OBSERVATION-BASED FINE GRAINED ACCESS CONTROL FOR RELATIONAL DATABASES

Raju Halder and Agostino Cortesi
Dipartimento di Informatica, Università Ca' Foscari di Venezia, Venezia, Italy
halder@unive.it, cortesi@unive.it

Keywords: Access control, Relational databases, Abstract interpretation.

Abstract: Fine Grained Access Control (FGAC) provides users the access to the non-confidential database information while preventing unauthorized leakage of the confidential data. It provides two extreme views to the database information: completely public or completely hidden. In this paper, we propose an Observation-based Fine Grained Access Control (OFGAC) mechanism based on the Abstract Interpretation framework where data are made accessible at various level of abstraction. In this setting, unauthorized users are not able to infer the exact content of a cell containing confidential information, while they are allowed to get partial information out of it, according to their access rights. Different level of sensitivity of the information correspond to different level of abstraction. In this way, we can tune different parts of the same database content according to different level of abstraction at the same time. The traditional FGAC can be seen as a special case of the OFGAC framework.

1 INTRODUCTION

Due to emerging trend of Internet and database technology, the information systems are shared by many people around the world. Also the businesses scenarios have become more dependent on information. As information usage proliferates among more and more users, organizations must deliver their data only to those people who are authorized for it. Thus, data accessibility to the authorized people is at the heart of the business processes, while on the other hand, information leakage to unauthorized people may lead to a huge loss in business.

Access control mechanism (Bertino et al., 1999; Jajodia et al., 1997; Griffiths and Wade, 1976) is one of the most effective solutions to ensure the safety of the information in an information system. The granularity of traditional access control mechanism is coarse-grained and can be applied on database or table level only. The need of more flexible business requirements and security policies mandate the use of Fine Grained Access Control (FGAC) mechanisms (Wang et al., 2007; Zhu et al., 2008; Zhu and Lü, 2007; Rizvi et al., 2004; Kabra et al., 2006) that provide the safety of the database information even at lower level such as individual tuple level or cell level. In general, FGAC aims at hiding the confidential information completely while giving the public access only to the non-confidential information. The proposed schemes on FGAC suggest to mask the confidential information by special symbols like NULL (LeFevre et al., 2004) or *Type*-1/*Type*-2 variables (Wang et al., 2007), or to execute the queries over the operational relations (Zhu et al., 2008; Shi et al., 2009) or authorized views (Rizvi et al., 2004; Kabra et al., 2006) etc. FGAC at database level can ensure to apply consistently to every user and every application.

The traditional FGAC policy provides only two extreme views: completely public or completely hidden. There are many application areas where some partial or relaxed view of the confidential information is desirable. For instance, if the database in an online transaction system contains credit card numbers for its customers, according to the disclosure policy, the employees of the customer-care section are able to see the last four digits of the credit card number whereas all the other digits must be completely hidden. The traditional FGAC policy is unable to implement this type of security framework without changing the database structure (*e.g.* by splitting the *Credit Card Number* attribute into two sub-attributes - one public, and the other private). To implement this relaxed scheme where an observer is allowed to observe specific properties or partial views of the private data, we introduce an Observation-based Fine Grained Access Control (OFGAC) mechanism on top of the traditional FGAC, based on the Abstract Inter-

pretation framework (Cousot and Cousot, 1977; Giacobazzi et al., 2000; Halder and Cortesi, 2010). We call the disclosure policy under OFGAC framework as observation-based disclosure policy.

Unlike traditional FGAC, the database information which are unauthorized under the observation-based disclosure policy *op*, are masked by the information at various level of abstraction representing some properties of interest. In other words, the confidential database information are masked by abstract values rather than NULL or special symbols (LeFevre et al., 2004; Wang et al., 2007). The unauthorized users, thus, could not be able to infer the exact content of the sensitive cells. Different level of sensitivity of the information correspond to different level of abstraction. Less sensitive values can be abstracted at lower level whereas higher sensitive information can be masked with abstract values at higher level of abstraction. In this way, we can tune different parts of the same database content according to different level of abstraction at the same time, giving rise to various observational access control for various part. Thus, in contrast to traditional FGAC, the OFGAC allows access control by splitting the database into multipart, where each part represents different level of sensitivity of the data.

The query issued by the external users will be directed to and executed over the abstract database which yield to a sound approximation of the query results. In particular, the result of a query contains all those data for which the precondition part of the query evaluates to either *true* or *unknown* logic values (Halder and Cortesi, 2010). Special care should be taken for the queries with aggregate functions (AVG, SUM, MIN, MAX, COUNT) and negation (MINUS, NOT EXISTS, NOT IN) to preserve soundness.

The traditional FGAC can be seen as a special case of our OFGAC framework.

The structure of the paper is as follows: Section 2 introduces the notion of observation-based disclosure policy. Section 3 discusses the possible multiparty collusion attacks under the same or different observation-based disclosure policies. In Sections 4 and 5 we discuss the referential integrity constraints issues and the query evaluation techniques under OFGAC. In Section 6 we survey the related work in the literature, and finally we draw our conclusions in Section 7.

2 OBSERVATION-BASED DISCLOSURE POLICIES

Whenever query Q is issued to a database, the traditional fine grained disclosure policy P determines which database information is allowed to disclose when answering the query Q. Given a database schema DB, the disclosure policy P splits the database states σ into two distinct parts: a public one (insensitive data) and a private one (sensitive data). Thus, under the disclosure policy P, we can represent the database state by a tuple $\sigma_P = \langle \sigma_h, \sigma_l \rangle$ where σ_h and σ_l represent the states which correspond to the private part and to the public part, respectively. Under policy P, the generic users are able to see the database information from the public part appearing in the query answer while the private part remains undisclosed. In reality, the disclosure policy P depends on the context in which the query is issued, for instance, the identity of the issuer, the purpose of the query, the data provider's policy etc. Given a database state σ_P under the policy P, and a query Q, the execution of Q over σ_P would return the query result $\xi = [\![Q]\!](\sigma_P)$ where the private part (σ_h) of σ_P is masked by special symbols (for example, NULL (LeFevre et al., 2004) or type-1/type-2 variable (Wang et al., 2007)). Thus, each cell in ξ either takes a constant value or special symbols (NULL or special variables) which indicates that the cell is unauthorized and can not be disclosed under P. As far as the security of the system is concerned, the disclosure policy should comply with the *non-interference* policy (Sabelfeld and Myers, 2003) *i.e.* the results of admissible queries should not depend on the confidential data in the database. The *non-interference* says that a variation of private (sensitive) database values does not cause any variation of the public (insensitive) view (see Definition 1).

Definition 1. *(Non-interference)*
Let $\sigma_P = \langle \sigma_h, \sigma_l \rangle$ and $\sigma'_P = \langle \sigma'_h, \sigma'_l \rangle$ be two database states under the disclosure policy P. The non-interference policy says that

$$\forall Q, \forall \sigma_P, \sigma'_P : \ \sigma_l = \sigma'_l \implies [\![Q]\!](\sigma_P) = [\![Q]\!](\sigma'_P)$$

That is, if the public (insensitive) part of any two database states under the disclosure policy P are the same, the execution of any admissible query Q over σ_P and σ'_P return the same results.

In the context of information flow security, the notion of non-interference is too restrictive and impractical in some real systems where intensional leakage of the information to some extent is allowed with the assumption that the power of the external observer is bounded. Thus, we need to weaken or downgrading the sensitivity level of the database information,

hence, the notion of non-interference which considers weaker attacker model. The weaker attacker model characterizes the observational characteristics of the attacker and can be able to observe specific properties of the private data.

Definition 2 defines the observation-based disclosure policy under the OFGAC framework.

Definition 2. *(Observation-based disclosure policy) Given a domain of observable properties D, and an abstraction function* $\alpha_D : \wp(val) \rightarrow D$*, an observation-based disclosure policy op assigned to the observer O is a tagging that assigns each value v in the database state* σ *a tag* $\alpha_D(v) \in D$*, meaning that O is allowed to access the value* $\alpha_D(v)$ *instead of its actual value v.*

Consider a database that consists of two tables as depicted in Table 1(a) and 1(b) respectively, containing information about employees and departments for an organization. Suppose the *names* of all employees, *phone numbers* of all departments, *age* and *salaries* of some employees are sensitive data. Cells containing sensitive data are marked with '*N*' within parenthesis. To implement the observational access control, we can mask or abstract those sensitive information by the abstract values representing specific properties of interest as depicted in Table 2(a) and 2(b) respectively. The corresponding abstract lattices for the attributes "*name*", "*sal*" and "*age*" are shown in Figure 1(a), 1(b) and 1(c) respectively. Observe that in $emp^\sharp$, the ages are abstracted by the elements from the domain of interval, the salaries are abstracted by the relative measures: *low*, *medium*, *high*, *very high* and the names are abstracted by their sex property. It is worthwhile to note that here we assume that salary is more sensitive than the age of employees and so, sensitive salary values are abstracted by a higher level of abstraction than that of age, although both are numeric data. The correspondence between the concrete values of salaries and the abstract values that partially hide sensitive salary values can be formally expressed by the abstraction and concretization functions α_{sal} and γ_{sal} respectively as follows:

$$\alpha_{sal}(\{\mu\}) \triangleq \begin{cases} low \;\; if\ \mu \in [500, 1999] \\ medium \;\; if\ \mu \in [2000, 3999] \\ high \;\; if\ \mu \in [4000, 5999] \\ very\ high \;\; if\ \mu \in [6000, 10000] \end{cases}$$

$$\gamma_{sal}(d) \triangleq \begin{cases} \{[a,b] : a \leq b,\ a \geq 500,\ b < 2000\}\ if\ d = low \\ \{[a,b] : a \leq b,\ a \geq 2000,\ b < 4000\}\ if\ d = medium \\ \{[a,b] : a \leq b,\ a \geq 4000,\ b < 6000\}\ if\ d = high \\ \{[a,b] : a \leq b,\ a \geq 6000,\ b < 10000\}\ if\ d = very\ high \end{cases}$$

Most importantly, in the abstract table $dept^\sharp$, since the phone numbers of all departments are strictly confidential they are abstracted by the top-most element $\top$ of their corresponding lattice. For the attribute *att*, the values which are completely public, we can consider the abstraction function α_{att} and concretization function γ_{att} as the identity function *id*.

Table 1: Database consists of table "*emp*" and "*dept*".

eID	*Name*	*Age*	*Dno*	*Sal*
1	Matteo (N)	30	2	2800 (N)
2	Pallab (N)	22	1	1500
3	Sarbani (N)	56 (N)	2	2300
4	Luca (N)	35	1	6700 (N)
5	Tanushree (N)	40	3	4900
6	Andrea (N)	52 (N)	1	7000 (N)
7	Alberto (N)	48	3	800
8	Mita (N)	29 (N)	2	4700 (N)

(a) Table "*emp*"

Dno	*Name*	*Loc*	*Phone*	*DmngrID*
1	Financial	Venice	111-1111 (N)	6
2	Research	Rome	222-2222 (N)	8
3	Admin	Treviso	333-3333 (N)	3

(a) Table "*dept*"

Table 2: Database consists of masked table "$emp^\sharp$" and "$dept^\sharp$" abstracted by different level of abstraction.

$eID^\sharp$	$Name^\sharp$	$Age^\sharp$	$Dno^\sharp$	$Sal^\sharp$
1	Male	30	2	Medium
2	Male	22	1	1500
3	Female	[50,59]	2	2300
4	Male	35	1	Very high
5	Female	[40,49]	3	4900
6	Male	[50,59]	1	Very high
7	Male	48	3	800
8	Female	[20,29]	2	High

(a) Abstract Table "$emp^\sharp$"

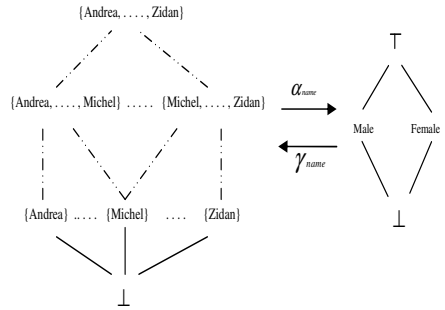
$Dno^\sharp$	$Name^\sharp$	$Loc^\sharp$	$Phone^\sharp$	$DmngrID^\sharp$
1	Financial	Venice	$\top$	6
2	Research	Rome	$\top$	8
3	Admin	Treviso	$\top$	3

(b) Abstract Table "$dept^\sharp$"

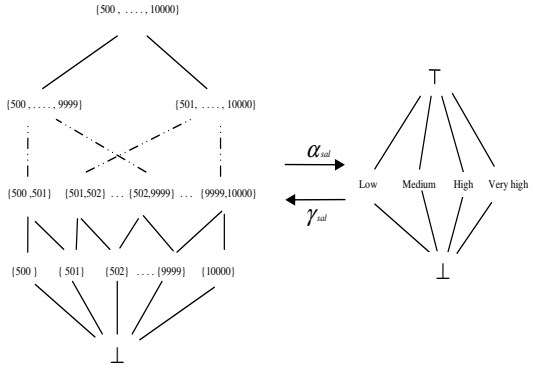
Observe that the traditional FGAC (LeFevre et al., 2004; Zhu et al., 2008; Shi et al., 2009) is a special case of our observation-based fine grained access control framework where each unauthorized cell is abstracted by the top-most element $\top$ of its corresponding abstract lattice.

A database is a collection of tables and a table t of arity k can be defined as $t \subseteq D_1 \times D_2 \times \cdots \times D_k$ where $attribute(t) = \{a_1, a_2, \ldots, a_k\}$ and attribute a_j ($j \in [1..k]$) corresponding to the typed domain D_j.

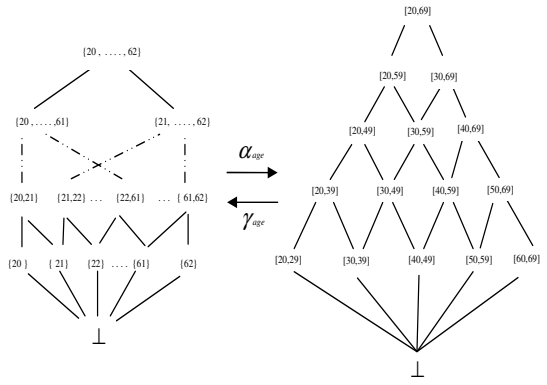
Given a database state σ without any access control policy. Under the OFGAC policy *op*, it is represented as σ_{op}. When observer O issues a query Q



(a) Abstract lattice for attribute "*name*"



(b) Abstract lattice for attribute "*sal*"



(c) Abstract lattice for attribute "*age*"

Figure 1: Abstract Lattices for attributes "*name*", "*sal*" and "*age*".

over σ_{op}, all the unauthorized cells of σ_{op} belonging to one or more attributes of the tables in the database are abstracted which results in an abstract database state $\sigma^{\sharp}_{op}$ and the abstract query $Q^{\sharp}$ obtained from Q is executed on this $\sigma^{\sharp}_{op}$. For each of these sensitive attributes there exists a corresponding abstract lattice. The value of the unauthorized cell c belonging to the attribute a in the database is abstracted by following the Galois Connection $(\wp(D_a^{con}), \alpha_a, \gamma_a, D_a^{abs})$ where $\wp(D_a^{con})$ represent the powerset of concrete domain of a whereas D_a^{abs} represents abstract domain of a, and α_a, γ_a are the abstraction and concretization function for a respectively. For insensitive attributes a, thus, we have the Galois Connection $(\wp(D_a^{con}), id, id, \wp(D_a^{con}))$ where id represents the identity function. For any sensitive attribute a, the abstraction and concretization function may be identity function when abstracting the cells of a containing public values. Thus when we consider the whole database state σ_{op} under the policy op, the abstract state is obtained by performing $\sigma^{\sharp}_{op} = \alpha(\sigma_{op})$ where abstraction function α can be expressed as collection of abstraction functions for all attributes in the database.

We assume that for each type of values in database there exists a hierarchy of abstractions such that Galois Connections combine consistently *i.e.* if $(X, \alpha_1, \gamma_1, Y)$ and $(Y, \alpha_2, \gamma_2, Z)$ represent two Galois Connection, then we have the following:

if $(X, \alpha_1, \gamma_1, Y)$ *and* $(Y, \alpha_2, \gamma_2, Z)$ *then* $(X, \alpha_2 \circ \alpha_1, \gamma_1 \circ \gamma_2, Z)$

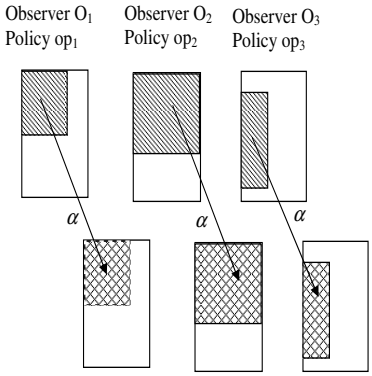
Definition 3. *Let σ_{op} be a database state under the observation-based disclosure policy op. Then $\sigma^{\sharp}_{op} = \alpha(\sigma_{op})$ where α is the abstraction function, is said to be abstract version of σ_{op} if there exists a representation function γ, called concretization function such that for all tuple $\langle x_1, x_2, \ldots, x_n \rangle \in \sigma_{op}$ there exists a tuple $\langle y_1, y_2, \ldots, y_n \rangle \in \sigma^{\sharp}_{op}$ such that $\forall i \in [1 \ldots n]$ $(x_i \in id(y_i) \vee x_i \in \gamma(y_i))$.*

In the example, the table "$emp^{\sharp}$" of Table 2(a) is a abstract version of "*emp*" of Table 1(a) since, for instance, for the tuples $\langle 6, Andrea, 52, 1, 7000 \rangle \in emp$ there exists $\langle 6, Male, [50,60], 1, Very\ high \rangle \in emp^{\sharp}$ such that $Andrea \in \gamma_{name}(Male)$, $52 \in \gamma_{age}([50,60])$, $7000 \in \gamma_{sal}(Very\ high)$ and for other values the concretization and abstraction functions represents identity function id. Similar for the other tuples.

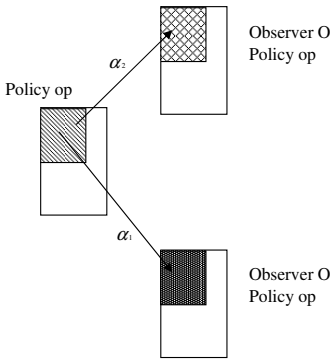
3 COLLUSION ATTACKS

Wang et al. in (Wang et al., 2007) illustrate the security of the system that implements traditional fine grained access control policy under the collusion and multi-query attack. They define the security aspect in *one-party single-query/weak security* and *multi-party multi-query/strong security* context and prove that the systems with weak-security is also secure under strong-security.

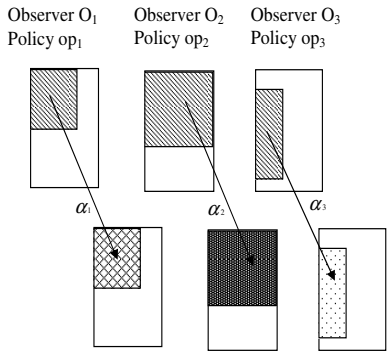
In observation-based fine grained access control policy, transforming abstract domains means transforming the attackers, and the attackers are modeled by abstractions. The robustness of the database under



(a) Case 1: Multiple policies/Single abstraction



(b) Case 2: Single policy/Multiple abstraction



(c) Case 3: Multiple policies-abstraction

Figure 2: Policies and Observations.

the OFGAC policy depends on the ability of the external observers to distinguish the database states based on the observable properties of the query results.

Here we consider three different scenarios: figure 2(a), 2(b) and 2(c) illustrates these three cases where the shaded portion indicates the sensitive information and α ($\alpha_i \neq \alpha_j$ if $i \neq j$) is the abstraction function used to abstract those sensitive information.

Case 1: Multiple Policies/Single Level Abstraction. Suppose each of the n observers under $op_1, op_2, \ldots, op_n$ respectively issues a query Q. Let σ be the database state without any policy. Under op_i, i=1,...,n the database state σ is represented as σ_{op_i} and is abstracted into $\sigma^{\sharp}_{op_i} = \alpha(\sigma_{op_i})$. Thus the observer O_i under op_i will get the query result $\xi^{\sharp}_i = [\![Q^{\sharp}]\!](\sigma^{\sharp}_{op_i})$ where $Q^{\sharp}$ is the abstract version of Q. When these n users collude, they feed the query results $\xi^{\sharp}_i$, $i = \ldots, n$ to a function f which can perform some comparison or computation (*viz*, difference operation) among the results and infer about some sensitive information for some observers.

For instance, suppose a portion of database information is sensitive under policy op_j while it is not sensitive under another policy op_k, $j \neq k$. For the first case this part of information will be abstracted while in the latter it will not. Thus, if this portion of information appears in both of the query results $\xi^{\sharp}_j$ and $\xi^{\sharp}_k$, then it is possible for the j^{th} observer to infer the exact content of that portion of information as it is not abstracted in ξ_k.

Let us explain this case with an example. Suppose the manager of a department can view all the details of employees working under him whereas the clerk under that manager can view only his own details. So, to the clerk all the details of other clerks (except him) under the same manager are abstracted and similarly, to the manager all the details of others except the employees under him are abstracted to the same level of abstraction. But if the manager and the clerk collude and share the query answers they obtain, then it is possible for the clerk to acquire some confidential information (confidential to him, but not to the manager) about the other clerks under the same manager.

Let $\sigma_{op} = \{\sigma_l, \sigma_h\}$ and $\sigma_{op'} = \{\sigma'_l, \sigma'_h\}$ be the database states under two different policies op and op'. The database state $\sigma_{op \bullet op'}$ obtained by combining two policies op and op' are defined as follows:

$$\sigma_{op \bullet op'} = \{((\sigma_l \cup \sigma_h) - (\sigma_h \cap \sigma'_h)), (\sigma_h \cap \sigma'_h)\}$$

This fact is depicted in Figure 3. So when the observers under op and op' collude and share the query results, both will act as equivalent to the observer under the policy $op \bullet op'$ and thus they can infer the values belonging to the public part of $op \bullet op'$ *i.e.* $((\sigma_l \cup \sigma_h) - (\sigma_h \cap \sigma'_h))$ by issuing a sequence of queries individually and by comparing the results together.

Case 2: Single Policy/Multiple Level Abstraction. Consider n different observers $O_1, O_2, \ldots, O_n$ under the same policy op and the sensitive

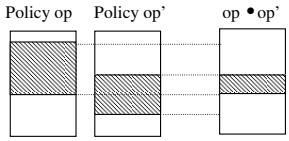


Figure 3: Combination of policies.

information part is abstracted to different level of abstraction to different observers. Higher levels of abstraction make the database information less precise whereas lower levels of abstraction represent them with more precision. Thus, the result of a query for the one with higher abstraction contains less precise information than that with lower abstraction.

Consider two different observers O_1 and O_2 under op where the sensitive database information of σ_{op} are abstracted by the domains of abstraction D_1^{abs} and D_2^{abs}, and results into $\sigma_{op}^{1\sharp}$ and $\sigma_{op}^{2\sharp}$ respectively.

First consider the case where D_2^{abs} is a more abstract domain than D_1^{abs}, *i.e.*, D_2^{abs} is an abstraction of D_1^{abs}. Since both observers are under the same policy, the query results over $\sigma_{op}^{1\sharp}$ and $\sigma_{op}^{2\sharp}$ may contain some common abstract information - one from D_1^{abs} and other from D_2^{abs}. Thus when O_1 and O_2 collude, it is possible for O_2 to obtain sensitive information with lower level of abstraction from the result obtained by O_1 as it is abstracted with lower level of abstraction for O_1. But no real collusion may arise in this case, as the overall information available to O_1 and O_2 together is at most as precise as the one already available to O_1.

The other case is where the two domains are not one the abstraction of the other. For example, let in a particular database state an attribute of a table have the sensitive values represented by an ordered list $\langle 5,0,2,3,1\rangle$. Suppose the observer O_1 is limited by the property *DOM* represented by domain of intervals as shown in Figure 4(a), while O_2 is limited by parity property represented by the abstract domain $PAR = \{\bot, EVEN, ODD, \top\}$ as depicted in Figure 4(c). Thus O_1 sees $\langle [4,5],[0,1],[2,3],[2,3],[0,1]\rangle$ while O_2 sees $\langle ODD, EVEN, EVEN, ODD, ODD\rangle$. When O_1 and O_2 collude they can infer the exact values for the attribute *i.e.* $\langle 5,0,2,3,1\rangle$ by combining the query results. The corresponding combined lattice obtained by combining the above two abstract lattices *DOM* and *PAR* is shown in Figure 5(a).

Given an OFGAC under *Single policy/Multiple level abstraction* scenario where same information under the same policy op is abstracted by n different level of abstraction to n different observers. Such OFGAC is *collusion-prone* when the intersection of the sets (not singletons) obtained by concretization of n different abstract values for the same sensitive cell appearing in the query results for n different observers, yield to a singleton. This is depicted in Definition 4.

We now show an example where no collusion takes place in practice. Consider the sensitive values $\langle -2,0,2,-1,1\rangle$. The observer O_1 and O_2 are limited by the sign property represented by the abstract domain $SIGN = \{\bot, +, -, \top\}$ and by the parity property represented by the abstract domain $PAR = \{\bot, EVEN, ODD, \top\}$ respectively. Thus O_1 sees $\langle -,+,+,-,+\rangle$ while O_2 sees $\langle EVEN, EVEN, EVEN, ODD, ODD\rangle$. Figure 4(b) and 4(c) shows the abstract lattices for *SIGN* and *PAR* respectively. When O_1 and O_2 collude they can infer the abstract values for the attribute as $\langle EVEN^-, EVEN^+, EVEN^+, ODD^-, ODD^+\rangle$ by combining the query results. However, although these combined abstract values represent more precise information than that of the individual queries, the observer still could not be able to infer the exact content. Figure 5(b) shows the combined abstract lattice obtained by combining two abstract lattices *SIGN* and *PAR*.

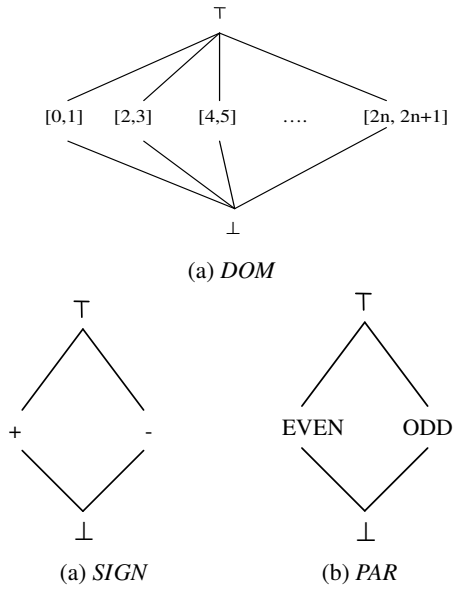
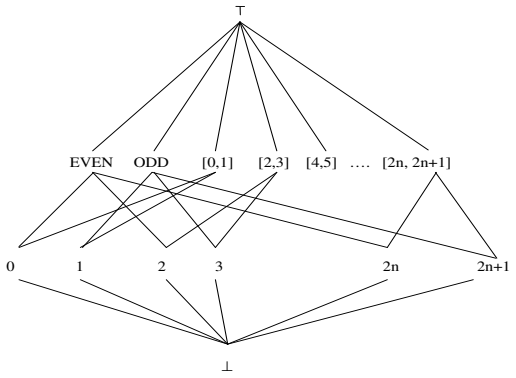
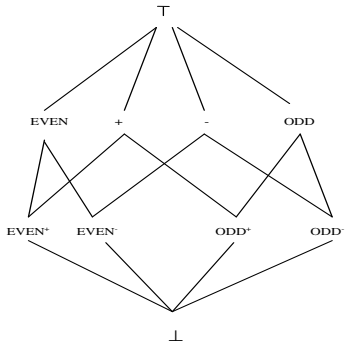


Figure 4: Abstract Lattices of *DOM*, *SIGN* and *PAR*.

Definition 4. *An OFGAC under Single policy/Multiple level abstraction scenario is collusion-prone for n different observers under the same policy op, if*



(a) Combined lattice of *DOM* and *PAR*



(b) Combined lattice of *SIGN* and *PAR*

Figure 5: Combination of lattices.

$\exists(d_1,d_2) \in D_1^{abs} \times D_2^{abs} : \gamma(d_1) \cap \gamma(d_2) = \{e\}$ *while* $\gamma(d_1) \neq \{e\}$, *and* $\gamma(d_2) \neq \{e\}$.

Theorem 1. *An OFGAC under Single policy/Multiple level abstraction scenario is collusion-free for n different observers under the same policy op, if*
$\forall(d_1,d_2) \in D_1^{abs} \times D_2^{abs}$, *if either* $\gamma(d_1)$ *or* $\gamma(d_2)$ *are not singletons, then* $\gamma(d_1) \cap \gamma(d_2)$ *is not a singleton.*

Theorem 2. *If the reduced product (Cousot and Cousot, 1977) of* D_1^{abs} *and* D_2^{abs} *is equivalent to the concrete domain D, then OFGAC is collusion-prone.*

Case 3: Multiple Policies / Level Abstractions.
This is the combination of the previous two cases. Observers may collude to act as the observer under the combination of their individual policies, or may try to infer about the confidential information appearing in the query results which are abstracted by different level of abstraction by combining (*e.g.* intersecting) their domain of abstract values.

4 PRESERVING REFERENTIAL INTEGRITY CONSTRAINTS BETWEEN DATABASE RELATIONS UNDER OFGAC

We know that the primary key values of a relation can not be NULL as they are used to identify individual tuple in the relation. The foreign key of the relation is used to maintain referential integrity constraints between relations of the database. Normalization of relations is mostly desirable operation to reduce redundant values and NULL values of the tuples and to disallow the possibility of generating spurious tuples. In case of normalization, the database relations are divided into multiple relations which are linkable through foreign key. However, we should keep in mind that the secure linking *i.e.* the referential integrity among the database relations can not be hampered by the abstraction operation of OFGAC policy. In (Wang et al., 2007), authors used *type*-2 variable to keep these referential integrity constraints intact while masking operation is performed. The attribute values which are sensitive and act as primary key or foreign key are masked by the *type*-2 variables.

We extend the same approach of Wang et al. (Wang et al., 2007) in our OFGAC where $Type-2$ variables are represented by tuple $\langle$*abstract value*, *variable*$\rangle$ where the variable can take only one concrete value from the set of values obtained by concretizing the abstract value. Formally, $Type-2$ variable is denoted by $\langle\beta,d\rangle$ where $d \in D^{abs}$ and β is a variable that can take its value from $\gamma(d)$. Definition 5 defines the $Type-2$ variables.

Definition 5. *(Type-2 Variable)*
A type-2 variable is represented by $\langle\beta,d\rangle$, *where* β *and d are the name and the abstract value of the variable, respectively. Given* β_1,β_2,d_1,d_2, "$\langle\beta_1,d_1\rangle$=$\langle\beta_1,d_1\rangle$" *and* "$\langle\beta_1,d_1\rangle \neq \langle\beta_2,d_1\rangle$" *are true, while whether* "$\langle\beta_1,d_1\rangle$=$\langle\beta_1,d_2\rangle$", "$\langle\beta_1,d_1\rangle$=$\langle\beta_2,d_2\rangle$", "$\langle\beta_1,d_1\rangle$=$c$ *are unknown, where c is a constant value.*

Example 1. *Consider the supplier-parts database and its abstract version under the observation-based access control policy as depicted in Table 3. The attributes S-id and P-id are the primary key for the tables "Supplier" and "Part" respectively whereas the composite attribute {S-id, P-id} is used as the primary key for the table "Supp-Part". Observe that S-id and P-id in "Supp-Part" table are used as the foreign key that link to the primary key of "Supplier" and "Part" respectively and relate the suppliers with the parts sold by them. Suppose, according to the policy the values of the attribute S-id, P-id and some*

Table 3: Preserving Referential Integrity Constraint by using $Type-2$ variable.

$S-id$	$Name$	Age
S230 (N)	Alice	24
S201 (N)	Bob	21
S368 (N)	Tea	22

(a) Table "$Supplier$"

$S-id$	$P-id$	QTY
S230 (N)	P140 (N)	120
S201 (N)	P329 (N)	260 (N)
S230 (N)	P563 (N)	200
S368 (N)	P329 (N)	450 (N)
S368 (N)	P140 (N)	430 (N)

(b) Table "$Supp-Part$"

$P-id$	$Pname$
P140 (N)	Screw
P329 (N)	Bolt
P563 (N)	Nut

(c) Table "$Part$"

$S-id^\sharp$	$Name^\sharp$	$Age^\sharp$
$\langle\beta_1,[S200,S249]\rangle$	Alice	24
$\langle\beta_2,[S200,S249]\rangle$	Bob	21
$\langle\beta_3,[S350,S399]\rangle$	Tea	22

(d) Abstract Table "$Supplier^\sharp$"

$S-id^\sharp$	$P-id^\sharp$	$QTY^\sharp$
$\langle\beta_1,[S200,S249]\rangle$	$\langle\beta_4,[P100,P149]\rangle$	120
$\langle\beta_2,[S200,S249]\rangle$	$\langle\beta_5,[P300,P349]\rangle$	[250,299]
$\langle\beta_1,[S200,S249]\rangle$	$\langle\beta_6,[P550,P599]\rangle$	200
$\langle\beta_3,[S350,S399]\rangle$	$\langle\beta_5,[P300,P349]\rangle$	[450,499]
$\langle\beta_3,[S350,S399]\rangle$	$\langle\beta_4,[P100,P149]\rangle$	[400,449]

(e) Abstract Table "$Supp-Part^\sharp$"

$P-id^\sharp$	$Pname^\sharp$
$\langle\beta_4,[P100,P149]\rangle$	Screw
$\langle\beta_5,[P300,P349]\rangle$	Bolt
$\langle\beta_6,[P550,P599]\rangle$	Nut

(f) Abstract Table "$Part^\sharp$"

QTY in "Supp-Part" table are confidential (marked with N in parenthesis). If we abstract S-id and P-id in "Supp-Part" only by the abstract values from the domain of intervals, we may loose the ability to identify the tuples uniquely and also the secure linking between "Supplier" and "Part" as well. To allow the linking and to preserving the uniqueness of values over abstract domain we use Type-2 variable while abstracting the confidential information as depicted in abstract tables "Supplier$^\sharp$", "Part$^\sharp$" and "Supp-Part$^\sharp$" in Table 3. To perform this we follow the labeling algorithm of (Wang et al., 2007). Observe that since the attribute QTY is not primary key or foreign key we abstract them only by the abstract values from the domain of intervals.

5 QUERY EVALUATION UNDER OFGAC

A general framework for Abstract Interpretation of Relational Databases has been introduced in (Halder and Cortesi, 2010). Here, we briefly recall some notions on query abstraction, and we extend them by considering queries on multiple abstractions as well.

Consider the following query Q_1 issued by an external user under the observational disclosure policy op:

$$Q_1 = SELECT\ *\ FROM\ emp\ WHERE\ Sal > 4800;$$

The original query above will be transformed into the following abstract query under the policy op:

$$Q_1^\sharp = SELECT\ *\ FROM\ emp^\sharp\ WHERE\ Sal^\sharp > 4800;$$

The result of $Q_1^\sharp$ is depicted in Table 4. Observe that for the first three tuples in the result, the condition in the $WHERE$ clause evaluates to $true$ whereas for the last tuple it evaluates to *unknown* (may be $true$ or may be $false$) logic value because x may be less, equal or greater than 4800 where $x \in \gamma(high)$. The result of $Q_1^\sharp$ is *sound* (Halder and Cortesi, 2010) as it over-approximate the result of the query Q_1. Observe in fact that $Q_1^\sharp$ includes also the "false positive" corresponding to the concrete information about $Mita$.

Table 4: The result of the query $Q_1^\sharp$.

$eID^\sharp$	$Name^\sharp$	$Age^\sharp$	$Dno^\sharp$	$Sal^\sharp$
4	Male	35	1	Very high
5	Female	[40,49]	3	4900
6	Male	[50,59]	1	Very high
8	Female	[20,29]	2	High

We denote any SQL command by a tuple $Q \triangleq \langle A_{sql}, \phi\rangle$. We call the first component A_{sql} the *active part* and the second component ϕ the *passive part* of Q. In an abstract sense, any SQL command Q first identifies an active data set from the database using the pre-condition ϕ and then performs the appropriate operations on that data set using the SQL action A_{sql}. The pre-condition ϕ appears in SQL commands as a well-formed formula in first-order logic (Halder and Cortesi, 2010).

Thus the evaluation of the pre-condition $\phi^\sharp$ of the abstract query over the abstract database may results into three logic values: $true$, $false$ or *unknown*. The evaluated value $true$ indicates that the tuple satisfies the semantic structure of $\phi^\sharp$ and $false$ indicates that the tuple does not satisfy $\phi^\sharp$. The logic value *unknown* indicates that the tuple may or may not satisfy the semantic structure of $\phi^\sharp$.

Given any abstract query $Q^\sharp$ and abstract database state $\sigma_{op}^\sharp$ under the observation-based policy op, the result for the query can, thus, be denoted by a tuple

$$\xi^\sharp = [\![Q^\sharp]\!](\sigma_{op}^\sharp) = \{\xi_{yes}^\sharp, \xi_{may}^\sharp\}$$

where $\xi_{yes}^\sharp$ is the part of the query result for which semantic structure of $\phi^\sharp$ evaluates to true and

$\xi^\sharp_{may}$ represents the remaining part for which $\phi^\sharp$ evaluates to unknown. Observe that we assume $\xi^\sharp_{yes} \cap \xi^\sharp_{may} = \emptyset$. For example, in the query result for $Q^\sharp_1$ as depicted in the Table 4, the first three tuples belong to $\xi^\sharp_{yes_1}$ whereas the last tuple belongs to $\xi^\sharp_{may_1}$.

Now consider the following query $Q^\sharp_2$:

$Q^\sharp_2 = SELECT\ *\ FROM\ emp^\sharp\ WHERE\ Age^\sharp\ BETWEEN\ 32\ AND\ 55;$

The result of $Q^\sharp_2$ is depicted in Table 5 where the tuples with age values 35, 40 and 48 belong to $\xi^\sharp_{yes_2}$ and the remaining tuples belong to $\xi^\sharp_{may_2}$.

Table 5: The result of the query $Q^\sharp_2$.

$eID^\sharp$	$Name^\sharp$	$Age^\sharp$	$Dno^\sharp$	$Sal^\sharp$
3	Female	[50,59]	2	2300
4	Male	35	1	Very high
5	Female	[40,49]	3	4900
6	Male	[50,59]	1	Very high
7	Male	48	3	800

5.1 Queries with Aggregate Functions

Let us apply the aggregate functions in the query $Q^\sharp_2$ which yield to the modified query $Q^\sharp_3$ as follows:

$Q^\sharp_3 = SELECT\ COUNT^\sharp(*),\ AVG^\sharp(Age^\sharp)$
$FROM\ emp^\sharp\ WHERE\ Age^\sharp\ BETWEEN\ 32\ AND\ 55;$

The query result of $Q^\sharp_3$ is depicted in Table 6.

Table 6: The result of the query $Q^\sharp_3$.

$COUNT^\sharp(*)$	$AVG^\sharp(Age^\sharp)$
[3,5]	[41.33,48.2]

To illustrate the execution of $Q^\sharp_3$ we first define some functions: let the elements of the domain of intervals are represented by $[x,y]$ where $x,y \in Z \wedge x \leq y$, and $\xi^\sharp = \{\xi^\sharp_{yes}, \xi^\sharp_{may}\}$ be the result of the query $Q^\sharp$ over the abstract database state $\sigma^\sharp_{op}$ under OFGAC policy op. Table 7 defines some essential functions over the concrete/abstract elements.

We are now in position to illustrate the execution of the aggregate functions involved in $Q^\sharp_3$.

1. $COUNT^\sharp(*)$**:** The result for $COUNT^\sharp(*)$ is represented by the abstract value $[a,b]$ where $a = count(\xi^\sharp_{yes})$ and $b = count(\xi^\sharp_{yes} \cup \xi^\sharp_{may})$.

   In the result of $Q^\sharp_3$, three tuples satisfy the precondition and belong to $\xi^\sharp_{yes_3}$ whereas remaining two tuples belong to $\xi^\sharp_{may_3}$. Thus the result of $COUNT^\sharp(*)$ is represented by the abstract value $[3,5]$. It means the count of the tuples in the result of $Q^\sharp_3$ is any value within the interval of 3 and 5.

Table 7: The functions over the concrete/abstract domain.

$min([x,y])$	returns x
$max([x,y])$	returns y
$average(\langle x,y\rangle)$	returns the average of x and y
$average(\langle[x,y],z\rangle)$	$[average(\langle x,z\rangle), average(\langle y,z\rangle)]$
$average(\langle[x,y],[w,z]\rangle)$	$[average(\langle x,w\rangle), average(\langle y,z\rangle)]$
$summation(\langle x,y\rangle)$	returns the sum of x and y
$summation(\langle[x,y],z\rangle)$	$[summation(\langle x,z\rangle), summation(\langle y,z\rangle)]$
$summation(\langle[x,y],[w,z]\rangle)$	$[summation(\langle x,w\rangle), summation(\langle y,z\rangle)]$
$minimum(\langle x,y\rangle)$	returns the minimum between x and y
$minimum(\langle[x,y],z\rangle)$	$[min([x,z]), min([y,z])]$
$minimum(\langle[x,y],[w,z]\rangle)$	$[min([x,w]), min([y,z])]$
$maximum(\langle x,y\rangle)$	returns the maximum between x and y
$maximum(\langle[x,y],z\rangle)$	$[max([x,z]), max([y,z])]$
$maximum(\langle[x,y],[w,z]\rangle)$	$[max([x,w]), max([y,z])]$
$\xi^\sharp(att^\sharp)$	returns the list of values of the attribute $att^\sharp$ appearing in the query result $\xi^\sharp$
$count(\xi^\sharp)$	returns the number of tuples in $\xi^\sharp$

2. $AVG^\sharp(Age^\sharp)$**:** The average value of ages over the abstract domain (represented by the domain of intervals) is denoted by $AVG^\sharp(Age^\sharp) = [min(a), max(b)]$ where $a = average(\xi^\sharp_{yes}(Age^\sharp))$ is the average value of those ages for which the precondition evaluates to *true* and $b = average(\xi^\sharp_{yes}(Age^\sharp) \cup \xi^\sharp_{may}(Age^\sharp))$ represents the average value of the ages for which precondition evaluates to either *true* or *unknown*.

   In the result of $Q^\sharp_3$, thus, we have a = [a_1, a_2] = $average$(35, [40,49], 48) = [41.33, 44] where a_1 = $average$(35, 40, 48) = 41.33 and a_2 = $average$(35, 49, 48) = 44. Similarly, b = [b_1, b_2] = $average$([50,59], 35, [40,49], [50,59], 48) = [44.6, 48.2] where b_1 = $average$(50, 35, 40, 50, 48) = 44.6 and b_2 = $average$(59, 35, 40, 59, 48) = 48.2.

   Thus $AVG^\sharp(Age^\sharp)$ = $[min(a), max(b)]$ = [41.33, 48.2] where $min(a) = min([41.33,44]) = 41.33$ and $max(b) = max([44.6,48.2]) = 48.2$.

Similarly, we can define the computation of other aggregate functions as follows:

- $SUM^\sharp(att^\sharp) = [min(a), max(b)]$,
  where $a = summation(\xi^\sharp_{yes}(att^\sharp)) \wedge b =$

$summation(\xi^{\sharp}_{yes}(att^{\sharp}) \cup \xi^{\sharp}_{may}(att^{\sharp}))$

- $MAX^{\sharp}(att^{\sharp}) = [max(a), max(b)]$, where $a = maximum(\xi^{\sharp}_{yes}(att^{\sharp})) \wedge b = maximun(\xi^{\sharp}_{yes}(att^{\sharp}) \cup \xi^{\sharp}_{may}(att^{\sharp}))$

- $MIN^{\sharp}(att^{\sharp}) = [min(a), min(b)]$, where $a = minimum(\xi^{\sharp}_{yes}(att^{\sharp})) \wedge b = minimum(\xi^{\sharp}_{yes}(att^{\sharp}) \cup \xi^{\sharp}_{may}(att^{\sharp}))$

5.2 Query with Negation *i.e.* MINUS, NOT IN, NOT EXISTS

Consider the query of the form $Q^{\sharp} = Q^{\sharp}_1 - Q^{\sharp}_2$. Let the query result for $Q^{\sharp}_1$ is $\xi^{\sharp}_1 = \langle \xi^{\sharp}_{yes_1}, \xi^{\sharp}_{may_1} \rangle$. Similarly, the query result for the query $Q^{\sharp}_2$ is $\xi^{\sharp}_2 = \langle \xi^{\sharp}_{yes_2}, \xi^{\sharp}_{may_2} \rangle$. The query result for $Q^{\sharp}$ after performing difference operation between the result of $Q^{\sharp}_1$ and $Q^{\sharp}_2$ is as follows:

$$\xi^{\sharp} = \langle \xi^{\sharp}_{yes_1} - (\xi^{\sharp}_{yes_2} \cup \xi^{\sharp}_{may_2}), \xi^{\sharp}_{may_1} - \xi^{\sharp}_{yes_2} \rangle$$

Observe that to ensure the soundness of the result of the query involving negation the first component $\xi^{\sharp}_{yes_1} - (\xi^{\sharp}_{yes_2} \cup \xi^{\sharp}_{may_2})$ contains those tuples for which the precondition strictly evaluates to *true* whereas for the second component $\xi^{\sharp}_{may_1} - \xi^{\sharp}_{yes_2}$ the precondition may evaluate to either *true* or *unknown*.
Let us illustrate with an example. Consider the following query:

$$Q^{\sharp} = Q^{\sharp}_1 - Q^{\sharp}_2$$

where,

$$Q^{\sharp}_1 = SELECT * FROM\ emp^{\sharp}\ WHERE\ sal^{\sharp} > 2500;$$

and

$$Q^{\sharp}_2 = SELECT * FROM\ emp^{\sharp}\ WHERE\ sal^{\sharp} > 5500;$$

The query results for $Q^{\sharp}_1$ and $Q^{\sharp}_2$ are depicted in Table 8(a) and 8(b) respectively. In Table 8(a), for the first tuple the pre-condition of $Q^{\sharp}_1$ evaluates to unknown (thus, belongs to $\xi^{\sharp}_{may_1}$) whereas for the remaining four tuples it evaluates to true logic value (thus, belongs to $\xi^{\sharp}_{yes_1}$). Similarly, in Table 8(b), for the first two tuples the pre-condition of $Q^{\sharp}_2$ evaluates to true (hence, belongs to $\xi^{\sharp}_{yes_2}$) whereas for the last one it evaluates to unknown logic value (hence, belongs to $\xi^{\sharp}_{may_2}$). Thus the first component $\xi^{\sharp}_{yes}$ of the result of $Q^{\sharp}$ is obtained by MINUS-ing $(\xi^{\sharp}_{yes_2} \cup \xi^{\sharp}_{may_2})$ from $\xi^{\sharp}_{yes_1}$ which consists of the tuple with $eID^{\sharp}$=5 and the second component $\xi^{\sharp}_{may}$ of the result of $Q^{\sharp}$ is obtained by MINUS-ing $\xi^{\sharp}_{yes_2}$ from $\xi^{\sharp}_{may_1}$ which consists of the tuple with $eID^{\sharp}$=1 as shown in Table 8(c).

Table 8: Query results for $Q^{\sharp}_1$, $Q^{\sharp}_2$ and $Q^{\sharp}$.

$eID^{\sharp}$	$Name^{\sharp}$	$Age^{\sharp}$	$Dno^{\sharp}$	$Sal^{\sharp}$
1	Male	30	2	Medium
4	Male	35	1	Very high
5	Female	[40,49]	3	4900
6	Male	[50,59]	1	Very high
8	Female	[20,29]	2	High

(a) Query result for $Q^{\sharp}_1$

$eID^{\sharp}$	$Name^{\sharp}$	$Age^{\sharp}$	$Dno^{\sharp}$	$Sal^{\sharp}$
4	Male	35	1	Very high
6	Male	[50,59]	1	Very high
8	Female	[20,29]	2	High

(b) Query result for $Q^{\sharp}_2$

$eID^{\sharp}$	$Name^{\sharp}$	$Age^{\sharp}$	$Dno^{\sharp}$	$Sal^{\sharp}$
1	Male	30	2	Medium
5	Female	[40,49]	3	4900

(c) Query result for $Q^{\sharp}$

Observe that the treatment of queries under our OFGAC framework extends the corresponding scenario discussed in (Wang et al., 2007; Zhu et al., 2008; Shi et al., 2009), that can be seen as a special case when the only abstraction considered on to NULL (and the "may" part of the query results is always empty).

6 RELATED WORKS

In (Wang et al., 2007) the authors proposed a formal notion of correctness in fine-grained database access control. They show why the existing approaches (LeFevre et al., 2004) fall short in some circumstances with respect to soundness and security requirements, like when a query contains any negation, as expressed using the keywords MINUS, NOT EXISTS or NOT IN. Moreover, they proposed a labeling approach for masking unauthorized information by using two types of special variables as well as a secure and sound query evaluation algorithm in case of cell-level disclosure policies.

In (Zhu et al., 2008; Shi et al., 2009), the authors observe that the proposed algorithm in (Wang et al., 2007) is unable to satisfy the soundness property for the queries which include the negation NOT IN or NOT EXISTS. They divide the approaches which implement FGAC into two: FGAC-then-enforced (FTE) and FGAC-first-enforced (FFE). In the first case *i.e.* FTE, three types of information leakages may take place: Truth Value Information Leakage, Range Information Leakage and UnNull Information Leakage. They propose an enforcing rule to control the information leakage by using FFE approach where the

query is executed on an operational relation rather than the original relation. The operational relation is obtained from the original one by disregarding those tuples in the original relation for which the value of the sensitive attributes (the attributes occur in the conditional part of the query) are not authorized. However, although the algorithm for Enforcing Rule satisfies the soundness and security properties for all SQL queries, it would not reach the maximum property.

The authors in (Böttcher et al., 2008) consider the environment where the secret information is shared among multiple partners and has a strong possibility for the secret information to be inferred from database queries and disclose to a third party illegally. They model the secret information by an existentially quantified boolean query. They have presented a formal model of secret information disclosure that defines a query to be suspicious if and only if the disclosed secret could be inferred from its answer.

Agrawal et al. (Agrawal et al., 2005) introduced the syntax of a fine grained restriction command at column level, row level, or cell level. The purposes and/or recipients for which the access is allowed can also be specified in the restriction. Multiple restriction can be combined by performing Intersection or Union of all restrictions. The enforcement algorithm automatically combines the restrictions relevant to individual queries annotated with purpose and recipient information and transforms the user's query into an equivalent query over a dynamic view that implements the restriction.

In (Zhu and Lü, 2007) the authors extend the SQL language to express the FGAC security policies. Many policy instances of a policy type can be created when needed. The created policy statement takes at least two parameters: subjects and target. The subject can be user, role or users in a group whereas target specifies a table or view or columns in the table. Moreover it specifies the operations on the objects that the security policy restricts and the filter list that specifies the data to be accessed in the specific objects. Finally it has constraint expressions whose truth value determine whether the policy will be executed or not.

Rizvi et al. in (Rizvi et al., 2004) described two models for fine-grained access control: the Truman and Non-Truman models. Both models support authorization-transparent querying. They defined the notions of unconditional and conditional validity of the query, and presented several inference rules for validity. They outlined an approach to validity testing, based on extending an existing query optimizer to carry out validity checking, minimizing the extra effort required during coding as well as during validity testing.

In the view replacement approach, the base relations in a query submitted by a user are replaced by authorized views. The original query is added with additional predicates/joins that ensure that the query accesses only authorized tuples. In such cases the additional authorization checks introduced by view replacement would be redundant. In (Kabra et al., 2006), authors described the set of transformation rules for redundancy removal. They defined when a query plan is safe with respect to user defined functions (UDFs) and other unsafe functions (USFs), and proposed techniques to generate safe query plans. However, this safe query plan may yield to unoptimized plan. The study also showed that leakage of information through USFs, exceptions and error messages can be efficiently tackled by choosing good safe plans.

In (Hsu et al., 2002), authors present a quantitative model for privacy protection. In the model, a formal representation of the user's information states is given, and they estimate the value of information for the user by considering a specific user model. Under the user model, the privacy protection task is to ensure that the user cannot profit from obtaining the private information. They further define the usefulness of information based on how easy the data user can locate individuals that fit the description given in the queries. The knowledge states and the usefulness of information can be changed or refined by receiving some answer to the user's query.

7 CONCLUSIONS

In this paper we introduce an Observation-based Fine Grained Access Control (OFGAC) framework on top of the traditional FGAC where the confidential information in the database are abstracted by their observable properties and the external observers are able to see this partial or abstract view of the confidential information rather than their exact contents. The traditional FGAC can be seen as a special case of our OFGAC, where the confidential information are abstracted by the top-most element of the corresponding abstract lattices.

ACKNOWLEDGEMENTS

Work partially supported by Italian MIUR COFIN'07 project "SOFT" and by RAS project TESLA - Tecniche di enforcement per la sicurezza dei linguaggi e delle applicazioni.

REFERENCES

Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S., and Rjaibi, W. (2005). Extending relational database systems to automatically enforce privacy policies. In *Proceedings of the 21st International Conference on Data Engineering, ICDE '05*, pages 1013–1022. IEEE Computer Society.

Bertino, E., Jajodia, S., and Samarati, P. (1999). A flexible authorization mechanism for relational data management systems. *ACM Transactions on Information Systems*, 17(2):101–140.

Böttcher, S., Hartel, R., and Kirschner, M. (2008). Detecting suspicious relational database queries. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security, ARES '08*, pages 771–778. IEEE Computer Society.

Cousot, P. and Cousot, R. (1977). Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, CA, USA. ACM Press.

Giacobazzi, R., Ranzato, F., and Scozzari, F. (2000). Making abstract interpretations complete. *Journal of the ACM (JACM)*, 47(2):361–416.

Griffiths, P. P. and Wade, B. W. (1976). An authorization mechanism for a relational database system. *ACM Transactions on Database Systems*, 1(3):242–255.

Halder, R. and Cortesi, A. (2010). Abstract interpretation for sound approximation of database query languages. In *Proceedings of the IEEE 7th International Conference on INFOrmatics and Systems (INFOS2010), Advances in Data Engineering and Management Track*, pages 53–59, Cairo, Egypt. IEEE Catalog Number: IEEE CFP1006J-CDR.

Hsu, T.-s., Liau, C.-J., Wang, D.-W., and Chen, J. K.-P. (2002). Quantifying privacy leakage through answering database queries. In *Proceedings of the 5th International Conference on Information Security, ISC '02*, pages 162–176, London, UK. Springer-Verlag.

Jajodia, S., Samarati, P., Subrahmanian, V. S., and Bertino, E. (1997). A unified framework for enforcing multiple access control policies. *SIGMOD Record*, 26(2):474–485.

Kabra, G., Ramamurthy, R., and Sudarshan, S. (2006). Redundancy and information leakage in fine-grained access control. In *Proceedings of the ACM SIGMOD international conference on Management of data, SIGMOD '06*, pages 133–144, Chicago, IL, USA. ACM Press.

LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y., and DeWitt, D. (2004). Limiting disclosure in hippocratic databases. In *Proceedings of the 30th international conference on Very large data bases, VLDB '04*, pages 108–119. VLDB Endowment.

Rizvi, S., Mendelzon, A., Sudarshan, S., and Roy, P. (2004). Extending query rewriting techniques for fine-grained access control. In *Proceedings of the ACM SIGMOD international conference on Management of data, SIGMOD '04*, pages 551–562, Paris, France. ACM Press.

Sabelfeld, A. and Myers, A. C. (2003). Language-based information-flow security. *IEEE Journal on selected areas in Communications*, 21(1):5–19.

Shi, J., Zhu, H., Fu, G., and Jiang, T. (2009). On the soundness property for sql queries of fine-grained access control in dbmss. In *ICIS '09: Proceedings of the 2009 Eigth IEEE/ACIS International Conference on Computer and Information Science*, pages 469–474, Shanghai, China. IEEE Computer Society.

Wang, Q., Yu, T., Li, N., Lobo, J., Bertino, E., Irwin, K., and Byun, J.-W. (2007). 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 '07*, pages 555–566, Vienna, Austria. VLDB Endowment.

Zhu, H. and Lü, K. (2007). Fine-grained access control for database management systems. In *Proceedings of the 24th British National Conference on Databases*, pages 215–223, Glasgow, UK. Springer Verlag LNCS.

Zhu, H., Shi, J., Wang, Y., and Feng, Y. (2008). Controlling information leakage of fine-grained access model in dbmss. In *Proceedings of the 9th International Conference on Web-Age Information Management, WAIM '08*, pages 583–590, Zhangjiajie, China. IEEE Computer Society.