*, find the set of visible attributes *V* w.r.t. which *W* is Γ-private, such that the cost $c(U) = \Sigma_{a \in U} c(a)$ is minimized where $U = \overline{V} \setminus H$.

The `Secure-View` problem is solved in [1] by calculating all safe subsets for each module in the workflow and then selecting one such subset for each module to minimize the cost of hidden attributes of the entire workflow. The solution of the `New-Secure-View` problem is slightly different because for the modules in *M*, the subset of attributes to be hidden is fixed. We leave the analysis of the algorithm for `New-Secure-View` problem for all-private and mixed workflows as a future work. The complexity of this problem should remain exponential in the number of attributes in the workflow provenance relation.

VI. Conclusion and Future work

In this paper, we have presented a modified Γ-privacy model to address the main limitations of the original Γ-privacy model. First we discussed the techniques behind Γ-privacy and then looked into all major limitations of the model. We highlighted that some of the limitations can reduce the applicability of the model in real world complex workflows. Then we evaluated three similar techniques used to preserve data privacy and discussed the applicability of those to address some limitations of Γ-privacy. Finally we presented our modified Γ-privacy model to address three main limitations of the original model and defined the `New-Secure-View` problem. We used the technique of adding noise to avoid completely hiding modules with small domain sizes in complex workflows. This model is far more easier to be used compared to the original Γ-privacy model from the users perspective. And also the technique of using noise to improve Γ for smaller modules is important to reduce the amount of hidden provenance data while preserving the strength of Γ-privacy.

Here we have only defined the `New-Secure-View` problem and have not evaluated the possible algorithms to solve it. Even though it will be close to `Secure-View` algorithm, it will be interesting to see the effect of fixed hidden attributes specially in the propagation model [9]. Another future work that we can think of is improving this model to address the other limitations of Γ-privacy to increase the applicability.

References

[1] S. B. Davidson, S. Khanna, T. Milo, D. Panigrahi, and S. Roy, "Provenance views for module privacy," in *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '11. New York, NY, USA: ACM, 2011, pp. 175–186. [Online]. Available: http://doi.acm.org/10.1145/1989284.1989305

[2] J. Cheney and R. Perera, "An analytical survey of provenance sanitization," *CoRR*, vol. abs/1405.5777, 2014.

[3] C. Dwork, "Differential privacy," in *Automata, languages and programming*. Springer, 2006, pp. 1–12.

[4] S. C. Dey, D. Zinn, and B. Ludäscher, "Propub: Towards a declarative approach for publishing customized, policy-aware provenance," in *Proceedings of the 23rd International Conference on Scientific and Statistical Database Management*, ser. SSDBM'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 225–243. [Online]. Available: http://dl.acm.org/citation.cfm?id=2032397.2032414

[5] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 18, no. 10, pp. 1039–1065, Aug. 2006. [Online]. Available: http://dx.doi.org/10.1002/cpe.v18:10

[6] T. Oinn, M. Addis, J. Ferris, D. Marvin, T. Carver, M. R. Pocock, and A. Wipat, "Taverna: A tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, p. 2004, 2004.

[7] Y. L. Simmhan, B. Plale, and D. Gannon, "A framework for collecting provenance in data-centric scientific workflows," in *Proceedings of the IEEE International Conference on Web Services*, ser. ICWS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 427–436. [Online]. Available: http://dx.doi.org/10.1109/ICWS.2006.5

[8] Komadu. [Online]. Available: http://d2i.indiana.edu/provenance_komadu/

[9] S. B. Davidson, T. Milo, and S. Roy, "A propagation model for provenance views of public/private workflows," in *Proceedings of the 16th International Conference on Database Theory*, ser. ICDT '13. New York, NY, USA: ACM, 2013, pp. 165–176. [Online]. Available: http://doi.acm.org/10.1145/2448496.2448517

[10] S. B. Davidson, S. Khanna, S. Roy, J. Stoyanovich, V. Tannen, Y. Chen, and T. Milo, "Enabling privacy in provenance-aware workflow systems," 2011.

[11] J. Bryans, M. Koutny, and C. Mu, "Towards quantitative analysis of opacity," in *Trustworthy Global Computing*, ser. Lecture Notes in Computer Science, C. Palamidessi and M. Ryan, Eds. Springer Berlin Heidelberg, 2013, vol. 8191, pp. 145–163. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41157-1_10

[12] D. Clark, S. Hunt, and P. Malacaria, "Quantitative analysis of the leakage of confidential data," *Electronic Notes in Theoretical Computer Science*, vol. 59, no. 3, pp. 238–251, 2002.

[13] R. D. Bailliage and L. Mazar, "Using unification for opacity properties," in *In Proceedings of the Workshop on Issues in the Theory of Security (WITS04*, 2004, pp. 165–176.

[14] J. W. Bryans, M. Koutny, L. Mazar&#x00e9;, and P. Y. A. Ryan, "Opacity generalised to transition systems," *Int. J. Inf. Secur.*, vol. 7, no. 6, pp. 421–435, Oct. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10207-008-0058-x

[15] Information Flow Security. [Online]. Available: http://en.wikipedia.org/wiki/Information_flow_(information_theory)

[16] Non-interference. [Online]. Available: http://en.wikipedia.org/wiki/Non-interference_(security)