Towards Generalization of Privacy Policy Specification and Property-based Information Leakage

Dileep Kumar Koshley, Sapana Rani and Raju Halder

Indian Institute of Technology Patna, India
{dileep.pcs15,sapana.pcs13,halder}@iitp.ac.in

Abstract. In spite of deep and intensive research, the existing data privacy preserving approaches primarily suffer from the lack of generality. Some solutions deal with direct information leakage, whereas others deal with indirect information leakage which occurs due to the presence of data or functional dependencies. Moreover, privacy policy specification supported by individual method has limited expressibility, which allows to express very specific forms of privacy concerns. In this paper, we formalize a privacy-preserving policy specification language which is highly expressive to adapt a wide range of constraints in various forms, fitting suitably to the real world scenarios. Furthermore, we introduce a new form of dependency, known as *Property-based Dependency*, which may also cause an indirect information leakage. Finally, we propose a preventive solution, on top of the existing ones, for privacy policies expressed in our proposed language.

Keywords: Data Privacy, Information Leakage, Policy Specification Language, Abstraction

1 Introduction

Privacy of sensitive database information in any information system has always been a prime concern to the scientific community over the past several decades. In the modern age of information, when amount and variety of data is increasing at an unprecedented rate, the evolution of technological solutions to meet various demands on data, *e.g.* publishing, sharing, storing into external disk-space, *etc.*, give rise to many research challenges on its privacy in relational databases [1, 2]. A widely accepted solution to this problem is to disclose sensitive information in such a way that illegitimate users can not infer anything about sensitive information in an exact way. A number of methods providing solutions to fulfill this objective are proposed. They include k-anonymity [3], l-diversity [4], t-closeness [5], *etc.* Their intent is to release sensitive information in an anonymized form. In k-anonymized dataset, each record is made identical to at least k - 1 other records with respect to those attributes acting as quasi-identifiers [3]. The principle of l-diversity ensures the presence of at least l distinct values for the sensitive field

in each equivalence class [4]. The t-closeness model [5] extends the l-diversity model by treating values of an attribute distinctly taking into account the data distribution for that attribute.

In addition to the privacy need of the sensitive data, Vimercati et al. [6] in their notable work first considered the visibility requirement as well. The sensitive information after fragmentation is released as "loose associations" in order to guarantee a specified degree of privacy. Later, they described how the publication of multiple loose associations between different pairs of fragments expose sensitive associations [7, 8].

Beside direct information leakage as addressed by the above techniques, there is a possibility of indirect information leakage due to the presence of non-sensitive information in public domain. Interestingly, Sweeney [3] showed that combinations of few characteristics, *e.g.* 5-digit ZIP, gender, date of birth, likely made 87% of US population unique. She combined the medical data with voter list and was able to identify the information leakage. There exists a series of works [9–11] addressing indirect information leakage which occurs due to the presence of functional dependencies (`FD`) and data dependencies (`DD`). Authors in [9] studied the privacy threat resulting from the full functional dependency as part of adversary knowledge, and they defined a privacy model, called (d, ℓ)-inference, as a way of prevention. [10] suggested the encryption of a small amount of non-sensitive data to defend `FD`-attack. The authors in [11] proposed fragmentation as a solution to prevent information leakage from data dependency.

Unfortunately, despite of many proposals in the literature as briefed above, this is observed that none of them is able to address different possible way of information leakage in a unified way. Some deal with direct information leakage, whereas others deal with indirect information leakage. Moreover, privacy policy specifications supported by individual method has limited expressibility. This allows only to express very specific privacy concerns which do not often fit to the real world scenarios. For example, [12] expresses the confidential information at cell level, whereas [10] and [8] express the confidentiality in the form of selection-projection and attribute-association respectively. Consequently, this gives rise to the scalability problem, resulting into a failure of adapting one solution to address the other. Observe, for example, that the solutions in [3–5] are not scalable enough to adapt as a solution to the visibility constraints. Furthermore, our observation revealed that the absence of data and functional dependencies may not always guarantee the absence of sensitive data leakage in database systems. This is due to the fact that, in reality, properties (instead of values) of some attributes may often depend on the properties or values of other attributes. This may lead to a partial inference about sensitive data. For example, an attacker who is able to observe the properties "*Age Group*" and "*Deficiency*" from the values of the attributes "*Age*" and "*Food Habit*" respectively, may guess some symptoms about patient's disease.

Addressing these concerns, we propose in this paper a privacy preserving scheme for sensitive information in relational databases, considering more generic scenarios where multiple form of privacy constraints exist. Furthermore, we in-

troduce a new kind of dependency – *property-based dependency* (PD) – that may lead, in addition to FD- and DD-dependences, a leakage of sensitive information.

To summarize, our main contributions in this paper are:

- Formalization of privacy policy specification language which is highly expressive so as to adapt a wide range of privacy concerns.
- To propose a solution, on top of the existing ones, that respects the privacy concerns expressed in our policy specification language.
- Identification of a new kind of dependency, named *property-based dependency* (PD), and to propose a solution to this implicit leakage using ordered binary decision diagram (OBDD).

The structure of the paper is as follows: Section 2 describes briefly the existing privacy preservation techniques in the literature. In Section 3, we introduce a generic form of privacy-policy specification language for relational databases in Back-us Naur form. In Section 4, we discuss how to remove redundancy from a set of privacy policies specified in our specification language by naive users. In Section 5, we propose a solution, on top of the existing ones, to prevent explicit information leakage from relational databases. Section 6 elaborates how information may implicitly be leaked due to dependencies. Section 7 introduces *property-based dependency*. In Section 8, we propose a unified solution for implicit leakage covering all three dependencies. We compare our proposals *w.r.t.* the literature in Section 9. Finally, we conclude our work in Section 10.

2 Related Works

Releasing non-sensitive information while maintaining confidentiality of the sensitive one is becoming extremely challenging due to the exponential growth in the number and variety of data collections. Broadly to prevent unauthorized information leakage there is a need to disclose sensitive information in such a way that illegitimate users can not infer anything about sensitive information in an exact way.

A wide range of privacy preserving solutions [3–5, 8–12] is proposed in the literature. Each of these proposals refer to very specific form of constraints. The authors in [3] identified some set of attributes as quasi-identifiers and then ensure that each sequence of values for these quasi-identifiers appears with at least k-occurrences. In [4], the equivalence classes are having at least l distinct values for the sensitive attribute. In order to overcome the limitations in [4], the authors in [5] proposed a solution which requires that the distribution of a sensitive attribute in any equivalence class should be close to the distribution of the attribute in the overall table. The solutions proposed in [3–5] consider only the confidentiality constraints: they considered neither visibility constraint nor any dependency-based leakage. In [8], the authors partitioned the database relation and published universal association at group level to increase the probability of anonymity. Authors in [10] proposed a solution based on the encryption of evidence records which may facilitate an information leakage about sensitive

records. In a different research line, to address implicit leakage, the authors in [9, 10] addressed the functional dependency, whereas [11] dealt with only data dependency. In particular, [11] proposed graph based fragmentation of database relations. The authors in [10] expressed confidentiality constraints in selection-projection form only. The cell level confidential constraints were addressed in [12] by masking the sensitive cells with a specific type of variables. The authors in [13] proposed the concept of δ-dependency in order to deal with hierarchical sensitive data taken from a hierarchical tree structure where data values become more specific as we move down the tree. The authors in [14] surveyed data inference problems resulted from indirect accesses to sensitive data through inferences.

This is worthwhile to mention here that our current policy language doesn't cover any access control policy as proposed in [15, 16]. Although in this work we refrain ourselves from considering access control policy specifications, however in future we plan to extend our work to include this part as well.

3 Privacy Policy Specifications

In this section, we formalize a privacy-preserving policy specification language which is highly expressive so as to cover a wide range of constraints defined on relational databases, considering the real-world situations.

We use Back-us Naur Form of Context-free Grammars [17] to express our specification language. This allows us to define two kind of policy specifications: *explicit/direct* and *implicit/indirect*. Explicit or direct policy specification refers to the part of data which are sensitive and prone to direct leakage. Indirect or implicit policy specification expresses some dependency information which may lead to an indirect information leakage. Observe that database owners are responsible to specify only explicit policy specification, whereas database designers are responsible for implicit policy specification.

3.1 Explicit/Direct Policy Specification

Explicit policy specification can be formalized in two forms: confidentiality and visibility constraints. Let $\mathtt{DB}$ be a relational database consisting of relations $\mathtt{T}_1, \mathtt{T}_2, ..., \mathtt{T}_n$. Let $\mathtt{A}_i = \mathtt{attribute}(\mathtt{T}_i)$ be the set of attributes of T_i and $\mathtt{A}_{\mathtt{DB}} = \bigcup_{i \in 1,...,n} \mathtt{A}_i$. The confidentiality and visibility constraints are formalized as follows:

Confidentiality Constraints: We consider various possible forms of constraints to include in our specification language as confidential constraints. They include *attribute-association*, *selection-projection*, *cell-level*, and their combinations. The confidentiality constraints $\mathtt{P}_c$ is formalized below:

$$\mathtt{P}_c \ ::= \ \mathtt{g} \mid \pi_{\mathtt{g}}\sigma_{\phi} \mid [\![\mathtt{g}]\!]_{\mathtt{Val(PK(g))}} \mid \mathtt{P}_c \wedge \mathtt{P}_c.$$

where $\mathtt{g} \in \wp(\mathtt{A}_{\mathtt{DB}})$ is a subset of $\mathtt{A}_{\mathtt{DB}}$ which represents that all attributes in $\mathtt{g}$ *must not be contained* in the same relation. Confidential policy $\pi_{\mathtt{g}}\sigma_{\phi}$ in terms

Table 1: "*Patients*"

SSN	Name	Age	Gender	ZIP	Food_Habit	Disease
101	Alice	4	M	100025	cereals, carrot, fish oil, yoghurt	Kwashiorkor
102	Bob	26	F	110091	cereals, fish, yoghurts	Night Blindness
103	Carol	39	M	100025	pulses, yellow fruits, cranberries	Osteomalacia
104	David	3	F	123001	pulses, yellow fruits, cranberries	Rickets
105	Ela	22	M	110091	pulses, yellow fruits, cranberries	Ostemalacia
106	Frudo	5	M	110091	pulses, milk, yellow fruits	Goiter
107	Gram	45	F	123001	cereals, fish, yoghurts	Cataract

of selection-projection operation represents that data for attributes $\mathtt{g}$ satisfying condition ϕ are *confidential*. $[\![\mathtt{g}]\!]_{\mathtt{Val(PK(g))}}$ specifies a fine-grained *confidential* constraint in terms of sensitive data cells. The data cells are uniquely represented by the attributes $\mathtt{g}$ and the tuple's primary key values $\mathtt{Val(PK(g))}$, where $\mathtt{PK(g)}$ denotes primary key (computed as $\mathtt{PK(g)} = \mathtt{Primary\_Key}(\bowtie_i \mathtt{T}_i)$ applying natural join $\bowtie$ on all $\mathtt{T}_i$ such that $\exists\ \mathtt{a} \in \mathtt{g}.\ \mathtt{a} \in \mathtt{atribute}(\mathtt{T}_i)$) and $\mathtt{Val(PK(g))}$ denotes the set of primary key values corresponding to $\mathtt{PK(g)}$.

Visibility constraints: We formalize the visibility constraints $\mathtt{P}_v$ as:

$$\mathtt{P}_v \ ::= \ \mathtt{g} \mid \pi_{\mathtt{g}}\sigma_{\phi} \mid [\![\mathtt{g}]\!]_{\mathtt{Val(PK(g))}} \mid \mathtt{P}_v \wedge \mathtt{P}_v \mid \ \mathtt{P}_v \vee \mathtt{P}_v.$$

where $\mathtt{g} \in \ \wp(\mathtt{A}_{\mathtt{DB}})$ is a subset of $\mathtt{A}_{\mathtt{DB}}$ which represents that all attributes in $\mathtt{g}$ *must be present* in at least one of the relations. Visibility policy $\pi_{\mathtt{g}}\sigma_{\phi}$ in terms of selection-projection operation represents that data for attributes in $\mathtt{g}$ satisfying condition ϕ must be *visible*. The fine grain visibility constraint $[\![\mathtt{g}]\!]_{\mathtt{Val(PK(g))}}$ specifies that data in cells uniquely identified by $\mathtt{g}$ and its corresponding primary key values $\mathtt{Val(PK(g))}$ are *visible*. Observe that, unlike confidential constraints, the policy language allows visibility constraints to express in the form of OR ($\vee$). For example, the visibility constraint *Phone_No* $\vee$ *Email_ID* specifies that either phone number or email ID or both must be present in one of the database relations.

Example 1. Consider the relation "*Patients*" shown in Table 1. Consider a composite confidentiality constraint $\mathtt{P}_c = \{Name, Disease\} \wedge \{SSN\} \wedge \{Age, ZIP, Gender\} \wedge \pi_{Disease}\sigma_{Gender=F} \wedge [\![Disease]\!]_{(104)}$ and visibility constraint $\mathtt{P}_v = \{Name, Age\} \wedge \pi_{Age}\sigma_{Gender=M} \wedge [\![Disease]\!]_{(101)}$. Let us now illustrate different forms of explicit constraints using $\mathtt{P}_c$ and $\mathtt{P}_v$.

Attribute-association: {*Name, Disease*} is an example of confidentiality constraint in the form of attribute association. This specifies that the combination of name and disease information is confidential and must not be present together when releasing. Similarly, a visibility constraint {*Name, Age*} represents

that these two attributes in combination must be released together in the same database relation.

Selection-projection: The confidentiality constraint $\pi_{Disease}\sigma_{Gender=F}$ represents that disease of all female must be confidential. Similarly, the visibility constraint $\pi_{Age}\sigma_{Gender=M}$ states that age of all male must be visible to public.

Cell-level: Cell level confidential constraint $[\![Disease]\!]_{(104)}$ represents that disease of patient having SSN (which is primary key) equal to 104 is confidential. Similarly, the visibility constraint $[\![Disease]\!]_{(101)}$ states that disease of patient having SSN equal to 101 must be publicly visible.

3.2 Implicit/Indirect Policy Specification

Implicit policy specification represents those which may lead to indirect information leakage. Various kind of dependencies (for example, data-dependency and functional dependency) may implicitly or indirectly leak sensitive-information. These are completely specific to the design of databases and their implementation. We use the notation [ ] and ⟨ ⟩ to represent data dependency and functional dependency respectively. Let R be the database relation where $\mathtt{X}, \mathtt{Y} \in \wp(\mathtt{attribute}(R))$ and $\mathtt{X} \cap \mathtt{Y} = \phi$.

Data Dependency (DD): We denote data dependency by $\mathtt{P_{DD}}$: $[\mathtt{X} \xrightarrow{DD} \mathtt{Y}]$. This represents that for each tuple $\mathtt{t} \in R$, `t(X)` determines `t(Y)`, where `t(X)` and `t(Y)` are the values of `X` and `Y` in tuple t.

Functional Dependency (FD): Functional dependency is denoted by $\mathtt{P_{FD}}$: $\langle \mathtt{X} \xrightarrow{FD} \mathtt{Y} \rangle$ which represents that for tuples $\mathtt{t_i}, \mathtt{t_j} \in R$, if $\mathtt{t_i(X)} = \mathtt{t_j(X)}$ then $\mathtt{t_i(Y)} = \mathtt{t_j(Y)}$.

The information leakage due to FD and DD are exemplified in Section 6. This is worthwhile to mention our observation that the absence of FD- and DD-based leakage does not always guarantee the preservation of privacy concerns, and in this context we introduce a new kind of dependency, named *property-based dependency*, which we will include in the specification later on.

4 Redundancy Removal from Explicit Policy Specification

The explicit policy specification provided by naive users may contain redundancy and inconsistency. In this section, we introduce an algorithm to remove the redundant explicit policy expressions, yielding a minimal form of policy specification.

Let us describe various cases where redundancy removal may take place: **Case-1:** Consider a database relation R. Let $\mathtt{P}_c = \mathtt{g}_0 \wedge \mathtt{g}_1 \wedge \cdots \wedge \mathtt{g}_n$ be a composite confidential constraint composed of a set of atomic constraints $\mathtt{g}_0$, $\mathtt{g}_1$, ..., $\mathtt{g}_n \in \mathtt{attribute}(R)$ each in attribute-association form. Let us suppose that

$\mathtt{g}_0$ is covered by the least upper bound of $S \subseteq \{\mathtt{g}_1, \mathtt{g}_2, \ldots, \mathtt{g}_n\}$. In this case, $\mathtt{g}_0$ is redundant as constraints in S are enough to express $\mathtt{g}_0$. Let us illustrate this with a simple example: suppose $\mathtt{attribute}(R) = \{w, x, y, z, u, v\}$ and $\mathtt{P}_c = \mathtt{g}_0 \wedge \mathtt{g}_1 \wedge \mathtt{g}_2 \wedge \mathtt{g}_3$ where $\mathtt{g}_0 = \{w, x, y\}$, $\mathtt{g}_1 = \{w, y\}, \mathtt{g}_2 = \{x, y, z\}, \mathtt{g}_3 = \{u, z, v\}$. This is depicted using Venn diagram in Figure 1(a). This is clear from the diagram that the constraint $\mathtt{g}_0$ is a subset of the least upper bound of $\mathtt{g}_1$ and $\mathtt{g}_2$, and therefore it is redundant in $\mathtt{P}_c$. **Case-2:** Given two confidentiality constraints $c_i = \pi_{\mathtt{g}_i}\sigma_{\phi_i}$ and $c_j = \pi_{\mathtt{g}_j}\sigma_{\phi_j}$ in $\mathtt{P}_c$. If $\phi_i \equiv \phi_j$, we can combine them into $\pi_{\mathtt{g}_i \sqcup \mathtt{g}_j}\sigma_{\phi_i}$. Similarly, if $\mathtt{g}_i = \mathtt{g}_j$, we can combine them into $\pi_{\mathtt{g}_i}\sigma_{\phi_i \vee \phi_j}$. This is depicted in Figure 1(b). **Case-3:** Given a composite confidentiality constraint $\mathtt{P}_c = \pi_{\mathtt{g}_0}\sigma_{\phi_0} \wedge \pi_{\mathtt{g}_1}\sigma_{\phi_1} \cdots \wedge \pi_{\mathtt{g}_n}\sigma_{\phi_n}$ which is composed of a set of atomic constraints each in selection-projection form. Let $S \subseteq \{\pi_{\mathtt{g}_i}\sigma_{\phi_i} \mid i = 1, \ldots, n\}$ such that $\mathtt{g}_0$ is the least upper bound of $\{\mathtt{g}_k \mid \pi_{g_k}\sigma_{\phi_k} \in S\}$ and ϕ_0 implies $\vee\{\phi_k \mid \pi_{\mathtt{g}_k}\sigma_{\phi_k} \in S\}$. Then we can say that $\pi_{\mathtt{g}_0}\sigma_{\phi_0}$ is subset of the union of S and hence it is redundant. An example of this considering three atomic constraints is depicted in Figure 1(c). **Case-4:** Given two cell-level confidential constraints $[\![g_i]\!]_{\mathtt{Val(PK}(g_i))}$ and $[\![g_j]\!]_{\mathtt{Val(PK}(g_j))}$ in $\mathtt{P}_c$, if $\mathtt{Val(PK}(g_i)) = \mathtt{Val(PK}(g_j))$ then we can combine them into $[\![g_i \sqcup g_j]\!]_{\mathtt{Val(PK}(g_i))}$.

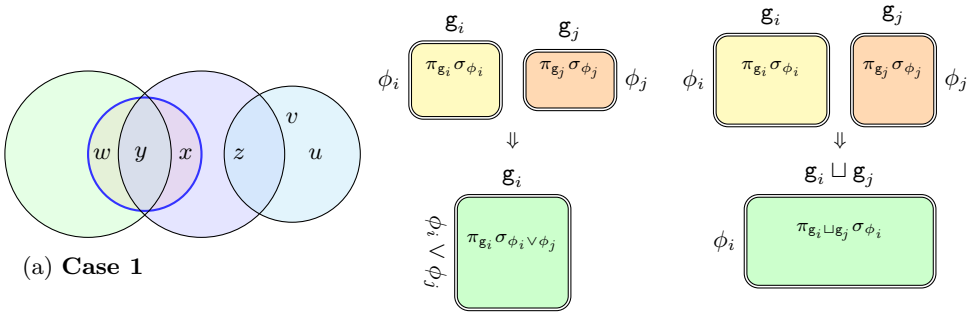


Fig. 1: Various cases of confidentiality constraints reduction

Case-5: If $\mathtt{P}_c = \mathtt{P}_{c_1} \wedge \mathtt{P}_{c_2}$ such that $\mathtt{P}_{c_1}$ and $\mathtt{P}_{c_2}$ are in different form then the redundancy can be removed based on the equivalence between them. For example, $\mathtt{P}_{c_1} = \pi_{\mathtt{g}}\sigma_{\phi}$ and $\mathtt{P}_{c_2} = \mathtt{g}$ are equivalent if ϕ is a tautology. In this case we can replace $\mathtt{P}_c$ by one of these equivalent constraints. **Case-6:** In case when a visibility constraint is contained within a confidentiality constraint, we always prioritize the confidentiality constraint considering the visibility constraint as

redundant. For example, let $\mathtt{P}_c = \mathtt{g}_0$ where $\mathtt{g}_0 = \{x, y\}$ and $\mathtt{P}_v = \mathtt{g}_1$ where $\mathtt{g}_1 = \{x, y\} \vee \{w, z\}$ then we can remove $\{x, y\}$ from $\mathtt{P}_v$.

A formal algorithm addressing above cases is depicted in Algorithm 1. Observe that similar algorithm to remove redundant visibility constraints considering these cases can easily be formalized this way. We skip it for space constraint.

Algorithm 1 SPEC_REDUCTION

Input: Explicit confidentiality constraint $\mathtt{P_c}$ which may contain redundancy
Output: Non-redundant confidentiality constraint

1: S:=**extractAtomicAA**($\mathtt{P_c}$); ▷ *Extracting all atomic constraints in attribute-association form.*
2: **for** all $\mathtt{g} \in S$ **do**
3: **if** $\mathtt{g}$ is least upper bound (lub) of $T \subseteq (S - \{\mathtt{g}\})$ **then**
4: $\mathtt{P_c}$=$\mathtt{P_c} \downarrow \{\mathtt{g}\}$ ▷ *Dropping $\mathtt{g}$ from P_c*
5: S:=**extractAtomicSP**($\mathtt{P_c}$); ▷ *Extracting all atomic constraints in projection-selection form.*
6: **for** all $\pi_{\mathtt{g}_i}\sigma_{\phi_i}$, $\pi_{\mathtt{g}_j}\sigma_{\phi_j} \in S$ **do**
7: **if** $\phi_i \equiv \phi_j$ **then**
8: $\mathtt{P_c} = (\mathtt{P_c} \downarrow \{\pi_{\mathtt{g}_i}\sigma_{\phi_i}, \pi_{\mathtt{g}_j}\sigma_{\phi_j}\}) \Uparrow \pi_{\mathtt{g}_i \sqcup \mathtt{g}_j}\sigma_{\phi_i}$ ▷ *Dropping $\{\pi_{g_i}\sigma_{\phi_i}, \pi_{g_j}\sigma_{\phi_j}\}$ and then adding $\pi_{g_i \sqcup g_j}\sigma_{\phi_i}$ to P_c*
9: **else if** $\mathtt{g}_i = \mathtt{g}_j$ **then**
10: $\mathtt{P_c} = (\mathtt{P_c} \downarrow \{\pi_{\mathtt{g}_i}\sigma_{\phi_i}, \pi_{\mathtt{g}_j}\sigma_{\phi_j}\}) \Uparrow \pi_{\mathtt{g}_i}\sigma_{\phi_i \vee \phi_j}$
11: S:=**extractAtomicSP**($\mathtt{P_c}$);
12: **for** all $\pi_{\mathtt{g}_i}\sigma_{\phi_i} \in S$ **do**
13: **if** $\exists\ T \subseteq (S - \{\pi_{\mathtt{g}_i}\sigma_{\phi_i}\})$ such that $\mathtt{g}_i$ is lub of $\{\mathtt{g}_k \mid \pi_{\mathtt{g}_k}\sigma_{\phi_k} \in T\}$ & ϕ_i implies $\vee\{\phi_k \mid \pi_{\mathtt{g}_k}\sigma_{\phi_k} \in T\}$ **then**
14: $\mathtt{P_c} = \mathtt{P_c} \downarrow \{\pi_{\mathtt{g}_i}\sigma_{\phi_i}\}$ ▷ *Dropping $\pi_{g_i}\sigma_{\phi_i}$ from P_c.*
15: S:=**extractAtomicCL**($\mathtt{P_c}$); ▷ *Extracting all atomic constraints in cell-level form.*
16: **for** all $[\![g_i]\!]_{\mathtt{Val(PK}(g_i))}$, $[\![g_j]\!]_{\mathtt{Val(PK}(g_j))} \in \mathtt{P}_c$ **do**
17: **if** $\mathtt{Val(PK}(g_i)) = \mathtt{Val(PK}(g_j))$ **then**
18: $\mathtt{P_c} = (\mathtt{P_c} \downarrow \{[\![g_i]\!]_{\mathtt{Val(PK}(g_i))}, [\![g_j]\!]_{\mathtt{Val(PK}(g_j))}\}) \Uparrow [\![g_i \sqcup g_j]\!]_{\mathtt{Val(PK}(g_i))}$
19: Apply equivalences between various forms of constraints to remove redundancy.

Example 2. Consider the relation "*Patients*" in our running Example 1. Consider the confidentiality constraint $\mathtt{P}_c$ and the visibility constraint $\mathtt{P}_v$ defined as: $\mathtt{P}_c = \{SSN\} \wedge \{Name, Age\} \wedge \{Age, Disease\} \wedge \{Name, Disease\} \wedge \{Name, Age, Disease\} \wedge \pi_{ZIP}\sigma_{Age \geq 10} \wedge \pi_{Name,ZIP}\sigma_{Age \geq 5}$ and $\mathtt{P}_v = \{Name\} \vee \{SSN\}$. Observe that the atomic confidentiality constraint $\{$*Name, Age, Disease*$\}$ is redundant since it is covered by least upper bound of the constraints $\{$*Name, Age*$\}$, $\{$*Age, Disease*$\}$ and $\{$*Name, Disease*$\}$. Similarly, the constraint $\pi_{ZIP}\sigma_{Age \geq 10}$ is redundant because $\{ZIP\} \subseteq \{Name, ZIP\}$ and $(Age \geq 10)$ *implies* $(Age \geq 5)$. The visibility constraint on "*SSN*" is filtered from the specification as it contradicts the confidentiality constraint on itself specified in $\mathtt{P}_c$. Therefore, the reduced form of confidentiality and visibility constraints are: $\mathtt{P}_c^{'} = \{SSN\} \wedge \{Name, Age\} \wedge \{Age, Disease\} \wedge \{Name, Disease\} \wedge \pi_{Name,ZIP}\sigma_{Age \geq 5}$ and $\mathtt{P}_v^{'} = \{Name\}$.

In the rest of the paper, we use the term "policy specification" which refers only the reduced form of policy specification.

5 Proposed Solution to Prevent Explicit Leakage

We are now in a position to propose solution to preserve privacy policies defined in our specification language. It is clear from the specification language that the proposed solution should aim to address any explicit constraints of the form *attribute-association*, *selection-projection*, *cell level* or their combination. We propose a simple way of solving this generalized form of privacy-preservation concern by adapting a combination of existing approaches proposed so far. To be more specific, our solution combines three existing solutions from [10, 12] and [6] respectively.

Before describing the algorithm, let us recall from [12] the notions of *Type-1* and *Type-2* variables:

Definition 1. Type-1 ***Variable [12].*** *A* Type-1 *variable is a symbol in some alphabet. Given two different* Type-1 *variables* v_1, v_2, "$v_1 = v_1$" *is true, while* "$v_1 = v_2$" *and* "$v_1 = c$" *are unknown, where* c *is a constant value.*

Definition 2. Type-2 ***Variable [12].*** *A* Type-2 *variable is given as* $\langle \alpha, d \rangle$, *where* α *and* d *are called the name and the domain of the variable, respectively. Given* α, β, d_1, d_2, "$\langle \alpha, d_1 \rangle = \langle \alpha, d_1 \rangle$" *and* "$\langle \alpha, d_1 \rangle \neq \langle \beta, d_1 \rangle$" *are true, while whether* "$\langle \alpha, d_1 \rangle = \langle \alpha, d_2 \rangle$", "$\langle \alpha, d_1 \rangle = \langle \beta, d_2 \rangle$", "$\langle \alpha, d_1 \rangle = v$" *or* "$\langle \alpha, d_1 \rangle = c$" *are unknown, where* v *is a* Type-1 *variable and* c *is constant value.*

Observe that *Type-2* Variables are used to mask sensitive cells while at the same time are also used to maintain a secure linking among various relational tables, whereas *Type-1* is used to mask only sensitive cells in a relation.

The overall algorithmic steps are depicted in Algorithm 2. In Step 1, we apply the reduction Algorithm 1 in order to obtain a minimal form of policy specification. In Step 2(a), cell level constraints are first addressed by replacing sensitive cells with their corresponding *Type-1* or *Type-2* variables, ensuring the non-disclosure of their exact values and at the same time maintaining the secure referential linking among various tables in the database [12]. Similarly, in Step 2(b), database-parts referred by selection-projection constraints are replaced by *Type-1* or *Type-2* variables with the similar aim. Lastly, in Step 3, the algorithm deals with attribute-association constraints by performing fragmentation and then publishing the loose-association at group level [6]. It is noteworthy to mention that the order of these steps are very crucial. For example, if any privacy constraint in the form of attribute-association is addressed first, then it may be impossible to respect any cell-level or selection-projection constraints on the same database. This is because when sensitive cells or sensitive database-parts are divided into different fragments, their corresponding cell-level or selection-projection constraint-expressions defined on previous relation may not be valid anymore.

Algorithm 2 Preventing Explicit Information Leakage

Input: Relation R, Privacy Policy Specification P in the form of *attribute-association* $\mathtt{g}$, *selection-projection* $\pi_{\mathtt{g}}\sigma_{\phi}$ and *cell level* $[\![\mathtt{g}]\!]_{\mathtt{Val(PK(g))}}$

Output: Relations satisfying P

1: Apply reduction algorithm in **Algorithm 1** and compute minimal form of policy specification.
2: For all $\mathtt{X} \in \mathsf{P}$, perform the following steps to obtain relation R':
2(a): *If* $\mathtt{X}$ *is in the form of* $[\![\mathtt{g}]\!]_{\mathtt{Val(PK(g))}}$:
If $\mathtt{X}$ acts as referential key linking other relation, then corresponding cell will be masked by *Type-2*. Otherwise, the value will be masked by *Type-1*.
2(b): *If* $\mathtt{X}$ *is in the form of* $\pi_{\mathtt{g}}\sigma_{\phi}$:
If any cell in $\mathtt{X}$ is involved in any referential integrity, then corresponding cell will be masked by *Type-2*. Otherwise, the value will be masked by *Type-1*.
3: For all $\mathtt{g} \in \mathsf{P}$, apply Fragmentation Algorithm in [6] on the relation R' which is obtained in step 2.

Example 3. Consider the relation "*Patients*" in the running Example 1. Consider the policy specification consisting of confidentiality and visibility constraints P_c and P_v on "*Patients*" defined in Example 1. The resulting database after applying Algorithm 2 is depicted in Table 2.

Table 2: Fragments and group association applying Algorithm 2 on "*Patients*"

(a) "F_1"

Name	ZIP	Age	Food_Habit	GID
Alice	100025	4	cereals, carrot, fish oil, yoghurt	g_1
Bob	110091	26	cereals, fish, yoghurts	g_2
Carol	100025	39	pulses, yellow fruits, cranberries	g_1
David	123001	3	pulses, yellow fruits, cranberries	g_3
Ela	110091	22	pulses, yellow fruits, cranberries	g_2
Frudo	110091	5	pulses, milk, yellow fruits	g_2
Gram	123001	45	cereals, fish, yoghurts	g_3

(b) $A_{F_1F_2}$

$\mathbf{GID_{F_1}}$	$\mathbf{GID_{F_2}}$
g_1	h_1
g_2	h_1
g_2	h_2
g_3	h_1
g_3	h_2

(c) "F_2"

GID	Disease	Gender
$h1$	Kwashiorkor	M
$h2$	α	F
$h1$	Osteomalacia	M
$h1$	$\top$	M
$h2$	β	F
$h1$	Goiter	M
$h2$	γ	F

Observe that Step 2(a) masks the sensitive cell represented by $[\![Disease]\!]_{(104)}$ with *Type-1* variable $\top$. Similarly, Step 2(b) masks the database part represented by selection-projection confidentiality constraint $\pi_{Disease}\sigma_{Gender=F}$ by other *Type-1* variables α, β, and γ.

In the third step, in order to preserve the constraints expressed in attribute-association form, the relation obtained in Step 2 is fragmented and all the fragments ("F_1" and "F_2") along with their group association ($A_{F_1F_2}$) are released, as shown in Table 2. Observe, as an instance, that Alice belongs to the group g_1

after fragmentation. It is clear from the group association Figure 2(b) that g_1 is associated with the group h_1 which contains four possible diseases, i.e. Kwashiorkor, Osteomalacia, Goiter and $\top$. This produces anonymized result when attackers try to infer the exact value of disease for Alice. For example, although attacker can infer that Alice may suffer from Kwashiorkor, Osteomalacia, Goiter or any of these diseases, but she will be unable to get exact value of the disease.

6 Functional and Data Dependency: A Gateway to Implicit Information Leakage

The purpose of this section is just to briefly recall the privacy concerns in presence of functional dependency (FD) and data dependency (DD). The authors in [10] and [11] explored the information leakage due to FD and DD respectively. In particular, as preventive measures, [10] proposed encryption of a small amount of non-sensitive data in order to nullify the effect of FD on the relation, whereas [11] applied fragmentation as a way to prevent leakage due to DD.

Examples 4 and 5 show how functional dependency (FD) and data dependency (DD), formalized in section 3.2, can serve as adversary knowledge and may cause privacy violation.

Example 4. Consider a hospital relation that stores its patients' details as given in Table 3(a). Suppose, Alice and Carol do not want to disclose their disease information, however Bob is ready to disclose his disease information. Suppose Table 3(b) is the resultant relation, after applying Algorithm 2, preserving these explicit privacy constraints.

Table 3: Leakage in $PD^{'}$ due to $\langle\{Disease\_Code\} \xrightarrow{FD} \{Disease\}\rangle$

(a) Original relation PD

Name	Disease_Code	Disease
Alice	HI15	HIV
Bob	CC22	Cancer
Carol	CC22	Cancer

(b) Masked relation $PD^{'}$

Name	Disease_Code	Disease
Alice	HI15	α
Bob	CC22	Cancer
Carol	CC22	β

Assume that the following functional dependency $F = \langle\{Disease\_Code\} \xrightarrow{FD} \{Disease\}\rangle$ exists. With the knowledge of F, an attacker can easily derive Carol's disease from Bob's record, as Bob and Carol have same disease-code resulting into same disease for both.

Example 5. Consider a relation "R" as depicted in Table 4(a) with confidential constraint $\mathtt{P}_c = \{Name, Disease\}$ and data dependency D: $[\{Birth, ZIP\} \xrightarrow{DD} \{Name\}]$. By following Algorithm 2, the resulting fragments satisfying $\mathtt{P}_c$ is shown in Table 4(b). Due to the presence of data dependency D, for each tuple an attacker can easily get the values of "$Name$" attribute if she knows the values of "$Birth$" and "ZIP" and hence the constraint $\mathtt{P}_c$ is violated.

Table 4: Leakage in R due to $[\{Birth, ZIP\} \xrightarrow{DD} \{Name\}]$

(a) Database Relation R

Name	Birth	ZIP	Disease	Drug	Dosage
Alice	11/01/2014	1111	soar throat	Diphen	6mg
Bob	12/11/1990	2222	sneezing	Beadryl1	20mg

(b) Correct fragmentation of R

Name	Birth	ZIP	Disease
Alice	11/01/2014	1111	soar throat
Bob	12/11/1990	2222	sneezing

Drug	Dosage
Diphen	6mg
Benadryl	20mg

7 Property-based Dependency: A New Adversarial Knowledge

In this section, we introduce a new kind of dependency which may also trigger implicit data leakage even though DD- and FD-based leakages are absent. We name this new kind of dependency as *property-based dependency*(PD). Let us start this section with the following research question:

"Does the absence of functional dependency and data dependency guarantee the absence of implicit information leakage?"

This can be understood by a simple example, illustrated below in Example 6, which shows a possible information leakage even in absence of DD- and FD-based leakages.

Example 6. Consider the relation in Table 5(a). Suppose that an attacker is able to observe the properties *Age_Group* and *Deficiency* from the attributes "*Age*" and "*Food_Habit*" according to following definitions:

$$Age\_Group(Age) = \begin{cases} child,\ if\ Age \in [0-15] \\ adult,\ if\ Age \in [15-40] \\ old,\ otherwise. \end{cases}$$

$$Deficiency(Food\_Habit) = \begin{cases} Vitamin\ A, & if\ (cord\ liver\ oil \vee yellow\ fruits) \notin Food\ Habit \\ Vitamin\ D, & if\ (milk \vee fish\ oil) \notin Food\ Habit \\ Iodine, & if\ (sea\ food \vee organic\ yoghurts \vee cranberries) \notin Food\ Habit \\ Protein, & if\ (pulses \vee fish) \notin Food\ Habit \end{cases}$$

Observe that there is no leakage of concrete database values due to FD or DD present in the system. In spite of that, an attacker may carefully observe

Table 5: Leakage based on property observation

(a) Relation "*Patient_diet*"

Name	ZIP	Age	Food_Habit	Disease
Alice	100025	4	cereals, carrot, fish oil, yoghurt	Kwashiorkar
Bob	110091	26	cereals, fish, yoghurts	Night Blindness
Carol	100025	39	pulses, yellow fruits, cranberries	Osteomalacia
David	123001	3	pulses, yellow fruits, cranberries	Rickets
Ela	110091	22	pulses, yellow fruits, cranberries	Osteomalacia
Frudo	110091	5	pulses, milk, yellow fruits	Goiter
Gram	123001	45	cereals, fish, yoghurts	Cataract

(b) Property-based Abstraction of "*Patient_diet*"

Name	ZIP	Age_Group	Deficiency	Disease_Effects
Alice	100025	Child	Protein	Distended abdomen
Bob	110091	Adult	Vitamin A	Vision problem in darkness
Carol	100025	Adult	Vitamin D	Joint pain, stiffness
David	123001	Child	Vitamin D	Stunted growth
Ela	110091	Adult	Vitamin D	Joint pain, stiffness
Frudo	110091	Child	Iodine	Swollen neck
Gram	123001	Old	Vitamin A	Complete loss of vision

the properties *Age_Group* and *Deficiency* of a person looking his or her *Age* and *Food_Habit* and may infer the property of disease by which he or she gets affected. This is due to the observational power of attackers [18] at property-level on the dependencies exhibited in the concrete level. This is depicted in Table 5(b).

Let us now formally define the *property-based dependency* (PD) and propose a preventive measure for this.

The formal definition is based on upper closure operator of Abstract Interpretation [19]. Abstract domains can be equivalently formulated either in terms of Galois Connections or Closure Operators. As usual, in Abstract Interpretation, a property is an upper closure operator on the concrete domain of computation.

Definition 3. ***Upper Closure Operator [19]:*** *An upper closure operator* $\rho : \mathcal{S} \to \mathcal{S}$ *on a poset* $\mathcal{S}$ *is (i) Monotone:* $\forall x_1, x_2 \in \mathcal{S}.\ x_1 \leq_{\mathcal{S}} x_2 \Rightarrow \rho(x_1) \leq_{\mathcal{S}} \rho(x_2)$. *(ii) Idempotent:* $\forall x \in \mathcal{S}.\ \rho^2(x) = \rho(x)$. *(iii) Extensive:* $\forall x \in \mathcal{S}.\ x \leq_{\mathcal{S}} \rho(x)$

The set of all closure operators on S is denoted by $uco(\mathtt{S})$. If $\langle \mathtt{H}, \leq, \sqcup, \sqcap, \bot, \top \rangle$ is a complete lattice, then $uco(\mathtt{H})$ is isomorphic to lattice of abstract interpretation of H. For example, the SIGN and PARITY properties [19] as represented in Figure 2 may act as the upper closure operators of the lattice H of integers.

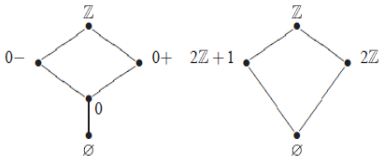


Fig. 2: SIGN and PARITY domain

Definition 4. ***Property-based dependency*** *(PD): Let a and b be two attributes of a relation R. Let C_a and C_b be the domains of a and b respectively, such that $r \subseteq C_a \times C_b$. Given $\rho \in uco(C_a)$ and $\eta \in uco(C_b)$, we say that there exists (ρ, η) - property-based dependency if either (1) $\forall\ t_i \in R:\ \rho(t_i(a))$ determines $\eta(t_i(b))$ or (2) $\forall t_i, t_j \in R$: if$(\rho(t_i(a)) = \rho(t_j(a)))$ then $(\eta(t_i(b)) = \eta(t_j(b)))$, where $t_i(a)$ denotes the value of attribute a in tuple t_i.*

In general, we denote PD dependency by the notation $\mathtt{P_{PD}}$: $\langle\!\langle \rho(\mathtt{X}) \xrightarrow{PD} \eta(\mathtt{Y}) \rangle\!\rangle$ where $\mathtt{X}, \mathtt{Y} \in \wp(\mathtt{attribute(DB)})$ and ρ and η are the respective closure operators. For example, in Example 6 the PD is represented as $\langle\!\langle$*Age_Group(Age), Deficiency(Food_Habit)* $\xrightarrow{PD}$ *Disease_Effects(Disease)*$\rangle\!\rangle$. This is worthwhile to mention here that the relation r on C_a and C_b represents either DD- or FD-dependency on the concrete relation R.

The above definitions can be extended for any number of attributes and the definition of the generalized form of PD-dependencies is given below:

Definition 5. ***Generalized Property-based Dependency****: The above definition can be generalized to $(\langle\rho_1, \rho_2, \ldots, \rho_k\rangle, \langle\eta_1, \eta_2, \ldots, \eta_k\rangle)$*-property-based dependency *if $\exists\ a_m, \ldots, a_n, b_u, \ldots, b_v \in$ **attribute**(R): (1) $\forall\ t_i \in R : \rho_1(t_i[a_m]) \wedge \cdots \wedge \rho_k(t_i[a_n])$ determines $\eta_1(t_i[b_u]) \wedge \cdots \wedge \eta_k(t_i[b_v])$ or (2) $\forall\ t_i, t_j \in R$: if $\rho_1(t_i[a_m]) = \rho_1(t_j[a_m]) \wedge \ldots \wedge \rho_k(t_i[a_n]) = \rho_k(t_j[a_n])$ then $\eta_1(t_i[b_u]) = \eta_1(t_j[b_u]) \wedge \cdots \wedge \eta_k(t_i[b_v]) = \eta_k(t_j[b_v])$.*

The existence of PD allows attackers to infer some partial information representing property of data, instead of their actual values. We can restrict the attacker to observe properties at a higher level of abstraction only, *e.g.* "Vitamins deficiency" instead of "Vitamin A" or "Vitamin D" deficiency so that the property based dependency $\langle\!\langle$*(Age_Group(Age), Deficiency(Food_Habit))* $\xrightarrow{PD}$ *Disease_Effects(Disease)*$\rangle\!\rangle$ is broken.

Considering all these three dependency-based information leakage, lets rewrite our formal policy specification language for implicit leakage below:

$$\mathtt{P_{IM}} ::= \mathtt{P_{DD}} \mid \mathtt{P_{FD}} \mid \mathtt{P_{PD}} \mid \mathtt{P_{IM}},\ \mathtt{P_{IM}}$$

where $\mathtt{P_{DD}}$: $[\mathtt{X} \xrightarrow{DD} \mathtt{Y}]$ is data dependency, $\mathtt{P_{FD}}$: $\langle \mathtt{X} \xrightarrow{FD} \mathtt{Y} \rangle$ is functional dependency, and $\mathtt{P_{PD}}$: $\langle\!\langle \rho(\mathtt{X}) \xrightarrow{PD} \eta(\mathtt{Y}) \rangle\!\rangle$ is *property-based dependency*, where $\mathtt{X}$, $\mathtt{Y} \in \wp(\mathtt{attribute(R)})$ and $\mathtt{X} \cap \mathtt{Y} = \phi$ and ρ, η are the respective closure operators.

8 Proposed Solution to Prevent Implicit Leakage

Although DD- and FD-dependences information are specified by the database designers, however the presence of possible *property based dependences* can be identified based on the database abstractions w.r.t. attackers observational power in abstract domains [18]. Due to space restriction, we do not detail this here.

In this section, we provide a unified solution framework to prevent implicit leakage in presence of all three dependences (i.e., DD, FD and PD). To this objective, we extend the Ordered Binary Decision Diagram (OBDD)-based approach which was primarily proposed in [20] to address explicit constraints in attribute-association form only. The idea behind this is to enable the approach to nullify the effect of dependencies.

Our solution consists of following phases: *a*) *representation of constraints in boolean logic*, *b*) *computing correct truth assignment of the OBDD of boolean formula*, *c*) *computing minimal correct truth assignments of the OBDDs.*

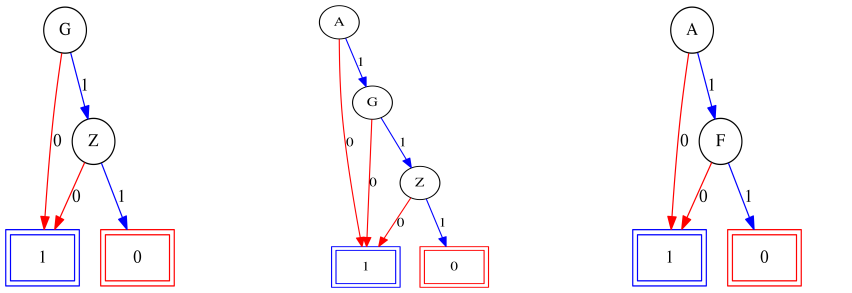
Representation of constraints in boolean logic. Let $\bar{x} \in \mathsf{Bvar}$ be a boolean formula corresponding to x where Bvar is the set of boolean variables. Given a relation R, any data dependency of the form $[\mathtt{X} \xrightarrow{DD} \mathtt{Y}]$ is represented as $(\bar{x_1} \wedge \cdots \wedge \bar{x_k}) \xrightarrow{DD} (\bar{y_1} \wedge \cdots \wedge \bar{y_l})$ for all $x_i \in \mathtt{X}$ and $y_i \in \mathtt{Y}$. Similarly functional dependency of form $\langle \mathtt{X} \xrightarrow{FD} \mathtt{Y} \rangle$ is represented as $\wedge x_i \xrightarrow{FD} \wedge y_i$ where $x_i \in \mathtt{X},\ y_i \in \mathtt{Y}$ the PD of the form $\langle\!\langle \rho(\mathtt{X}) \xrightarrow{PD} \eta(\mathtt{Y}) \rangle\!\rangle$ is represented as $\rho(\wedge x_i) \xrightarrow{PD} \eta(\wedge y_i)$.

Computing correct truth assignment of the OBDD of boolean formula. Once we get the boolean representation, we compute the correct truth assignment of the boolean formula. Let us consider a DD $[\mathtt{X} \xrightarrow{DD} \mathtt{Y}]$ and its corresponding boolean formula $(\bar{x_1} \wedge \cdots \wedge \bar{x_k}) \xrightarrow{DD} (\bar{y_1} \wedge \cdots \wedge \bar{y_l})$. If all the attributes in X belong to a single fragment then it may lead to implicit information leakage. Therefore, one-paths of the OBDD of the boolean formula $\neg(\bar{x_1} \wedge \cdots \wedge \bar{x_k})$ represent the truth assignment that breaks the dependency. The same holds true for FD and PD.

The formal definition of the correct truth assignment by considering both explicit and implicit policies is as follows:

Definition 6. ***Correct truth assignments*** *Given a relation R, let there exist a DD: $(\bar{x_1} \wedge \cdots \wedge \bar{x_k}) \xrightarrow{DD} (\bar{y_1} \wedge \cdots \wedge \bar{y_l})$, a FD: $(\bar{w_1} \wedge \cdots \wedge \bar{w_m}) \xrightarrow{FD} (\bar{z_1} \wedge \cdots \wedge \bar{z_n})$ and a PD: $\rho(\bar{q_1} \wedge \cdots \wedge \bar{q_i}) \xrightarrow{PD} \eta(\bar{r_1} \wedge \cdots \wedge \bar{r_j})$. The one-paths of the OBDDs of boolean formulas $\neg(\bar{x_1} \wedge \cdots \wedge \bar{x_k}), \neg(\bar{w_1} \wedge \cdots \wedge \bar{w_m}), \neg(\bar{q_1} \wedge \cdots \wedge \bar{q_i})$ represent the truth assignments that break the dependencies.*

Example 7. Consider the relation "*Patients*" in our running example depicted in Example 1. Suppose, there exists a data dependency D: $[\{Gender, ZIP\} \xrightarrow{DD} \{Age\}]$, a functional dependency F : $\langle \{Age, Gender, ZIP\} \xrightarrow{FD} \{Name\} \rangle$ and a property-based dependency P: $\langle\!\langle \{Age\_Group(Age), Deficiency(FoodHabit)\}$



(a) "OBDD1" (b) "OBDD2" (c) "OBDD3"

Fig. 3: OBDD for boolean formula of running example

$\xrightarrow{PD}$ $\{Disease\_Effects(Disease)\}\rangle\rangle$. The corresponding representation in boolean formula for these constraints are as follows: $\bar{\texttt{DD}}$: $(\overline{Gender} \wedge \overline{ZIP}) \rightarrow \overline{Age}$, $\bar{\texttt{FD}}$: $(\overline{Age} \wedge \overline{Gender} \wedge \overline{ZIP}) \rightarrow \overline{Name}$ and $\bar{\texttt{PD}}$: $(Age\_Group(\overline{Age}) \wedge Deficiency(\overline{Food\_Habit})) \rightarrow Disease\_Effects(\overline{Disease})$.

The OBDDs[1] representing the boolean formula $\neg(\overline{Gender} \wedge \overline{ZIP}), \neg(\overline{Age} \wedge \overline{Gender} \wedge \overline{ZIP})$ and $\neg(\overline{Age} \wedge \overline{Food\_Habit}))$ are depicted in Figure 3(a), 3(b) and 3(c) respectively. The nodes in the OBDDs are the attributes of "*Patients*" relation. The one-paths of these OBDDs represent the correct truth assignment that breaks the dependencies. In truth assignments, all variables not appearing in one-paths are considered as don't care variables represented as "-" depicted in Table 6.

Table 6: Truth assignments for one-paths

(a) Truth table for OBDD1

S	N	A	G	Z	F	D
-	-	-	0	-	-	-
-	-	-	1	0	-	-

(b) Truth table for OBDD2

S	N	A	G	Z	F	D
-	-	0	-	-	-	-
-	-	1	0	-	-	-
-	-	1	1	0	-	-

(c) Truth table for OBDD3

S	N	A	G	Z	F	D
-	-	0	-	-	-	-
-	-	1	-	-	0	-

Computing minimal correct truth assignments of the OBDDs. After computing OBDDs and correct truth assignments in step (b), the minimal set of truth assignments need to be calculated based the following two properties:

Definition 7. ***Link-able truth assignments*** *Given two assignments I_i and I_j over boolean variables Bvar in an OBDD, we say that I_i and I_j are link-able if and only if $\exists b \in$ Bvar: $I_i(b) = I_j(b) = 1$.*

Definition 8. ***Merge-able truth assignments*** *Given two assignments I_i and I_j over Bvar in an OBDD, we say that I_i and I_j are merge-able if and only if $\forall b \in$ Bvar s.t. $I_i(b) = 1$, $I_j(b) = 1$ or $I_j(b) =$ - and vice versa, where "-" denotes "don't care".*

[1] Generated by BDD interface **http://formal.cs.utah.edu:8080/pbl/BDD.php**

The heuristic algorithm proposed in [20] can easily be extended to compute minimal set of truth assignments for DD, FD and PD. This enables to fragment the relations nullifying the effect of dependencies.

Example 8. Recall the "*Patients*" relation of Example 1. Consider the truth assignments in Table 6. After applying the heuristic algorithm proposed in [20], the minimal set of truth assignments is {[-,-,-,0,-,-,-], [-,-,-,1,0,-,-], [-,-,1,1,0,0,-]}, resulting into a possible fragmentation which is depicted in Table 7. Observe that all the dependencies (*i.e.* DD, FD, PD) are broken and hence there is no implicit information leakage.

Table 7: Correct fragments preserving privacy concerns

(a) "Fragment 1"

Name	Age	Gender
Alice	4	M
Bob	26	F
Carol	39	M
David	3	F
Ela	22	M
Frudo	5	M
Gram	45	F

(b) "Fragment 2"

ZIP	Food_Habit	Disease
100025	cereals, carrot, fish oil, yoghurt	Kwashiorkor
110091	cereals, fish, yoghurts	Night Blindness
100025	pulses, yellow fruits, cranberries	Osteomalacia
123001	pulses, yellow fruits, cranberries	Rickets
110091	pulses, yellow fruits, cranberries	Ostemalacia
110091	pulses, milk, yellow fruits	Goiter
123001	cereals, fish, yoghurts	Cataract

9 Discussions

Although a large number of proposals exist in the literature, none of them is suitable to address privacy concerns expressed as a combination of different possible forms. Authors in [12] consider confidentiality constraints in cell-level form, whereas authors in [10] consider the same in selection-projection form only. In addition to confidentiality constraints, authors in [8] consider both confidentiality and visibility constraints in a different form of attribute association. The solutions proposed in [3–5], similarly, consider only confidentiality constraints but in the form of attribute-association. Unlike explicit information leakage, authors in [9–11] proposed solutions to implicit information leakage occurring due to the presence of various dependencies. Unfortunately, while authors in [9, 10] address the leakage due to functional dependency, the proposal in [11] separately deals with the same concern for data dependency only. In a different research direction, although the authors in [21–23] define a number of specification languages with different objectives such as providing user access control or encoding website privacy policies, however in context of relational databases, they can not be adapted.

Due to the lack of expressibility of privacy concerns in a generic way, no proposal in the literature is powerful enough to address different forms of constraints together and therefore does not fit to the real world scenarios. In this paper, we introduced a more generic privacy policy specification language considering a wide range of explicit and implicit privacy constraints. Besides, our proposed

solutions aim at addressing the privacy concerns expressed in our specification language. Furthermore, we introduced a new kind of dependency, property-based dependency in our specification language and provided a solution to address this. It is to be noted that the number of constraints affect the number of fragments. As the number of fragments increases, the utility of the data may be compromised.

10 Conclusions

This paper formalized a generic privacy-preserving policy specification language which is highly expressive to adapt a wide range of constraints in various forms, fitting to the real world scenarios. We identified that absence of data dependency and functional dependency does not guarantee the absence of indirect information leakage. In this context, we introduced a new form of dependency, known as "*Property-based Dependency*". Moreover, we proposed solutions to address privacy concerns expressed in our specification language. Currently, we are in the process of building a prototype based on our proposal.

References

1. Chen, D., Zhao, H.: Data security and privacy protection issues in cloud computing. In: Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on. Volume 1., IEEE (2012) 647–651
2. Bertino, E., Byun, J.W., Li, N.: Privacy-preserving database systems. In: FOSAD, Springer (2005) 178–206
3. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(05) (2002) 557–570
4. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) **1**(1) (2007) 3
5. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, IEEE (2007) 106–115
6. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragments and loose associations: Respecting privacy in data publishing. Proceedings of the VLDB Endowment **3**(1-2) (2010) 1370–1381
7. di Vimercati, S.D.C., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Extending loose associations to multiple fragments. In: IFIP Annual Conference on Data and Applications Security and Privacy, Springer (2013) 1–16
8. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Loose associations to increase utility in data publishing1. Journal of Computer Security **23**(1) (2015) 59–88
9. Wang, H.W., Liu, R.: Privacy-preserving publishing data with full functional dependencies. In: International Conference on Database Systems for Advanced Applications, Springer (2010) 176–183

10. Dong, B., Wang, W., Yang, J.: Secure data outsourcing with adversarial data dependency constraints. In: Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on, IEEE (2016) 73–78
11. di Vimercati, S.D.C., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Fragmentation in presence of data dependencies. IEEE Transactions on Dependable and Secure Computing **11**(6) (2014) 510–523
12. Wang, Q., Yu, T., Li, N., Lobo, J., Bertino, E., Irwin, K., Byun, J.W.: On the correctness criteria of fine-grained access control in relational databases. In: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment (2007) 555–566
13. Landberg, A.H., Nguyen, K., Pardede, E., Rahayu, J.W.: δ-dependency for privacy-preserving xml data publishing. Journal of biomedical informatics **50** (2014) 77–94
14. Farkas, C., Jajodia, S.: The inference problem: a survey. ACM SIGKDD Explorations Newsletter **4**(2) (2002) 6–11
15. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In: Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment (2002) 143–154
16. Paci, F., Zannone, N.: Preventing information inference in access control. In: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, ACM (2015) 87–97
17. Earley, J.: An efficient context-free parsing algorithm. Communications of the ACM **13**(2) (1970) 94–102
18. Mastroeni, I.: On the role of abstract non-interference in language-based security. In: Asian Symposium on Programming Languages and Systems, Springer (2005) 418–433
19. Giacobazzi, R., Mastroeni, I.: Abstract non-interference: Parameterizing non-interference by abstract interpretation. ACM SIGPLAN Notices **39**(1) (2004) 186–197
20. Ciriani, V., Di Vimercati, S.D.C., Foresti, S., Livraga, G., Samarati, P.: Enforcing confidentiality and data visibility constraints: An obdd approach. In: DBSec, Springer (2011) 44–59
21. Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S., Rjaibi, W.: Extending relational database systems to automatically enforce privacy policies. In: Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, IEEE (2005) 1013–1022
22. Iyilade, J., Vassileva, J.: P2u: A privacy policy specification language for secondary data sharing and usage. In: Security and Privacy Workshops (SPW), 2014 IEEE, IEEE (2014) 18–22
23. Cranor, L.F.: P3p: Making privacy policies more useful. IEEE Security & Privacy **99**(6) (2003) 50–55